Text Analysis in R - Part 1

Robert Ackland & Timothy Graham

13 July 2015

Ensure that the required packages are loaded. There are two steps. If the packages are not already installed, we can run the following commands to install them (note: may take several minutes to complete):

```
install.packages("tm")
install.packages("wordcloud")
install.packages("RColorBrewer")
install.packages("SnowballC")
install.packages("ape")
install.packages("dendextend")
install.packages("stringr")
install.packages("topicmodels")
install.packages("slam")
install.packages("Rmpfr")
```

The next step is to 'load' the installed packages into our R session, to enable their use:

```
require("tm")
require("wordcloud")
require("RColorBrewer")
require("SnowballC")
require("ape")
require("dendextend")
require("stringr")
require("topicmodels")
require("slam")
require("Rmpfr")
require("lattice")
```

Loading in our data set

The first step is to load the dataset for this session. We will be using data collected from 'The Banana Girl' Facebook page, which is a popular Facebook page relating to the 'healthy eating' movement:

```
fbData <- read.csv("data/thebananagirl_facebook_data.csv")
```

We can view information about our newly loaded *data frame* object (which was generated using the SocialMedia package):

```
str(fbData)
```

```
## 'data.frame':    55316 obs. of  9 variables:
##  $ X            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ from         : Factor w/ 22470 levels "'Fran Holts Petersen",..: 20814 13914 19060 5782 13781 127
##  $ to           : Factor w/ 101 levels "122654954464834_934899013240420",..: 101 101 101 101 101 101
```

```
##  $ relationship : Factor w/ 2 levels "User-Post Based on Comments",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ edgeWeight   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ postType     : Factor w/ 5 levels "event","link",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ postLink     : Factor w/ 76 levels "http://www.buzzfeed.com/jamestreacher/meet-freelee-the-super-
##  $ postTimestamp: Factor w/ 101 levels "2015-05-02T11:16:48+0000",..: 101 101 101 101 101 101 101 10
##  $ commentText  : Factor w/ 7561 levels "","_???? Tell me much much more",..: NA NA NA NA NA NA NA NA
```

We just want to keep the character vector of 'comments' data, for our purposes in this session:

```
fbData <- fbData$commentText
```

Cleaning the data

We only want comments with text (many rows represent simply 'likes', as it is a bimodal Facebook network). First we remove comments with no text ('empty' elements in our fbData vector):

```
toRemove <- which(fbData=="")
fbData <- fbData[-toRemove] # remove 'offending' rows (if there are no matches, do not run this command
```

Then we remove rows with NA values. Note that the NA values in this data set represent 'likes' (i.e. the user has 'liked' a post on the Facebook page, rather than commented, so the 'comment' has been given an NA value because it does not apply in this context):

```
toRemove <- which(is.na(fbData))
fbData <- fbData[-toRemove]
```

How many comments do we have left now?

```
length(fbData)
```

```
## [1] 7695
```

We convert the character encoding to UTF-8. This avoids errors relating to 'odd' characters in the text. This is usually a good idea, but there may be situations when it is not useful, or even detrimental. Note: Mac users may encounter errors/bugs relating to character encoding, and a workaround is to convert to 'utf-8-mac':

```
fbData <- iconv(fbData, to = 'utf-8')
# Mac users should use this instead:
fbData <- iconv(fbData,to="utf-8-mac")
```

Working with text using the *tm* package

We convert our character vector `fbData` to a Vcorpus object:

```
fbCorpus <- VCorpus(VectorSource(fbData))
```

Individual comments (a.k.a. 'documents') can be accessed via the double brackets notation or the 'dollar sign' notation for accessing list elements. Let's look at comment #4.

```
fbCorpus[[4]][[1]]
```

```
## [1] "Is it true that you eat 51 bananas a day?"
```

```
fbCorpus[[4]]$content
```

```
## [1] "Is it true that you eat 51 bananas a day?"
```

We can perform a number of highly useful transformations of text using tm_map function (i.e. 'mapping to the corpus'). Note: not all of these transformations are useful in every scenario! They should be used only when it makes sense, or as required, etc.

Converting all the text to lowercase:

```
fbCorpus <- tm_map(fbCorpus, content_transformer(tolower))
```

Remove numbers from the text:

```
fbCorpus <- tm_map(fbCorpus, removeNumbers)
```

Remove punctuation from the text:

```
fbCorpus <- tm_map(fbCorpus, removePunctuation)
```

Perform 'word stemming' on the text. Note: this transformation can be highly useful, but also highly detrimental!

```
fbCorpus <- tm_map(fbCorpus, stemDocument)
```

We can also remove English 'stop words' from the text. These are common words (e.g. 'the', 'and', 'or') that we may want to exclude from our analysis. Once again, this is highly useful but also needs to be carefully applied.

```
fbCorpus <- tm_map(fbCorpus, removeWords, stopwords("english"))
```

Eliminate unnecessary 'white space' from the text. For example, "hello everyone my name is fred" becomes "hello everyone my name is fred":

```
fbCorpus <- tm_map(fbCorpus, stripWhitespace)
```

We can observe the difference now by examining comment #4 again:

```
fbCorpus[[4]]
```

```
## <<PlainTextDocument (metadata: 7)>>
##  true eat banana day
```

We could also define our own stop words and transform the text using these:

```
myStopwords <- c("banana")
fbCorpus <- tm_map(fbCorpus, removeWords, myStopwords)
```

Now we can see that comment #4 is missing 'banana'...

```
fbCorpus[[4]]
```

```
## <<PlainTextDocument (metadata: 7)>>
##  true eat  day
```

Creating a document-term matrix (DTM)

Next we create a document-term matrix (DTM) from the fbCorpus object. DTMs are a very important concept for text analysis and are highly useful. DTMs can be thought about as a table (i.e. matrix) where the rows are 'documents' (i.e. Facebook comments in our dataset), and the columns are 'terms' (i.e. each unique word found across all the documents in the dataset). The 'cells' (i.e. elements) of the matrix indicate how many times term n occurred in document m.

Note: we use the `control` argument to specify that we only want to retain words that are minimum character length of 3, up to any number of characters (Inf).

```
dtm <- DocumentTermMatrix(fbCorpus,control = list(wordLengths=c(3, Inf)))
dtm
```

```
## <<DocumentTermMatrix (documents: 7695, terms: 9910)>>
## Non-/sparse entries: 85203/76172247
## Sparsity           : 100%
## Maximal term length: 168
## Weighting          : term frequency (tf)
```

What we have is a sparse matrix, i.e. most of the elements of the matrix are 0, i.e. in our dataset most Facebook comments contain only a small percentage of 'vocabulary' of terms observed across the entire set of comments. What we want to do is remove terms that occur very infrequently, which will leave us with the most 'important' terms. We remove sparse terms using the `removeSparseTerms` function, which removes terms that occur equal to or less than a percentage threshold. For example, if we set it to 0.99, then all terms that are at least 99% sparse are removed.

```
removeSparseTerms(dtm, 0.99)
```

```
## <<DocumentTermMatrix (documents: 7695, terms: 211)>>
## Non-/sparse entries: 42015/1581630
## Sparsity           : 97%
## Maximal term length: 10
## Weighting          : term frequency (tf)
```

0.99 will do the trick for us in this session. Our DTM is extremely sparse so we will use an extreme threshold for removal.

```
dtmSparseRemoved <- removeSparseTerms(dtm, 0.99)
```

Descriptive text analysis

We can examine term frequencies in our data:

```
freqTerms <- colSums(as.matrix(dtmSparseRemoved))
freqTerms
```

```
##     actual       agre       also      alway       amaz        ani
##        158        232        219        157        117        222
##       anim      anoth      anyon      anyth     around     awesom
##       1225         95        137        101        104         99
##       back        bad     beauti     becaus      becom      befor
##        134        133         88        542        123        124
##     believ       best     better        big       bodi       call
##        170        103        206         81        158        104
##        can       cant       carb       care       caus      chang
##        586        273        105        154        135        279
##      choic       come    comment      daili        day     deserv
##        170        202        148        107        320        178
##      didnt        die       diet     differ    disgust        doe
##        113        189        188        164        120        160
##     doesnt       done       dont        eat      eater       educ
##        175         88        903       1305        136        139
##        els        end     enough        etc       even       ever
##         96         90        101        101        320        146
##      everi    everyon    everyth       fact        fat       feel
##        186        179         93        104        121        359
##       find      first     follow       food       free     freele
##         81         91        131        339        106       1046
##     friend      fruit       fuck        fur        get       girl
##         89        226        355        234        628        325
##       give        god       good        got      great        guy
##        196        117        409        124        215        116
##     happen      happi       hard       hate     health    healthi
##         89         92         92        169        101        111
##       help       hope      human      idiot      ignor     inspir
##        223        133        440         82        169        145
##       isnt        ive       just       keep       kill       kind
##         94        200        942        307        304         85
##       know      least       less        let       life   lifestyl
##        437         80         82        151        342        132
##       like      littl       live        lol       long       look
##        982        101        496        250        125        368
##        lot       love       made       make       mani       mayb
##        200        742        152        618        283        112
##       mean       meat     messag       mind      minut      money
##        155        861        189        102         85         93
##       much      natur       need      never        new       nice
##        380        152        491        236         84        101
##       noth        now        one       onli    opinion       part
##        102        290        516        286         99         88
##      peopl     person     pictur     planet      plant      pleas
##       1566        241         83        205        174        197
##      point       post    problem    product        put        raw
```

```
##        146        193        106        101        159        121
##       read       real     realli     reason    respect      right
##        102        140        318        124        132        404
##        sad       said        say        see    serious      share
##        100        128        490        371         82         94
##       shes       shit       show       sick       sinc     someon
##        115        148        127         93         89        166
##     someth      sorri      speak     spread      start      still
##        174         87        112        110        176        228
##       stop     stupid    support       sure       take       talk
##        252        151        128        109        233        132
##       tell      thank      thing      think     though    thought
##        182        335        342        600        111        143
##       time      total        tri       true      truth       turn
##        335         89        244        119        162         87
## understand        use      vegan vegetarian       veri      video
##        162        248       1349         89        189        593
##       view       want      watch        way       wear       well
##         86        492        255        485        140        254
##        whi       will       wish    without     wonder       wont
##        331        551         85         83         82         82
##       word       work      world        wow      wrong       year
##         98        214        287        119        146        212
##        yes
##        166
```

We order the term frequencies and look at the 5 *most* frequent terms and then 5 *least* frequent terms:

```
orderTerms <- order(freqTerms,decreasing=TRUE)
freqTerms[head(orderTerms)]
```

```
##  peopl  vegan    eat   anim freele   like
##   1566   1349   1305   1225   1046    982
```

```
freqTerms[tail(orderTerms)]
```

```
## serious  wonder    wont     big    find   least
##      82      82      82      81      81      80
```

Which terms occurred at least 300 times?

```
findFreqTerms(dtmSparseRemoved, 300)
```

```
##  [1] "anim"   "becaus" "can"    "day"    "dont"   "eat"    "even"
##  [8] "feel"   "food"   "freele" "fuck"   "get"    "girl"   "good"
## [15] "human"  "just"   "keep"   "kill"   "know"   "life"   "like"
## [22] "live"   "look"   "love"   "make"   "meat"   "much"   "need"
## [29] "one"    "peopl"  "realli" "right"  "say"    "see"    "thank"
## [36] "thing"  "think"  "time"   "vegan"  "video"  "want"   "way"
## [43] "whi"    "will"
```

We can do a basic correlation analysis by looking at the correlations between terms with the `findAssocs` function. If two words always appear together then corr = 1. If two terms never appear together then corr = 0. Let's look at which terms co-occur with the term "meat", with a lower correlation limit of 0.25.

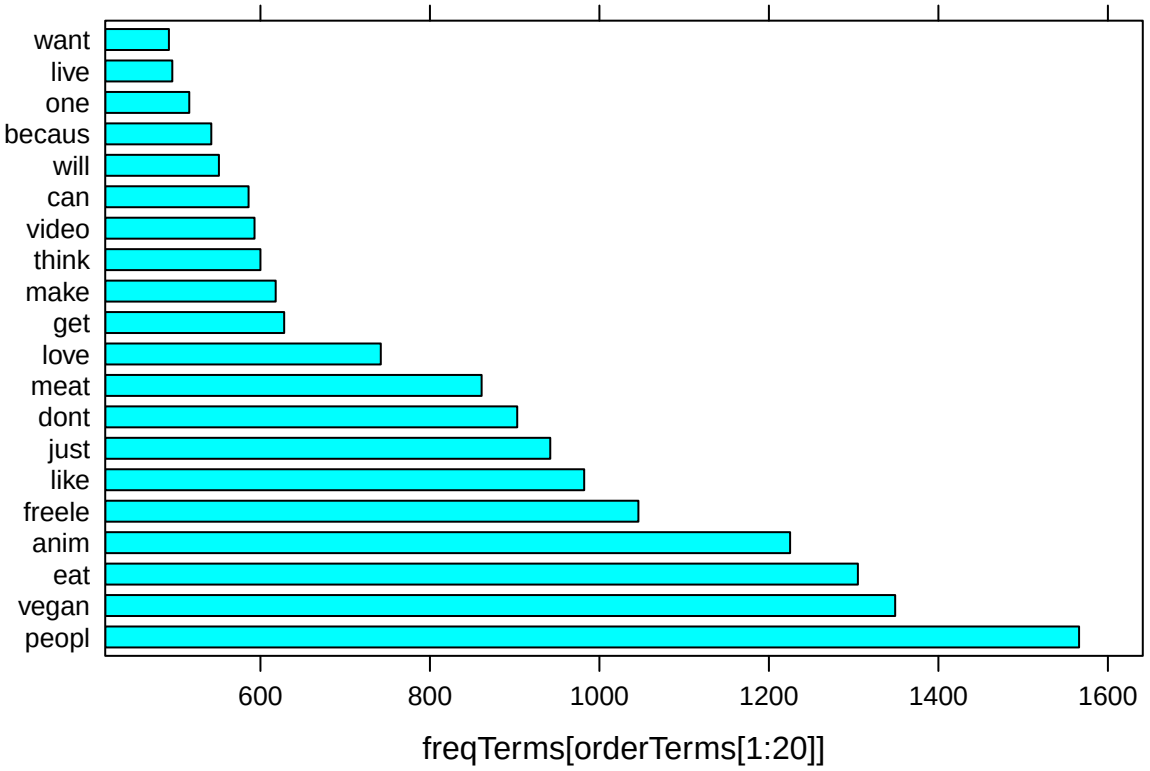
```
findAssocs(dtmSparseRemoved, "meat", corlimit=0.25)
```

```
##        meat
## eat    0.62
## eater  0.41
## anim   0.40
## human  0.34
## becaus 0.31
## peopl  0.30
## vegan  0.30
## kill   0.28
## dont   0.26
## think  0.26
## food   0.25
## reason 0.25
```

Text visualisation

We can plot our descriptive statistics in various ways. For example, using a barchart to visualise the 20 most frequent terms:

```
barchart(freqTerms[orderTerms[1:20]])
```



Wordclouds enable us to visualise term frequency in a very attractive way. We draw on the wordcloud package and provide various parameters to 'tune' the output of our wordcloud:

```
wordcloud(names(freqTerms), freqTerms, min.freq=40,max.words=100,colors=brewer.pal(6, "Dark2"),scale=c(5
```

```
## Warning in wordcloud(names(freqTerms), freqTerms, min.freq = 40, max.words
## = 100, : anim could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(names(freqTerms), freqTerms, min.freq = 40, max.words
## = 100, : freele could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(names(freqTerms), freqTerms, min.freq = 40, max.words
## = 100, : vegan could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(names(freqTerms), freqTerms, min.freq = 40, max.words
## = 100, : becaus could not be fit on page. It will not be plotted.
```