Wordcloud of Top 100 Words in tweets using the keyword "pulse"

Using 5000 tweets with the hashtag #pulse from June 16, 2016, we will create a wordcloud of the top words used to describe the tragedy.

Global parameters

Set working directory by pointing to the location on your computer where you have stored the files. Below, we have chosen to Save the folder "RAnalysis" on the Desktop on a Mac. It contains all the other R scripts, texts, notebooks, and results. If you have branched the github, simply note where you have save the folder. If you are on a PC, you will need to use an absolute path such as "C:Users:XXX." ***

```
setwd("~/Desktop/R/Text_Analysis/data/twitter/")
```

Include necessary packages for notebook ***

```
library(knitr)
library(markdown)
library(rmarkdown)
library(wordcloud)
```

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

```
library(qdap)
```

```
## Loading required package: qdapDictionaries
## Loading required package: qdapRegex
## Loading required package: qdapTools
##
## Attaching package: 'qdap'
##
## The following object is masked from 'package:base':
##
##     Filter
```

```
library(RColorBrewer)
```

Load data (this time a curated set of tweets grabbed using the twitterR library and API authentication, then saved with the .RData extension) ***

```
load(file = "~/Desktop/R/Text_Analysis/data/twitter/pulse2016-06-16.RData")
```

Prepare text data While it seems logical to use tm (text mining package) again to create our wordcloud, there is a bug in the library that causes it to fail with this kind of data when run on a mac. Instead, we will take the opportunity to process the data in a different way.

First we are going to take the list of 5000 tweets created when we grabbed the file above and convert it into a character vector, grabbing only the text. The "sapply" function to traverse the list "tweets" applying a function "x" which grabs the text. ***

```
tweetlist <- sapply(tweets, function(x) x$text)
```

Let's look at the first few tweets:

```
head(tweetlist)
```

```
## [1] "RT @sunlightyoga: Yoga benefits athletes @HighDesertYoga1  https://t.co/77XpVO2v5C"
## [2] "Life has become cognitively more demanding, with increasing use of communication and informatio
## [3] "Poll where only 59% of Trump supporters rule out Obama involvement in Pulse shooting. Also favo
## [4] "Pulse nightclub owner speaks out for the first time: 'The club will reopen' - Gay Star News htt
## [5] "RT @hormonegoddess: Wednesday Hormone Wisdom... https://t.co/1nwWSxrMqb"
## [6] "\"How To Avoid A #Divorce\" https://t.co/qev9X7W90I by @RealLoveCompany on @LinkedIn #RealLove
```

Next we want to strip the URLs from our tweets. Using "gsub" to replace URLS in our list with any of the characters below with nothing (""), our tweetlist is now free of them. Notice that the "|" is used to created a concatenated "http..." or "https..." with wild cards (*) for text after the hypertext transfer protocol.* **

```
tweetlist=gsub("(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", "", tweetlist)
```

Now we will go on to do the same with mentions, usernames. There may be times when you want to keep these (for example #blacklivesmatter is driven by some central tweeters in the community, including including DeRay Mckesson (@deray), Johnetta Elzie (@nettaaaaaaaa), Shaun King (@shaunking), Daniel Jose Older (@djolder), Bassem Masri (@bassem_masri), and others. ***

```
tweetlist=gsub("@(.*)", "", tweetlist)
```

Now we will go on to do the same with anything that is NOT alphanumeric such as punctuation, and, more importantly for tweets, emojis. ***

```
tweetlist=gsub( "[^[:alnum:] ]", "", tweetlist )
```

Let's look at the cleaned tweelist.

```
head(tweetlist)
```

```
## [1] "RT "
## [2] "Life has become cognitively more demanding with increasing use of communication and information
## [3] "Poll where only 59 of Trump supporters rule out Obama involvement in Pulse shooting Also favor
## [4] "Pulse nightclub owner speaks out for the first time The club will reopen  Gay Star News "
## [5] "RT "
## [6] "How To Avoid A Divorce "
```

Now it's time to split (*strsplit*) the big string into separate words (\W+), an argument from perl.

```
words <-strsplit(tweetlist, "\\W+", perl=TRUE)
head(words)
```

```
## [[1]]
## [1] "RT"
##
## [[2]]
##  [1] "Life"          "has"           "become"        "cognitively"
##  [5] "more"          "demanding"     "with"          "increasing"
##  [9] "use"           "of"            "communication" "and"
## [13] "information"   "technology"    "The"           "vol"
##
## [[3]]
##  [1] "Poll"        "where"       "only"        "59"          "of"
##  [6] "Trump"       "supporters"  "rule"        "out"         "Obama"
## [11] "involvement" "in"          "Pulse"       "shooting"    "Also"
## [16] "favor"       "RE"          "Lee"         "over"        "MLK"
##
## [[4]]
##  [1] "Pulse"     "nightclub" "owner"     "speaks"    "out"
##  [6] "for"       "the"       "first"     "time"      "The"
## [11] "club"      "will"      "reopen"    "Gay"       "Star"
## [16] "News"
##
## [[5]]
## [1] "RT"
##
## [[6]]
## [1] "How"     "To"      "Avoid"   "A"       "Divorce"
```

It's time to remove stopwords. In our plain text wordcloud, we used tm's options for a wordlist; here we are using qdap's word list and specifying that we wish to use the top 100 words from E.B. Fry's top 1000 words. Note that if we were to change the argument to "Top1000Words," we'd eliminate about 90% of our text! Concatenated onto the list are common twitter words not yet eliminated such at "rt" (retweet), "amp" (&), and our search term "pulse." ***

```
words=rm_stopwords(words,c(Top100Words,"rt", "amp", "pulse"))
```

Having removed unwanted items, it's now time to ditch the empty elements."lapply" returns a list of the same length as tweelist, each element of which is the result of applying the argument "length>0" to each element. ***

```
words=words[lapply(words,length)>0]
head(words)
```

```
## [[1]]
## [1] "life"          "become"        "cognitively"   "demanding"
## [5] "increasing"    "communication" "information"   "technology"
## [9] "vol"
##
## [[2]]
##  [1] "poll"        "where"       "only"        "59"          "trump"
##  [6] "supporters"  "rule"        "obama"       "involvement" "shooting"
## [11] "also"        "favor"       "re"          "lee"         "over"
## [16] "mlk"
##
## [[3]]
## [1] "nightclub" "owner"     "speaks"    "club"      "reopen"    "gay"
## [7] "star"      "news"
##
## [[4]]
## [1] "avoid"   "divorce"
##
## [[5]]
## [1] "development" "online"      "reservation" "application" "restaurants"
##
## [[6]]
## [1] "nightclub"   "attack"      "brings"      "invisible"   "demographic"
## [6] "worldwide"   "view"
```

Currently, we have a list of items (our original tweets) that contain a list of the words used in that tweet. For our wordcloud, we would simply like a list of words. ***

```
words=unlist(words,recursive = FALSE)
head(words)
```

```
## [1] "life"          "become"        "cognitively"   "demanding"
## [5] "increasing"    "communication"
```

Next we'd like to create a table of these words in descending order of use. For plotting purposes, we also need to create a vector (a sequence of data elements of the same type) of frequencies and keep just the words from the tweets in that list. ***

```
words=sort(table(words),decreasing=T)
freqs=as.vector(words)
words=names(words)
```

Plot the wordcloud! There are a number of arguments you can customize: "random.order" is false so that words are plotted in order of decreasing frequency; "scale"" indicates the size of the words; "rot.per" lets you customize the proportion of words that are rotated 90 degrees; "max.words" controls how many words show up in the wordcloud; and we've used the library "RColorBrewer" to give us access to some predefined palettes. Note that if you change the palette, you need to tell the argument how many colors are in the new palette. ***

```
plot = wordcloud(words,freqs,scale=c(3,.7),max.words=75, rot.per=0,
          colors=brewer.pal(8, "Dark2"))
```