DS745\_SentimentalAnalysis\_BHOWMICK

Gautam

December 9, 2018

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

# Install and load the following packages if not installed  
#install.packages("twitteR")  
#install.packages("openssl")  
#install.packages("httpuv")  
#install.packages("RCurl")  
#install.packages("tm")  
#install.packages("wordcloud")  
#install.packages("SnowballC")  
#install.xts("xts")  
#source("http://bioconductor.org/biocLite.R")  
#biocLite("Rgraphviz")  
#install.package("igraph")  
#install("qdap")

# load all the libraries required for analysis

library(openssl)  
library(xts)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(httpuv)  
library(ggplot2)   
library(twitteR)  
library(RCurl)

## Loading required package: bitops

library(tm)

## Loading required package: NLP

##   
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':  
##   
## annotate

library(wordcloud)

## Loading required package: RColorBrewer

library(SnowballC)  
library(graph)

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## The following object is masked from 'package:NLP':  
##   
## annotation

## The following object is masked from 'package:twitteR':  
##   
## as.data.frame

## The following objects are masked from 'package:stats':  
##   
## IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, basename, cbind,  
## colMeans, colnames, colSums, dirname, do.call, duplicated,  
## eval, evalq, Filter, Find, get, grep, grepl, intersect,  
## is.unsorted, lapply, lengths, Map, mapply, match, mget, order,  
## paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,  
## Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,  
## table, tapply, union, unique, unsplit, which, which.max,  
## which.min

library(Rgraphviz)

## Loading required package: grid

##   
## Attaching package: 'Rgraphviz'

## The following object is masked from 'package:twitteR':  
##   
## name

library(topicmodels)  
library(igraph)

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:graph':  
##   
## degree, edges, intersection, union

## The following objects are masked from 'package:BiocGenerics':  
##   
## normalize, path, union

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

## The following object is masked from 'package:base':  
##   
## union

library(plyr)

##   
## Attaching package: 'plyr'

## The following object is masked from 'package:graph':  
##   
## join

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

library(dplyr)

##   
## Attaching package: 'dplyr'

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

## The following objects are masked from 'package:igraph':  
##   
## as\_data\_frame, groups, union

## The following object is masked from 'package:graph':  
##   
## union

## The following objects are masked from 'package:BiocGenerics':  
##   
## combine, intersect, setdiff, union

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

## The following objects are masked from 'package:xts':  
##   
## first, last

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

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

library(stringr)

##   
## Attaching package: 'stringr'

## The following object is masked from 'package:graph':  
##   
## boundary

require(devtools)

## Loading required package: devtools

#install\_github("okugami79/sentiment140")  
library(sentiment)

## Loading required package: rjson

library(qdap)

## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

##   
## Attaching package: 'qdapRegex'

## The following object is masked from 'package:dplyr':  
##   
## explain

## The following object is masked from 'package:ggplot2':  
##   
## %+%

## Loading required package: qdapTools

##   
## Attaching package: 'qdapTools'

## The following object is masked from 'package:dplyr':  
##   
## id

## The following object is masked from 'package:plyr':  
##   
## id

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

##   
## Attaching package: 'qdap'

## The following object is masked from 'package:stringr':  
##   
## %>%

## The following object is masked from 'package:dplyr':  
##   
## %>%

## The following objects are masked from 'package:igraph':  
##   
## %>%, diversity

## The following objects are masked from 'package:BiocGenerics':  
##   
## counts, Filter, pos

## The following objects are masked from 'package:tm':  
##   
## as.DocumentTermMatrix, as.TermDocumentMatrix

## The following object is masked from 'package:NLP':  
##   
## ngrams

## The following object is masked from 'package:base':  
##   
## Filter

# Application keys

consumer\_key<-'Pnn9DsMpxfM0MSKznlA3RPgsM'  
consumer\_secret<-'AOczWZ3fjePKzIuxRiObwUvpQAYUL1n0fP2dCb1Xjkn0oUOAvU'  
access\_token<-'784558015030632448-7oWodTyR7IcRlzefr3ZTz6aB0vgQkHl'  
access\_secret<-'3hqG2EOnhrm5Xy8eWMSB17dLSvK1oM3nzDb6AnJDTKfJ9'  
setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

## [1] "Using direct authentication"

# Collecting tweets

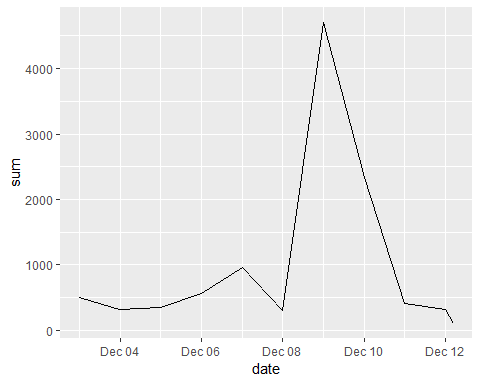
# \*\*\*\*Used first time to extract tweets\*\*\*Comment Code  
findfd= "#gunviolence"  
number= 30000  
bd\_tweets = searchTwitter(findfd, n=number, lang="en")

## Warning in doRppAPICall("search/tweets", n, params = params,  
## retryOnRateLimit = retryOnRateLimit, : 30000 tweets were requested but the  
## API can only return 10875

tweets.df <- twListToDF(bd\_tweets) # convert tweets to a data frame

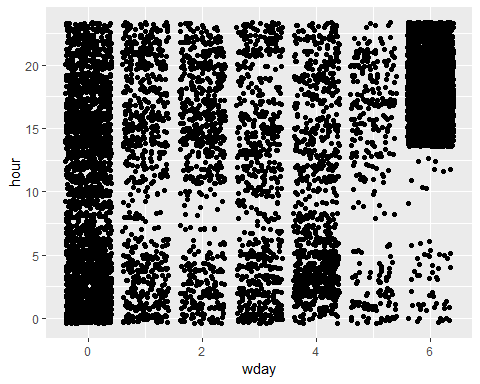
# Time Series of tweets

##The xts function creates a timeline from a vector of values and a vector of timestamps.  
#If we know how many tweets we have, we can just create a simple list or vector containing that number of 1s  
  
ts=xts(rep(1,times=nrow(tweets.df)),tweets.df$created)  
  
#We can now do some handy number crunching on the timeseries, such as applying a formula to values contained with day, week, month, quarter or year time bins.  
#So for example, if we sum the unit values in daily bin, we can get a count of the number of tweets per day  
ts.sum=apply.daily(ts,sum)   
#also apply. weekly, monthly, quarterly, yearly  
  
#If for any resason we need to turn the timeseries into a dataframe, we can:  
ts.sum.df=data.frame(date=index(ts.sum), coredata(ts.sum))  
  
colnames(ts.sum.df)=c('date','sum')  
  
#We can then use ggplot to plot the timeseries...  
ggplot(ts.sum.df)+geom\_line(aes(x=date,y=sum))



# Chart a count of the number of tweets by day, week, or hour

tweets.df <- tweets.df[ -c(2:4, 6:16) ]  
# Create a corpus  
bd\_corpus = VCorpus(VectorSource(tweets.df$text))  
#label a tweet with the month number  
tweets.df$month=sapply(tweets.df$created, function(x) {p=as.POSIXlt(x);p$mon})  
#label a tweet with the hour  
tweets.df$hour=sapply(tweets.df$created, function(x) {p=as.POSIXlt(x);p$hour})  
#label a tweet with a number corresponding to the day of the week  
tweets.df$wday=sapply(tweets.df$created, function(x) {p=as.POSIXlt(x);p$wday})  
  
ggplot(tweets.df)+geom\_jitter(aes(x=wday,y=hour))

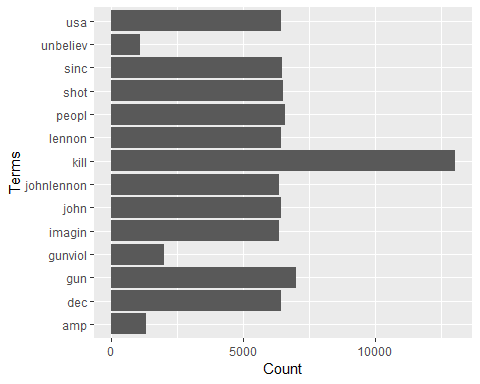


# Clean the corpus by removing punctuation, numbers, and white spaces

bd\_clean <- tm\_map(bd\_corpus, tolower)  
bd\_clean <- tm\_map(bd\_clean, removePunctuation)  
bd\_clean <- tm\_map(bd\_clean, removeNumbers)  
#This means that all the words are converted to their stem (e.g. learing->learn)  
bd\_clean <- tm\_map(bd\_clean, stemDocument)  
# add two extra stop words: "available" and "via"  
myStopwords <- c(stopwords('english'), "available", "via")  
# remove "r" and "big" from stopwords  
myStopwords <- setdiff(myStopwords, c("r", "big"))  
#Stopwords are commonly used words in the English language such as I, me etc  
bd\_clean <- tm\_map(bd\_clean, removeWords, myStopwords)  
bd\_clean <- tm\_map(bd\_clean, PlainTextDocument)  
bd\_clean <- tm\_map(bd\_clean, stripWhitespace)

# 25 most frequest words in word cloud

# 25 most frequest words in word cloud  
dtm <- DocumentTermMatrix(bd\_clean)  
tdm <- TermDocumentMatrix(bd\_clean)   
freq.terms <- findFreqTerms(tdm, lowfreq = 700)  
term.freq <- rowSums(as.matrix(tdm))  
term.freq <- subset(term.freq, term.freq >= 700)  
df <- data.frame(term = names(term.freq), freq = term.freq)  
ggplot(df, aes(x = term, y = freq)) + geom\_bar(stat = "identity") +  
 xlab("Terms") + ylab("Count") + coord\_flip()



mat\_rix <- as.matrix(dtm)  
vec\_tor <- sort(colSums(mat\_rix),decreasing=TRUE)  
words <- names(vec\_tor)  
data <- data.frame(word=words, freq=vec\_tor)  
# display 25 most frequent words  
head(data,25)

## word freq  
## kill kill 13015  
## gun gun 7009  
## peopl peopl 6563  
## shot shot 6518  
## sinc sinc 6465  
## usa usa 6427  
## dec dec 6425  
## john john 6422  
## lennon lennon 6419  
## johnlennon johnlennon 6359  
## imagin imagin 6350  
## gunviol gunviol 2014  
## amp amp 1298  
## unbeliev unbeliev 1090  
## tell tell 694  
## give give 690  
## media media 669  
## social social 666  
## senatorpark senatorpark 665  
## numbskul numbskul 662  
## password password 662  
## condorlaw condorlaw 661  
## talk talk 536  
## america america 535  
## josephsakran josephsakran 517

# Word Association

findAssocs(tdm, "kill", 0.2)

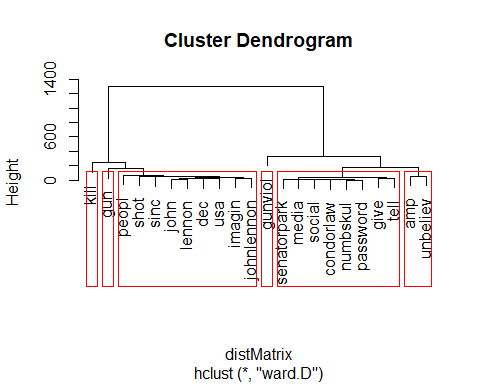
## $kill  
## dec john lennon usa shot sinc   
## 0.99 0.99 0.99 0.99 0.98 0.98   
## imagin johnlennon peopl gun   
## 0.97 0.97 0.97 0.86

findAssocs(tdm, "gun", 0.2)

## $gun  
## dec john lennon sinc usa kill   
## 0.87 0.87 0.87 0.87 0.87 0.86   
## peopl imagin johnlennon shot   
## 0.86 0.85 0.85 0.85

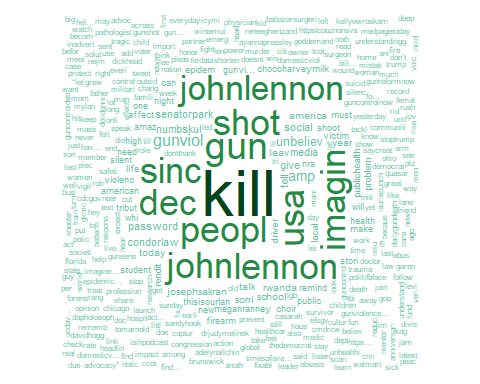
# Word Clustering

# remove sparse terms  
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)  
m2 <- as.matrix(tdm2)  
# cluster terms  
distMatrix <- dist(scale(m2))  
fit <- hclust(distMatrix, method = "ward.D")  
plot(fit)  
rect.hclust(fit, k = 6)

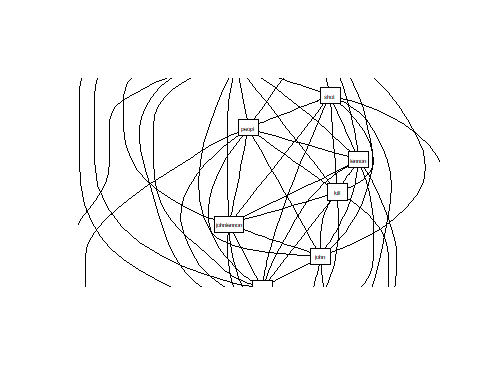


# wordcloud and network diagram

m <- as.matrix(tdm)  
# calculate the frequency of words and sort it by frequency  
word.freq <- sort(rowSums(m), decreasing = T)  
# colors  
pal <- brewer.pal(9, "BuGn")[-(1:4)]  
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 3,  
 random.order = F, colors = pal)

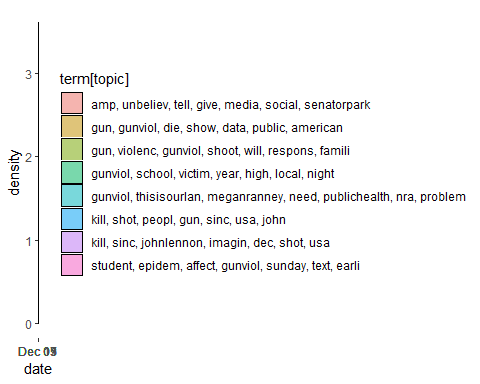


plot(tdm, term = freq.terms, corThreshold = 0.1)



# Topic Modelling

lda <- topicmodels::LDA(dtm, k = 8) # find 8 topics  
term <- terms(lda, 7) # first 7 terms of every topic  
term <- apply(term, MARGIN = 2, paste, collapse = ", ")  
topics <- topics(lda) # 1st topic identified for every document (tweet)  
topics <- data.frame(date=as.Date(tweets.df$created), topic=topics)  
ggplot(topics, aes(date, fill = term[topic])) + geom\_density(alpha = 0.5, position = "stack")



# scan the positive word file and negative word file

pos = scan('positive-word-list.txt', what='character', comment.char=';')  
neg= scan('negative-word-list.txt', what='character', comment.char=';')

# returnpscore for counting the positive matching words.

returnpscore=function(tweets) {  
 # clean up sentences with R's regex-driven global substitute, gsub():  
 sentence = gsub('[[:punct:]]', '', tweets)  
 sentence = gsub('[[:cntrl:]]', '', sentence)  
 sentence = gsub('\\d+', '', sentence)  
 # and convert to lower case:  
 sentence = tolower(sentence)  
 # split into words. str\_split is in the stringr package  
 word.list = str\_split(sentence, '\\s+')  
 # sometimes a list() is one level of hierarchy too much  
 words = unlist(word.list)  
 pos.match=match(words,pos)  
 pos.match=!is.na(pos.match)  
 pos.score=sum(pos.match)  
 return(pos.score)  
}  
#Next we apply this function to the tweetclean list  
positive.score=lapply(bd\_clean,function(x) returnpscore(x))  
#Next we define a loop to count the total number of positive words present in the tweets  
pcount=0  
for (i in 1:length(positive.score)) {  
 pcount=pcount+positive.score[[i]]  
}  
pcount

## [1] 1293

# returnnscore for counting the positive matching words.

returnnscore=function(tweets) {  
 # clean up sentences with R's regex-driven global substitute, gsub():  
 sentence = gsub('[[:punct:]]', '', tweets)  
 sentence = gsub('[[:cntrl:]]', '', sentence)  
 sentence = gsub('\\d+', '', sentence)  
 # and convert to lower case:  
 sentence = tolower(sentence)  
 # split into words. str\_split is in the stringr package  
 word.list = str\_split(sentence, '\\s+')  
 # sometimes a list() is one level of hierarchy too much  
 words = unlist(word.list)  
 neg.match=match(words,neg)  
 neg.match=!is.na(neg.match)  
 neg.score=sum(neg.match)  
 return(neg.score)  
}  
#Next we apply this function to the tweetclean list  
negative.score=lapply(bd\_clean,function(x) returnnscore(x))  
#Next we define a loop to count the total number of positive words present in the tweets  
ncount=0  
for (i in 1:length(negative.score)) {  
 ncount=ncount+negative.score[[i]]  
}  
ncount

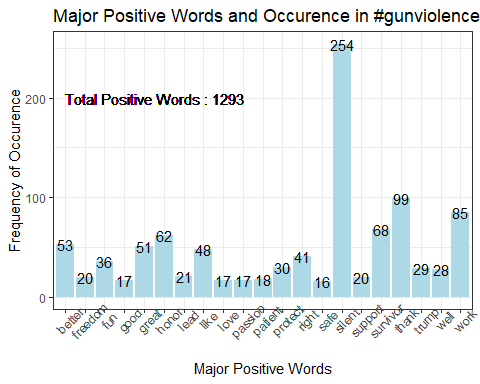
## [1] 26918

# The following code retrieves the positive matching words.

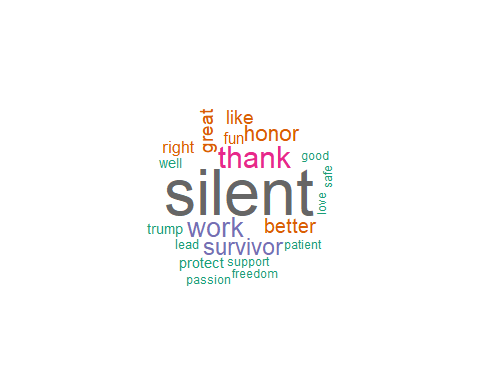
poswords=function(tweets){  
 # clean up sentences with R's regex-driven global substitute, gsub():  
 sentence = gsub('[[:punct:]]', '', tweets)  
 sentence = gsub('[[:cntrl:]]', '', sentence)  
 sentence = gsub('\\d+', '', sentence)  
 # and convert to lower case:  
 sentence = tolower(sentence)  
 # split into words. str\_split is in the stringr package  
 word.list = str\_split(sentence, '\\s+')  
 # sometimes a list() is one level of hierarchy too much  
 words = unlist(word.list)  
 pmatch=match(words,pos)  
 posw=pos[pmatch]  
 posw=posw[!is.na(posw)]  
 return(posw)  
 }

# The code below creates matches of positive words

pdatamart=data.frame()  
for (t in bd\_clean) {  
 pdatamart=c(poswords(t),pdatamart)  
}  
pwords=unlist(pdatamart)  
dpwords=data.frame(table(pwords))  
dpwords=dpwords%>%  
 mutate(pwords=as.character(pwords))%>%  
 filter(Freq>15)  
  
ggplot(dpwords,aes(pwords,Freq))+geom\_bar(stat="identity",fill="lightblue")+theme\_bw()+  
 geom\_text(aes(pwords,Freq,label=Freq),size=4)+  
 labs(x="Major Positive Words", y="Frequency of Occurence",title=paste("Major Positive Words and Occurence in",findfd,"twitter feeds, n =",number))+  
 geom\_text(aes(1,200,label=paste("Total Positive Words :",pcount)),size=4,hjust=0)+theme(axis.text.x=element\_text(angle=45))



wordcloud(words = dpwords$pwords, freq = dpwords$Freq, min.freq = 1,  
 max.words=150, random.order=FALSE, rot.per=0.35,   
 colors=brewer.pal(8, "Dark2"))

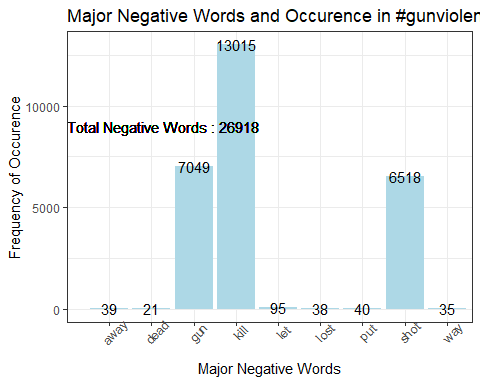


# The following code retrieves the negative matching words.

negwords=function(tweets){  
 # clean up sentences with R's regex-driven global substitute, gsub():  
 sentence = gsub('[[:punct:]]', '', tweets)  
 sentence = gsub('[[:cntrl:]]', '', sentence)  
 sentence = gsub('\\d+', '', sentence)  
 # and convert to lower case:  
 sentence = tolower(sentence)  
 # split into words. str\_split is in the stringr package  
 word.list = str\_split(sentence, '\\s+')  
 # sometimes a list() is one level of hierarchy too much  
 words = unlist(word.list)  
 nmatch=match(words,neg)  
 nosw=neg[nmatch]  
 nosw=nosw[!is.na(nosw)]  
 return(nosw)  
 }

# The code below creates matches of negative words

ndatamart=data.frame()  
for (t in bd\_clean) {  
 ndatamart=c(negwords(t),ndatamart)  
}  
nwords=unlist(ndatamart)  
dnwords=data.frame(table(nwords))  
dnwords=dnwords%>%  
 mutate(nwords=as.character(nwords))%>%  
 filter(Freq>20)  
  
ggplot(dnwords,aes(nwords,Freq))+geom\_bar(stat="identity",fill="lightblue")+theme\_bw()+  
 geom\_text(aes(nwords,Freq,label=Freq),size=4)+  
 labs(x="Major Negative Words", y="Frequency of Occurence",title=paste("Major Negative Words and Occurence in",findfd,"twitter feeds, n =",number))+  
 geom\_text(aes(0,9000,label=paste("Total Negative Words :",ncount)),size=4,hjust=0)+theme(axis.text.x=element\_text(angle=45))



wordcloud(words = dnwords$nwords, freq = dnwords$Freq, min.freq = 1,  
 max.words=100, random.order=FALSE, rot.per=0.35,   
 colors=brewer.pal(8, "Dark2"))

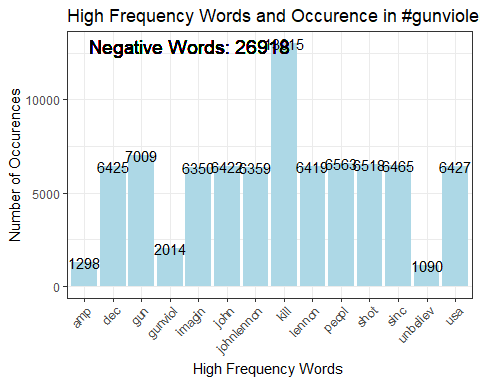


# finally, we convert the matrix to a data frame, filter for Minimum frequency > 75and plot using ggplot2

# #removing sparse terms  
dtms=removeSparseTerms(dtm,.99)  
freq=sort(colSums(as.matrix(dtm)),decreasing=TRUE)  
#get some more frequent terms  
findFreqTerms(dtm,lowfreq=100)

## [1] "affect" "amaz" "america"   
## [4] "american" "amp" "can"   
## [7] "chocoharveymilk" "choir" "condorlaw"   
## [10] "cut" "dec" "doesnt"   
## [13] "driver" "epidem" "expect"   
## [16] "firearm" "give" "gun"   
## [19] "guncontrol" "gunvi…" "gunviol"   
## [22] "health" "high" "imagin"   
## [25] "john" "johnlennon" "josephsakran"   
## [28] "kill" "leav" "lennon"   
## [31] "life" "local" "make"   
## [34] "mani" "media" "meganranney"   
## [37] "must" "need" "new"   
## [40] "night" "nra" "numbskul"   
## [43] "one" "password" "peopl"   
## [46] "prevent" "problem" "public"   
## [49] "publichealth" "remind" "rendit"   
## [52] "rwanda" "school" "senatorpark"   
## [55] "shoot" "shorten" "shot"   
## [58] "silent" "sinc" "social"   
## [61] "sorri" "speak" "stori"   
## [64] "student" "talk" "tell"   
## [67] "thisisourlan" "today" "tribut"   
## [70] "unbeliev" "usa" "victim"   
## [73] "violenc" "week" "whi"   
## [76] "will" "year"

wf=data.frame(word=names(freq),freq=freq)  
wfh=wf%>%  
 filter(freq>=700,!word==tolower(findfd))  
ggplot(wfh,aes(word,freq))+geom\_bar(stat="identity",fill='lightblue')+theme\_bw()+  
 theme(axis.text.x=element\_text(angle=45,hjust=1))+  
 geom\_text(aes(word,freq,label=freq),size=4)+labs(x="High Frequency Words ",y="Number of Occurences", title=paste("High Frequency Words and Occurence in",findfd," twitter feeds, n =",number))+  
 geom\_text(aes(1,max(freq)-100,label=paste("Positive Words:",pcount,"\n","Negative Words:",ncount,"\n")),size=5, hjust=0)



# sentiment score function

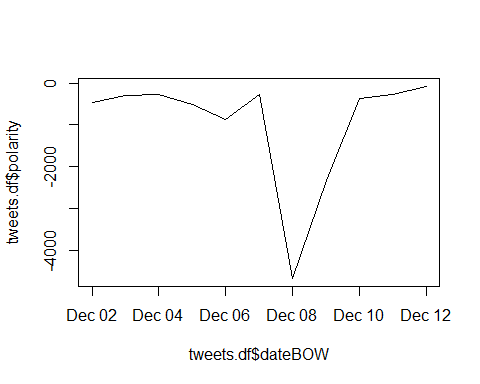
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')  
{  
# we got a vector of sentences. plyr will handle a list # or a vector as an "l" for us # we want a simple array ("a") of scores back, so we use # "l" + "a" + "ply" = "laply":  
 scores = laply(sentences, function(sentence, pos.words, neg.words) {  
 # clean up sentences with R's regex-driven global substitute, gsub():  
 sentence = gsub('[[:punct:]]', '', sentence)  
 sentence = gsub('[[:cntrl:]]', '', sentence)  
 sentence = gsub('\\d+', '', sentence)  
 # and convert to lower case:  
 sentence = tolower(sentence)  
 # split into words. str\_split is in the stringr package  
 word.list = str\_split(sentence, '\\s+')  
 # sometimes a list() is one level of hierarchy too much  
 words = unlist(word.list)  
 # compare our words to the dictionaries of positive & negative terms  
 pos.matches = match(words, pos.words)  
 neg.matches = match(words, neg.words)  
 # match() returns the position of the matched term or NA # we just want a TRUE/FALSE:  
 pos.matches = !is.na(pos.matches)  
 neg.matches = !is.na(neg.matches)  
 # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():  
 score = sum(pos.matches) - sum(neg.matches)  
 return(score)  
 }, pos.words, neg.words, .progress=.progress )  
 scores.df = data.frame(score=scores, text=sentences)  
return(scores.df)  
   
}

# run the sentiment score function on the clean\_text column and export the file

scoreBOW = score.sentiment(tweets.df$text, pos, neg)  
tweets.df$scoreBOW = scoreBOW$score

# create a new column sentiment, score >0 as positive, score=0 as neutral and score<0 as neutral

tweets.df$sentimentBOW= ifelse(tweets.df$score>0, "Positive",ifelse(tweets.df$score<0,"Negative","Neutral"))  
tweets.df$polarity <- 0  
tweets.df$polarity[tweets.df$sentimentBOW == "Positive"] <- 1  
tweets.df$polarity[tweets.df$sentimentBOW == "Negative"] <- -1  
tweets.df$dateBOW <- as.Date(tweets.df$created)  
result <- aggregate(tweets.df$polarity ~ tweets.df$dateBOW, data = tweets.df, sum)  
plot(result, type = "l")



poswords.sentiment <- nrow(subset(tweets.df, tweets.df$sentimentBOW=="Positive", select=c(sentimentBOW)))  
negwords.sentiment <- nrow(subset(tweets.df, tweets.df$sentimentBOW=="Negative", select=c(sentimentBOW)))  
neuwords.sentiment <- nrow(subset(tweets.df, tweets.df$sentimentBOW=="Neutral", select=c(sentimentBOW)))  
  
write.csv(tweets.df, file = "tweets\_gun\_violence.csv",row.names=FALSE)