OK Cupid basic word clouds

Introduction

Text-mining requires considerable preparation of a list of words, eliminating common words (like 'and' or 'the'), and then compiling frequencies of common words and sequences of words (like "Big Data")

In this illustration, we take the text self-descriptive essays submitted to *OK Cupid* and create a graphic commonly known as a *Word Cloud*.

The raw text for this assignment is contained in a dataset called `profiles` within the R package `okcupiddata`. You will need to install this package, along with the package `tidytext`.

Note that the DataCamp course also uses `qdap` as an alternative to `tm`, and makes good use of these additional packages:

- `plotrix`
- `dendextend`
- `RWeka`
- `lsa`

This demo lays out several common and fundamental operations in a text-mining project. The code uses the following packages, which are all invoked before the code that appears below. Check your RStudio list of *Packages* and install any that are not present.

- okcupiddata – contains cleaned profile data for the case study
- tm – provides functionality for text mining
- SnowballC – to "stem" terms (group related roots)
- wordcloud – to produce word cloud graphics
- RColorBrewer – to enhance graphs with color

```
library(okcupiddata)
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
```

Create and Clean the Corpus

The initial code chunk reads in the body of data from the `profiles` object. Most of the columns are responses to closed-ended multiple-choice questions where the phrasing is supplied by OK Cupid. The column we want is called `essay0`. Then the data is converted to a

Corpus – a collection of text documents. Because this is such a huge body of data, we'll randomly sample 10% or about 6000 responses for initial analysis.

```
str(profiles)
```

```
## 'data.frame':    59946 obs. of  22 variables:
##  $ age        : int  22 35 38 23 29 29 32 31 24 37 ...
##  $ body_type  : chr  "a little extra" "average" "thin" "thin" ...
##  $ diet       : chr  "strictly anything" "mostly other" "anything"
"vegetarian" ...
##  $ drinks     : chr  "socially" "often" "socially" "socially" ...
##  $ drugs      : chr  "never" "sometimes" NA NA ...
##  $ education  : chr  "working on college/university" "working on space
camp" "graduated from masters program" "working on college/university" ...
##  $ ethnicity  : chr  "asian, white" "white" NA "white" ...
##  $ height     : int  75 70 68 71 66 67 65 65 67 65 ...
##  $ income     : int  NA 80000 NA 20000 NA NA NA NA NA NA ...
##  $ job        : chr  "transportation" "hospitality / travel" NA "student"
...
##  $ last_online: POSIXct, format: "2012-06-28 20:30:00" "2012-06-29
21:41:00" ...
##  $ location   : chr  "south san francisco, california" "oakland,
california" "san francisco, california" "berkeley, california" ...
##  $ offspring  : chr  "doesn't have kids, but might want them" "doesn't
have kids, but might want them" NA "doesn't want kids" ...
##  $ orientation: chr  "straight" "straight" "straight" "straight" ...
##  $ pets       : chr  "likes dogs and likes cats" "likes dogs and likes
cats" "has cats" "likes cats" ...
##  $ religion   : chr  "agnosticism and very serious about it" "agnosticism
but not too serious about it" NA NA ...
##  $ sex        : chr  "m" "m" "m" "m" ...
##  $ sign       : chr  "gemini" "cancer" "pisces but it doesn't matter"
"pisces" ...
##  $ smokes     : chr  "sometimes" "no" "no" "no" ...
##  $ speaks     : chr  "english" "english (fluently), spanish (poorly),
french (poorly)" "english, french, c++" "english, german (poorly)" ...
##  $ status     : chr  "single" "single" "available" "single" ...
##  $ essay0     : chr  "about me:    i would love to think that i was some
some kind of intellectual: either the dumbest smart guy, or "| __truncated__
"i am a chef: this is what that means.  1. i am a workaholic.  2. i love to
cook regardless of whether i am at w"| __truncated__ "i'm not ashamed of
much, but writing public text on an online dating site makes me pleasantly
uncomfortable. i'"| __truncated__ "i work in a library and go to school. . ."
...
```

```
set.seed(1138) # for reproducible results
n <- nrow(profiles)

allessays <- data.frame(profiles$essay0)
test_idx <- sample.int(n, size= round(0.1 * n))  # sample 10% of rows
```

```
text <- data.frame(allessays[test_idx,])
corpus <- Corpus(VectorSource(text[,1]))
```

Next comes the phase of cleaning up the corpus. The following code chunk produces no output directly, but converts all text to lowercase, removes punctuation, and removes numerals.

```
#Clean-up
corpus <- tm_map(corpus, tolower)  #make all text lower case
corpus <- tm_map(corpus, removePunctuation) #remove all punctuation
corpus <- tm_map(corpus, removeNumbers) # remove numbers
```

In any text, we often find plurals and other word forms that we'd prefer to treat as one word. "Stemming" refers to truncating words so that related terms are counted together. This is where package `SnowballC` does its work.

```
#stemming to treat related terms alike
####  NOTE FOR 2018: SHOULD STEM BEFORE STOP WORDS
cleanset <- tm_map(corpus, stripWhitespace) # purge extra white space
cleanset <- tm_map(cleanset, PlainTextDocument)
cleanset <- tm_map(cleanset,stemDocument)
```

The next recommended step is to apply *stopwords* – specifying terms that should be considered non-informative in the mining stages to follow. These may be common English words (articles, pronounds, etc.) or words that appear often in the corpus but have special status in this particular corpus.

To see the default set of English stopwords, in the console type `stopwords("en")`. We can add or remove stopwords, depending on the nature of the data. For example, all of these OK Cupid clients live within 25 miles of San Francisco, so we may want to add those terms to the list of stopwords. Similarly, because this is a dating site where individuals meet other people, we might want the word "other" to be retained in the analysis, but note that "other" is a standard stop word.

```
# apply standard English stopwords, and add "san" and "francisco"
special <- c("san","francisco")
myStopwords <- c(stopwords('english'),special)
# remove "other" from English stop words

myStopwords <- setdiff(myStopwords, c("other"))
# now remove stopwords from the corpus
cleanset <- tm_map(corpus, removeWords, myStopwords)
```

After stemming and removing stop words, we create a *term document matrix*, omitting very short words. Lastly, we list frequent words – we may want to revise our stopwords list to remove repeated but unremarkable words.

```
#Build term document matrix
cleanset <- tm_map(cleanset, PlainTextDocument)
cleanset <- Corpus(VectorSource(cleanset))
```

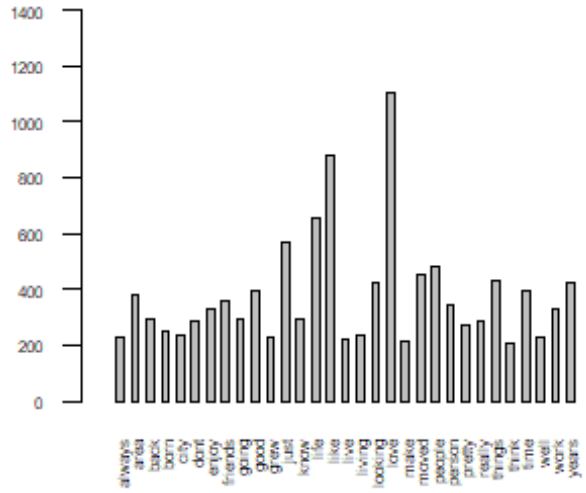
```
tdm <- TermDocumentMatrix(cleanset, control=list(wordLengths = c(4, Inf)))
#overlook 2 letter words

# inspect frequent words
findFreqTerms(tdm, lowfreq=100)
```

```
##  [1] "also"       "always"     "anything"   "area"       "around"
##  [6] "back"       "best"       "born"       "california" "city"
## [11] "coast"      "college"    "currently"  "dont"       "east"
## [16] "easy"       "enjoy"      "family"     "feel"       "find"
## [21] "first"      "friends"    "funny"      "girl"       "going"
## [26] "good"       "great"      "grew"       "happy"      "hard"
## [31] "just"       "kind"       "know"       "laid"       "last"
## [36] "laugh"      "life"       "like"       "little"     "live"
## [41] "lived"      "living"     "looking"    "love"       "loving"
## [46] "make"       "many"       "meet"       "moved"      "much"
## [51] "music"      "name"       "native"     "never"      "nice"
## [56] "open"       "originally" "other"      "outgoing"   "passionate"
## [61] "people"     "person"     "pretty"     "profile"    "raised"
## [66] "really"     "recently"   "right"      "school"     "sense"
## [71] "someone"    "something"  "sometimes"  "still"      "take"
## [76] "things"     "think"      "time"       "travel"     "trying"
## [81] "want"       "well"       "will"       "work"       "working"
## [86] "world"      "year"       "years"
```

```
# NOTE: At this point, might choose to alter stop words list
```

A simple list is helpful, but gives little insight into how often each word occurs. Let's make a graph:

```
#Bar plot
termFrequency <- rowSums(as.matrix(tdm))
termFrequency <- subset(termFrequency, termFrequency>=200)

barplot(termFrequency, width=2, space = 1, xlim=c(0,180), ylim=c(0,1500),
cex.names = .5,
    cex.axis = .5,  las=2)
```




```
# las makes axis labels perpendicular, cex.names controls size of font
```

Create Word Clouds

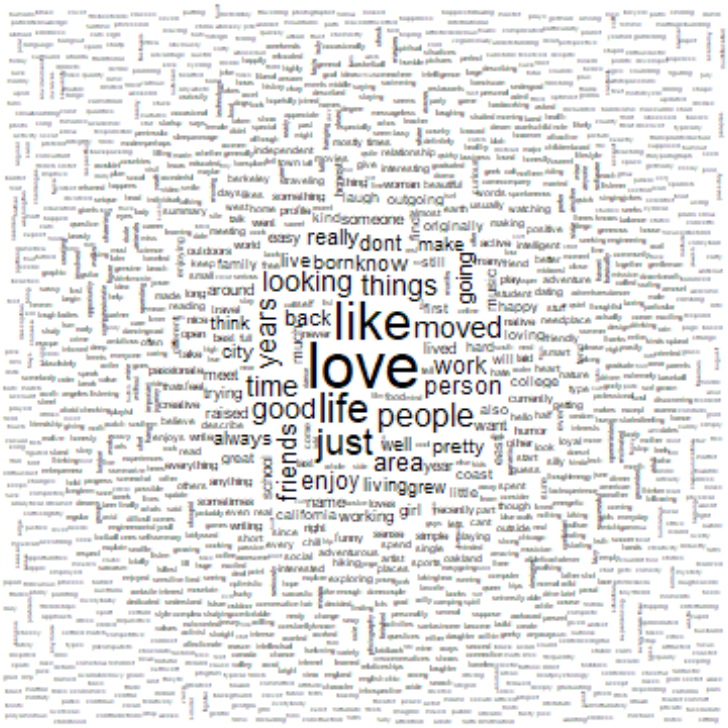
Again, we might want to go back and alter the stopwords list. Once we are satisfied with the list, we're ready to make a word cloud, using package `wordcloud`.

```
m <- as.matrix(tdm)
wordFreq <- sort(rowSums(m), decreasing=TRUE)

grayLevels <- gray((wordFreq+10)/(max(wordFreq)+10)) # sets number of gray
shades to use

# the command wordcloud is the main function:
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=50, random.order=F,
colors=grayLevels)
```

The word cloud above includes any word that occurs 50 times or more (`min.freq=50`); this creates an overcrowded curve that "spills over" the allocated space.

The next few commands show different ways to modify the size and appearance of a word cloud:

```
#Use same number of words, but re-scale the size
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3,
     scale = c(2, 0.2), random.order=F, colors=grayLevels)
```




```
# increase the minimum frequency to 100 occurrences, further re-scale the
size.
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=100,
     scale = c(3, 0.2), random.order=F, colors=grayLevels)
```

The package `RColorBrewer` adds a wide variety of color palettes. You can experiment with this endlessly!

```
# Add Color

wordcloud(words=names(wordFreq), freq=wordFreq, max.words=100, scale = c(3,
0.2),random.order=F, colors=brewer.pal(6, "Dark2"))
```


```
wordcloud(words=names(wordFreq), freq=wordFreq, max.words=100, scale = c(3,
0.2), random.order=F, colors=brewer.pal(9,"Reds"))
```