Basic Text Mining in R

Philip Murphy, PhD

4/4/2017

Basic Text Mining in R

Back to the QDA Course Site (https://sites.google.com/site/qdaatmiis/) **Note: The QDA Course Site** is open only to students that are, or have been, registered for the Qualitative Data Analysis course at the Middlebury Institute of International Studies at Monterey. (http://miis.edu/)

To start, install the packages you need to mine text
You only need to do this step once.

```
Needed <- c("tm", "SnowballCC", "RColorBrewer", "ggplot2", "wordcloud", "biclust",
    "cluster", "igraph", "fpc")
install.packages(Needed, dependencies = TRUE)

install.packages("Rcampdf", repos = "http://datacube.wu.ac.at/", type = "source")
```

If you get the following message:
'Update all/some/none? [a/s/n]:' enter "a" and press return

Loading Texts

Start by saving your text files in a folder titled: "texts" This will be the "corpus" (body) of texts you are mining.

Note: The texts used in this example are a few of Donald Trump's speeches that were copied and pasted into a text document. You can use a variety of media for this, such as PDF and HTML. The text example was chosen out of curiosity. If you would like to use these same texts, you can download them here. (https://drive.google.com/open?id=0B914dXn0AXvlaWhmVXdPclZrTjA)

Read this next part carefully. You need to do three unique things here:
1. Create a file named "texts" where you'll keep your data.
2. Save the file to a particular place
+ **Mac**: Desktop
+ **PC**: C: drive
3. Copy and paste the appropriate scripts below.

On a Mac, save the folder to your desktop and use the following code chunk:

```
cname <- file.path("~", "Desktop", "texts")
cname
```

```
## [1] "~/Desktop/texts"
```

```
dir(cname)   # Use this to check to see that yo
ur texts have loaded.
```

```
##  [1] "Trump Black History Month Speech.txt"
##  [2] "Trump CIA Speech.txt"
##  [3] "Trump Congressional Address.txt"
##  [4] "Trump CPAC Speech.txt"
##  [5] "Trump Florida Rally 2-18-17.txt"
##  [6] "Trump Immigration Speech 8-31-16.txt"
##  [7] "Trump Inauguration Speech.txt"
##  [8] "Trump National Prayer Breakfast.txt"
##  [9] "Trump Nomination Speech.txt"
## [10] "Trump Police Chiefs Speech.txt"
## [11] "Trump Response to Healthcare Bill Fail
ure.txt"
```

On a PC, save the folder to your C: drive and use the following code chunk:

```
cname <- file.path("C:", "texts")
cname
dir(cname)
```

Load the R package for text mining and then load your texts into R.

```
library(tm)
```

```
## Loading required package: NLP
```

```
docs <- VCorpus(DirSource(cname))
summary(docs)
```

```
##
Length Class
## Trump Black History Month Speech.txt
2      PlainTextDocument
## Trump CIA Speech.txt
2      PlainTextDocument
## Trump Congressional Address.txt
2      PlainTextDocument
## Trump CPAC Speech.txt
2      PlainTextDocument
## Trump Florida Rally 2-18-17.txt
2      PlainTextDocument
## Trump Immigration Speech 8-31-16.txt
2      PlainTextDocument
## Trump Inauguration Speech.txt
2      PlainTextDocument
## Trump National Prayer Breakfast.txt
2      PlainTextDocument
## Trump Nomination Speech.txt
2      PlainTextDocument
## Trump Police Chiefs Speech.txt
2      PlainTextDocument
## Trump Response to Healthcare Bill Failure.tx
t 2      PlainTextDocument
##
Mode
## Trump Black History Month Speech.txt
list
## Trump CIA Speech.txt
list
## Trump Congressional Address.txt
list
## Trump CPAC Speech.txt
list
## Trump Florida Rally 2-18-17.txt
list
## Trump Immigration Speech 8-31-16.txt
list
## Trump Inauguration Speech.txt
list
## Trump National Prayer Breakfast.txt
list
## Trump Nomination Speech.txt
list
## Trump Police Chiefs Speech.txt
list
## Trump Response to Healthcare Bill Failure.tx
t list
```

For details about documents in the corpus, use the `inspect(docs)` command.

```
inspect(docs[1])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document leve
l (indexed): 0
## Content:  documents: 1
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 3974
```

In this case, you are getting the details on only the second document in the corpus. But this is not a lot of information. Essentially, all you get is the number of characters in each document in the corpus. Documents are identified by the number in which they are loaded.

If you so desire, you can read your documents in the R terminal using `writeLines(as.character(docs))` . Or, if you prefer to look at only one of the documents you loaded, then you can specify which one using something like:

```
writeLines(as.character(docs[1]))
```

```
## list(list(content = c("Well, the election, i
t came out really well. Next time we’ll triple
the number or quadruple it. We want to get it o
ver 51, right? At least 51.", "", "Well this is
Black History Month, so this is our little brea
kfast, our little get-together. Hi Lynn, how ar
e you? Just a few notes. During this month, we
honor the tremendous history of African-America
ns throughout our country. Throughout the world
, if you really think about it, right? And thei
r story is one of unimaginable sacrifice, hard
work, and faith in America. I’ve gotten a real
glimpse—during the campaign, I’d go around with
Ben to a lot of different places I wasn’t so fa
miliar with. They’re incredible people. And I w
ant to thank Ben Carson, who’s gonna be heading
up HUD. That’s a big job. That’s a job that’s n
ot only housing, but it’s mind and spirit. Righ
t, Ben? And you understand, nobody’s gonna be b
etter than Ben.",
## "", "Last month, we celebrated the life of R
everend Martin Luther King, Jr., whose incredib
le example is unique in American history. You r
ead all about Dr. Martin Luther King a week ago
when somebody said I took the statue out of my
office. It turned out that that was fake news.
Fake news. The statue is cherished, it’s one of
the favorite things in the—and we have some goo
d ones. We have Lincoln, and we have Jefferson,
and we have Dr. Martin Luther King. But they sa
id the statue, the bust of Martin Luther King,
was taken out of the office. And it was never e
ven touched. So I think it was a disgrace, but
that’s the way the press is. Very unfortunate."
,
## "", "I am very proud now that we have a muse
um on the National Mall where people can learn
about Reverend King, so many other things. Fred
erick Douglass is an example of somebody who’s
done an amazing job and is being recognized mor
e and more, I noticed. Harriet Tubman, Rosa Par
ks, and millions more black Americans who made
America what it is today. Big impact.", "", "I’
m proud to honor this heritage and will be hono
ring it more and more. The folks at the table i
n almost all cases have been great friends and
supporters. Darrell—I met Darrell when he was d
efending me on television. And the people that
were on the other side of the argument didn’t h
ave a chance, right? And Paris has done an amaz
ing job in a very hostile CNN community. He’s a
ll by himself. You’ll have seven people, and Pa
ris. And I’ll take Paris over the seven. But I
don’t watch CNN, so I don’t get to see you as m
uch as I used to. I don’t like watching fake ne
ws. But Fox has treated me very nice. Wherever
Fox is, thank you.",
## "", "We’re gonna need better schools and we
need them soon. We need more jobs, we need bett
er wages, a lot better wages. We’re gonna work
very hard on the inner city. Ben is gonna be do
ing that, big league. That’s one of the big thi
ngs that you’re gonna be looking at. We need sa
fer communities and we’re going to do that with
law enforcement. We’re gonna make it safe. We’r
e gonna make it much better than it is right no
w. Right now it’s terrible, and I saw you talki
ng about it the other night, Paris, on somethin
g else that was really—you did a fantastic job
the other night on a very unrelated show.",
## "", "I’m ready to do my part, and I will say
this: We’re gonna work together. This is a grea
t group, this is a group that’s been so special
to me. You really helped me a lot. If you remem
ber I wasn’t going to do well with the African-
American community, and after they heard me spe
aking and talking about the inner city and lots
of other things, we ended up getting—and I won’
t go into details—but we ended up getting subst
antially more than other candidates who had run
in the past years. And now we’re gonna take tha
t to new levels. I want to thank my television
star over here—Omarosa’s actually a very nice p
erson, nobody knows that. I don’t want to destr
oy her reputation but she’s a very good person,
and she’s been helpful right from the beginning
of the campaign, and I appreciate it. I really
do. Very special.",
## "", "So I want to thank everybody for being
here."), meta = list(author = character(0), dat
etimestamp = list(sec = 7.1230309009552, min =
40, hour = 1, mday = 6, mon = 3, year = 117, wd
ay = 4, yday = 95, isdst = 0), description = ch
aracter(0), heading = character(0), id = "Trump
Black History Month Speech.txt", language = "en
", origin = character(0))))
## list()
## list()
```

In this case, you will have called up only the second document you loaded - as indicated by the ‘[2]’. Be careful. Either of these commands will fill up your screen fast.

Preprocessing

Once you are sure that all documents loaded properly, go on to preprocess your texts.
This step allows you to remove numbers, capitalization, common words, punctuation, and otherwise prepare your texts for analysis.
This can be somewhat time consuming and picky, but it pays off in the end in terms of high quality analyses.

Removing punctuation:
Your computer cannot actually read. Punctuation and other special characters only look like more words to your computer and R. Use the following to methods to remove them from your text.

```
docs <- tm_map(docs,removePunctuation)
# writeLines(as.character(docs[1])) # Check to
see if it worked.
    # The 'writeLines()' function is commented
out to save space.
```

If necesasry, such as when working with standardized documents or emails, you can remove special characters.
This list has been customized to remove punctuation that you commonly find in emails. You can customize what is removed by changing them as you see fit, to meet your own unique needs.

```
for (j in seq(docs)) {
    docs[[j]] <- gsub("/", " ", docs[[j]])
    docs[[j]] <- gsub("@", " ", docs[[j]])
    docs[[j]] <- gsub("\\|", " ", docs[[j]])
    docs[[j]] <- gsub("\u2028", " ", docs[[j]])
# This is an ascii character that did not trans
late, so it had to be removed.
}
# writeLines(as.character(docs[1])) # You can c
heck a document (in this case
# the first) to see if it worked.
```

Removing numbers:

```
docs <- tm_map(docs, removeNumbers)
# writeLines(as.character(docs[1])) # Check to
see if it worked.
```

Converting to lowercase:
As before, we want a word to appear exactly the same every time it appears. We therefore change everything to lowercase.

```
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, PlainTextDocument)
DocsCopy <- docs
# writeLines(as.character(docs[1])) # Check to
see if it worked.
```

Removing "stopwords" (common words) that usually have no analytic value.
In every text, there are a lot of common, and uninteresting words (a, and, also, the, etc.). Such words are frequent by their nature, and will confound your analysis if they remain in the text.

```
# For a list of the stopwords, see:
# length(stopwords("english"))
# stopwords("english")
docs <- tm_map(docs, removeWords, stopwords("en
glish"))
docs <- tm_map(docs, PlainTextDocument)
# writeLines(as.character(docs[1])) # Check to
see if it worked.
```

Removing particular words:
If you find that a particular word or two appear in the output, but are not of value to your particular analysis. You can remove them, specifically, from the text.

```
docs <- tm_map(docs, removeWords, c("syllogism"
, "tautology"))
# Just remove the words "syllogism" and "tautol
ogy".
# These words don't actually exist in these tex
ts. But this is how you would remove them if th
ey had.
```

Combining words that should stay together
If you wish to preserve a concept is only apparent as a collection of two or more words, then you can combine them or reduce them to a meaningful acronym before you begin the analysis. Here, I am using examples that are particular to qualitative data analysis.

```
for (j in seq(docs))
{
  docs[[j]] <- gsub("fake news", "fake_news", d
ocs[[j]])
  docs[[j]] <- gsub("inner city", "inner-city",
docs[[j]])
  docs[[j]] <- gsub("politically correct", "pol
itically_correct", docs[[j]])
}
docs <- tm_map(docs, PlainTextDocument)
```

Removing common word endings (e.g., “ing”, “es”, “s”)
This is referred to as “stemming” documents. We stem the documents so that a word will be recognizable to the computer, despite whether or not it may have a variety of possible endings in the original text.

Note: The “stem completion” function is currently problemmatic, and stemmed words are often annoying to read. For now, I have this section commented out. But you are welcome to try these functions (by removing the hashmark from the beginning of the line) if they interest you. Just don’t expect them to operate smoothly.

This procedure has been a little hanky in the recent past, so I change the name of the data object when I do this to keep from overwriting what I have done to this point.

```
## Note: I did not run this section of code for
this particular example.
docs_st <- tm_map(docs, stemDocument)
docs_st <- tm_map(docs_st, PlainTextDocument)
writeLines(as.character(docs_st[1])) # Check to
see if it worked.
# docs <- docs_st
```

Then add common endings to improve intrepretability.

```
# This appears not to be working right now. You
are welcome to try it, but there are numerous r
eports of
#   the stemCompletion function not being curre
ntly operational.
# Note: This code was not run for this particul
ar example either.
#   Read it as a potential how-to.
docs_stc <- tm_map(docs_st, stemCompletion, dic
tionary = DocsCopy, lazy=TRUE)
docs_stc <- tm_map(docs_stc, PlainTextDocument)
writeLines(as.character(docs_stc[1])) # Check t
o see if it worked.
# docs <- docs_stc
```

If the stemming and/or stem completion worked, then convert the corpus beck to "docs" using 'docs <- docs_st' or 'docs <- docs_stc'.

Stripping unnecesary whitespace from your documents:
The above preprocessing will leave the documents with a lot of "white space". White space is the result of all the left over spaces that were not removed along with the words that were deleted. The white space can, and should, be removed.

```
docs <- tm_map(docs, stripWhitespace)
# writeLines(as.character(docs[1])) # Check to
see if it worked.
```

To Finish
Be sure to use the following script once you have completed preprocessing.
This tells R to treat your preprocessed documents as text documents.

```
docs <- tm_map(docs, PlainTextDocument)
```

This is the end of the preprocessing stage.

Stage the Data

To proceed, create a document term matrix.
This is what you will be using from this point on.

```
dtm <- DocumentTermMatrix(docs)
dtm
```

```
## <<DocumentTermMatrix (documents: 11, terms:
3610)>>
## Non-/sparse entries: 8376/31334
## Sparsity           : 79%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

To inspect, you can use: `inspect(dtm)`
This will, however, fill up your terminal quickly. So you may prefer to view a subset:
`inspect(dtm[1:5, 1:20])` view first 5 docs & first 20 terms - modify as you like
`dim(dtm)` This will display the number of documents & terms (in that order)

You’ll also need a transpose of this matrix. Create it using:

```
tdm <- TermDocumentMatrix(docs)
tdm
```

```
## <<TermDocumentMatrix (terms: 3610, documents
: 11)>>
## Non-/sparse entries: 8376/31334
## Sparsity           : 79%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

Explore your data

Organize terms by their frequency:

```
freq <- colSums(as.matrix(dtm))
length(freq)
```

```
## [1] 3610
```

```
ord <- order(freq)
```

If you prefer to export the matrix to Excel:

```
m <- as.matrix(dtm)
dim(m)
```

```
## [1]   11 3610
```

```
write.csv(m, file="DocumentTermMatrix.csv")
```

Focus!

Er, that is, you can focus on just the interesting stuff…
The 'removeSparseTerms()' function will remove the infrequently used words, leaving only the most well-used words in the corpus.

```
#  Start by removing sparse terms:
dtms <- removeSparseTerms(dtm, 0.2) # This make
s a matrix that is 20% empty space, maximum.
dtms
```

```
## <<DocumentTermMatrix (documents: 11, terms:
91)>>
## Non-/sparse entries: 886/115
## Sparsity           : 11%
## Maximal term length: 11
## Weighting          : term frequency (tf)
```

Word Frequency

There are a lot of terms, so for now, just check out some of the **most and least frequently occurring words.**

```
freq <- colSums(as.matrix(dtm))
```

Check out the frequency of frequencies.
The 'colSums()' function generates a table reporting how often *each word frequency* occurs. Using the 'head()" function, below, we can see the distribution of the least-frequently used words.

```
head(table(freq), 20) # The ", 20" indicates th
at we only want the first 20 frequencies. Feel
free to change that number.
```

```
## freq
##    1    2    3    4    5    6    7    8    9
10   11   12   13   14   15
## 1602  615  324  212  126  103   77   57   60
51   38   32   29   24   22
##   16   17   18   19   20
##   16    8   15   15   11
```

The resulting output is two rows of numbers. The top number is the frequency with which words appear and the bottom number reflects how many words appear that frequently. Here, considering only the 20 lowest word frequencies, we can see that 1602 terms appear only once. There are also a lot of others that appear very infrequently.

For a look at the most frequently used terms, we can use the 'tail()' function.

```
tail(table(freq), 20) # The ", 20" indicates th
at we only want the last 20 frequencies.  Feel
free to change that number, as needed.
```

```
## freq
##  76  77  79  83  88  95 101 102 105 107 111
122 127 139 140 163 174 265
##   1   1   2   2   1   1   2   2   1   1   1
1   1   1   1   1   1   1
## 279 428
##   1   1
```

Considering only the 20 greatest frequencies, we can see that there is a huge disparity in how frequently some terms appear.

For a less, fine-grained look at term freqency we can view a table of the terms we selected when we removed sparse terms, above. (Look just under the word "Focus".)

```
freq <- colSums(as.matrix(dtms))
freq
```

```
##        also      always     america    ameri
can     another        back 
##          54          24         122         
107          22          75 
##         bad     believe         big         c
ame         can        care 
##          35          60          45          
20         101          37 
##        come     country         day   differ
ent        done        dont 
##          55         174          36          
16          24          95 
## enforcement        even        ever         ev
ery         get     getting 
##          43          55          42          
49          79          23 
##        give       going        good        gr
eat       group      happen 
##          25         265          58         
163          20          36 
##         ive         job        just         k
now        last         law 
##          35          38          88         
127          44          59 
##         let        life        like       lit
tle        long        look 
##          40          27          79          
24          36          52 
##         lot        love        made         m
any        much        must 
##          44          45          33         
101          68          53 
##      nation        need       never          
new         now      office 
##          48          32          83          
69         111          24 
##         one      people   president          
put      really    remember 
##         139         279          46          
35          57          27 
##       right        safe        said          
say         see        seen 
##         102          35          83          
66          48          34 
##   something     special      states         t
ake        tell       thank 
##          25          26          67          
64          50         105 
##       thats      things       think         t
ime       today    together 
```

```
##          74          40          77          
76          33          34 
##     totally       truly  understand      uni
ted        want         way 
##          18          16          29          
64         140          71 
##        well        weve        will        w
ork       world        year 
##          55          43         428          
64          56          47 
##       years 
##          54
```

The above matrix was created using a data transformation we made earlier. What follows is an alternative that will accomplish essentially the same thing.

```
freq <- sort(colSums(as.matrix(dtm)), decreasin
g=TRUE)
head(freq, 14)
```

```
##     will   people    going  country    great 
want      one     know 
##      428      279      265      174      163 
140      139      127 
##  america      now american    thank    right 
theyre 
##      122      111      107      105      102 
102
```

An alternate view of term frequency:
This will identify all terms that appear frequently (in this case, 50 or more times).

```
findFreqTerms(dtm, lowfreq=50)   # Change "50"
to whatever is most appropriate for your text d
ata.
```

```
##  [1] "also"        "america"     "american"
"back"        "believe"
##  [6] "can"         "come"        "country"
"dont"        "even"
## [11] "get"         "going"       "good"
"great"       "immigration"
## [16] "jobs"        "just"        "know"
"law"         "like"
## [21] "look"        "make"        "many"
"much"        "must"
## [26] "never"       "new"         "now"
"one"         "people"
## [31] "really"      "right"       "said"
"say"         "states"
## [36] "take"        "tell"        "thank"
"thats"       "theyre"
## [41] "think"       "time"        "united"
"want"        "way"
## [46] "well"        "will"        "work"
"world"       "years"
```

Yet another way to do this:

```
wf <- data.frame(word=names(freq), freq=freq)
head(wf)
```

```
##            word freq
## will       will  428
## people   people  279
## going     going  265
## country country  174
## great     great  163
## want       want  140
```

Plot Word Frequencies

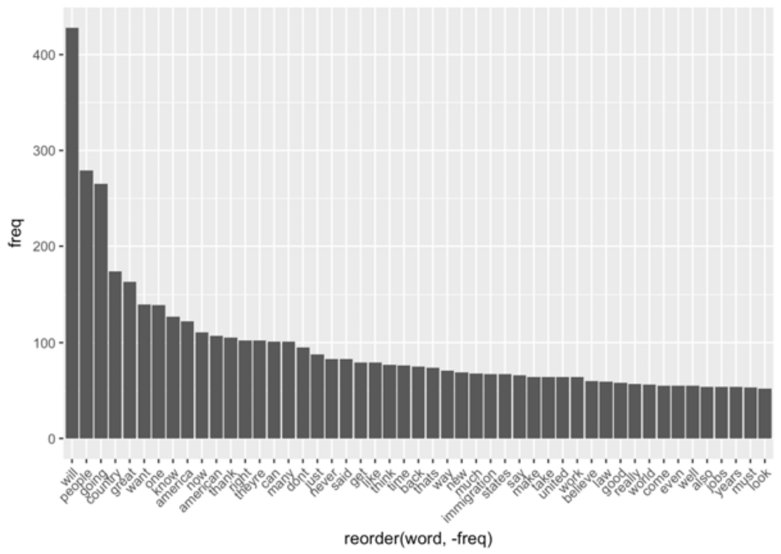
Plot words that appear at least 50 times.

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package
:NLP':
##
##     annotate
```

```
p <- ggplot(subset(wf, freq>50), aes(x = reorde
r(word, -freq), y = freq)) +
          geom_bar(stat = "identity") +
          theme(axis.text.x=element_text(angle=
45, hjust=1))
p
```



Relationships Between Terms

Term Correlations

If you have a term in mind that you have found to be particularly meaningful to your analysis, then you may find it helpful to identify the words that most highly correlate with that term.

If words always appear together, then correlation=1.0.

```
findAssocs(dtm, c("country" , "american"), corl
imit=0.85) # specifying a correlation limit of
0.85
```

```
## $country
##    cities countries   nothing      jobs       
come   biggest    donors 
##      0.94      0.94      0.94      0.92      
0.91      0.90      0.90 
##    second     begin    border      plan    c
rimes     globe     meant 
##      0.90      0.88      0.88      0.88      
0.87      0.87      0.87 
## thousands     means   workers      also   de
spite      take 
##      0.87      0.86      0.86      0.85      
0.85      0.85 
## 
## $american
##  restore     task     fair   budget    cycle 
new promises  dollars 
##     0.97     0.93     0.92     0.91     0.89 
0.89     0.89     0.88 
##  finally millions national     tens  foreign 
middle  justice  program 
##     0.88     0.88     0.88     0.88     0.87 
0.87     0.86     0.86 
##    break  joining   united 
##     0.85     0.85     0.85
```

In this case, “country” and “american” were highly correlated with numerous other terms. Setting `corlimit=` to 0.85 prevented the list from being overly long. Feel free to adjust the `corlimit=` to any value you feel is necessary.

```
findAssocs(dtms, "think", corlimit=0.70) # spec
ifying a correlation limit of 0.95
```

```
## $think
## really   well   care    lot happen    see 
good 
##   0.89   0.87   0.86   0.76   0.72   0.72   
0.71
```

Word Clouds!

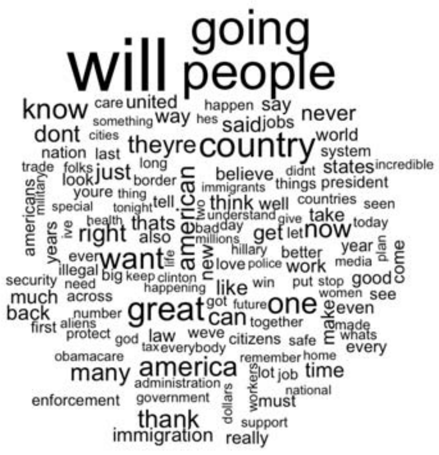
Humans are generally strong at visual analytics. That is part of the reason that these have become so popular. What follows are a variety of alternatives for constructing word clouds with your text.

But first you will need to load the package that makes word clouds in R.

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

Plot words that occur at least 25 times.

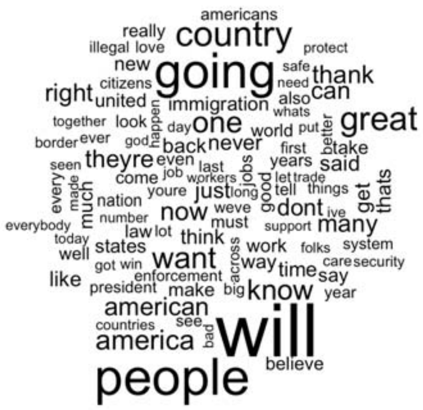
```
set.seed(142)
wordcloud(names(freq), freq, min.freq=25)
```



Note: The “set.seed() function just makes the configuration of the layout of the clouds consistent each time you plot them. You can omit that part if you are not concerned with preserving a particular layout.

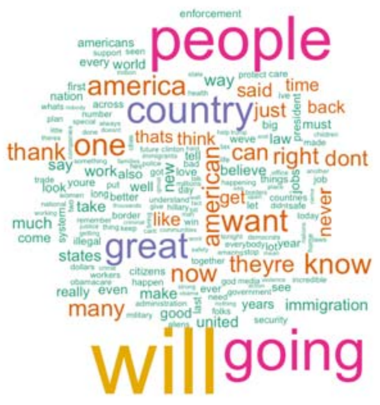
Plot the 100 most frequently used words.

```
set.seed(142)
wordcloud(names(freq), freq, max.words=100)
```



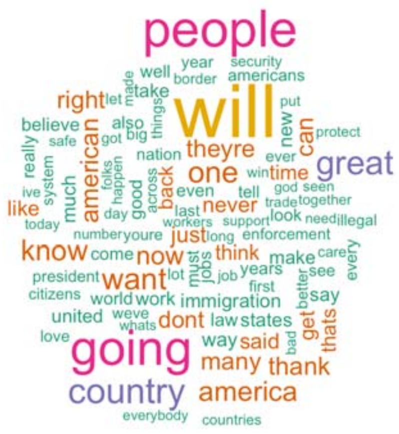
Add some color and plot words occurring at least 20 times.

```
set.seed(142)
wordcloud(names(freq), freq, min.freq=20, scale
=c(5, .1), colors=brewer.pal(6, "Dark2"))
```



Plot the 100 most frequently occurring words.

```
set.seed(142)
dark2 <- brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100, rot
.per=0.2, colors=dark2)
```



Clustering by Term Similarity

To do this well, you should always first remove a lot of the uninteresting or infrequent words. If you have not done so already, you can remove these with the following code.

```
dtmss <- removeSparseTerms(dtm, 0.15) # This ma
kes a matrix that is only 15% empty space, maxi
mum.
dtmss
```

```
## <<DocumentTermMatrix (documents: 11, terms:
45)>>
## Non-/sparse entries: 472/23
## Sparsity           : 5%
## Maximal term length: 9
## Weighting          : term frequency (tf)
```

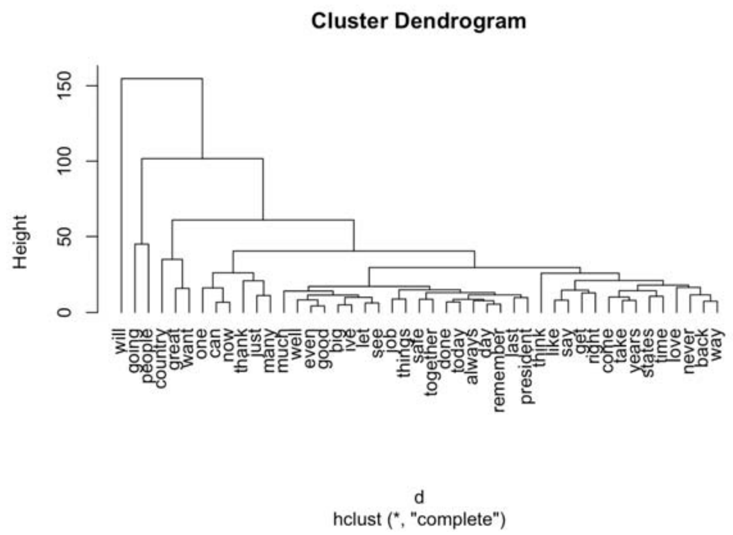
Hierarchal Clustering

First calculate distance between words & then cluster them according to similarity.

```
library(cluster)
d <- dist(t(dtmss), method="euclidian")
fit <- hclust(d=d, method="complete")   # for a
different look try substituting: method="ward.D
"
fit
```

```
##
## Call:
## hclust(d = d, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 45
```

```
plot(fit, hang=-1)
```

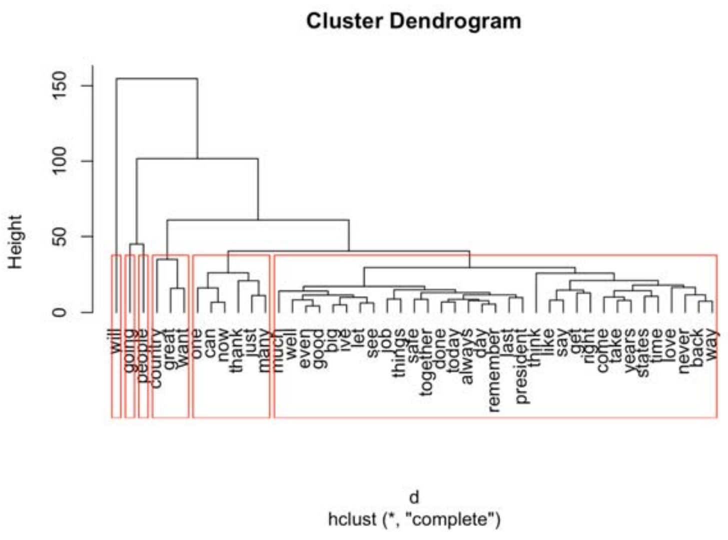


Some people find dendrograms to be fairly clear to read. Others simply find them perplexing. Here, we can see two, three, four, five, six, seven, or many more groups that are identifiable in the dendrogram.

Helping to Read a Dendrogram

If you find dendrograms difficult to read, then there is still hope. To get a better idea of where the groups are in the dendrogram, you can also ask R to help identify the clusters. Here, I have arbitrarily chosen to look at five clusters, as indicated by the red boxes. If you would like to highlight a different number of groups, then feel free to change the code accordingly.

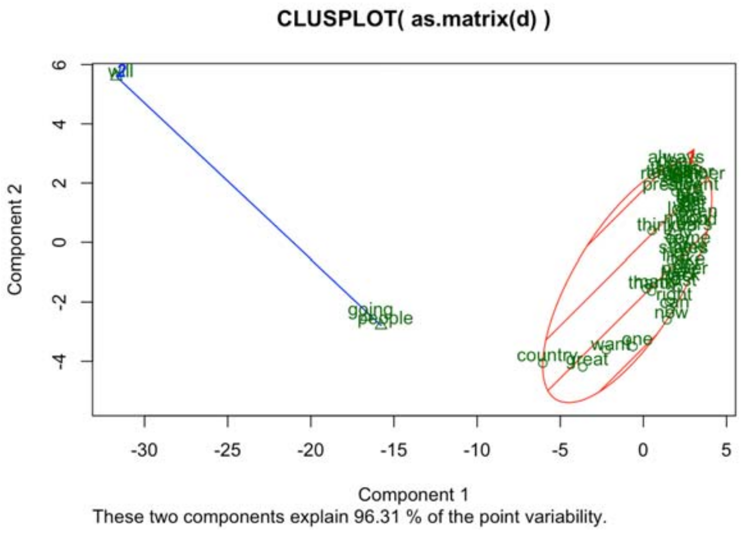
```
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=6)   # "k=" defines the number of clusters you are using
rect.hclust(fit, k=6, border="red") # draw dendogram with red borders around the 6 clusters
```



K-means clustering

The k-means clustering method will attempt to cluster words into a specified number of groups (in this case 2), such that the sum of squared distances between individual words and one of the group centers. You can change the number of groups you seek by changing the number specified within the `kmeans()` command.

```
library(fpc)
d <- dist(t(dtmss), method="euclidian")
kfit <- kmeans(d, 2)
clusplot(as.matrix(d), kfit$cluster, color=T, s
hade=T, labels=2, lines=0)
```



To learn more about text mining specifically, or data mining in general, there is an online course by Graham Williams (http://onepager.togaware.com/), author of Data Mining with Rattle and R: The Art of Excavating Data for Knowledge

Discovery (http://www.amazon.com/gp/product/1441998896/ref=as_li_qf_sp_asin_tl?ie=UTF8&tag=togaware-20&linkCode=as2&camp=217145&creative=399373&creativeASIN=1441998896). Quite a lot of the material above was initially derived from his "Text Mining" segment.

Back to the QDA Mashup Site (https://sites.google.com/site/miisqda/)

Just the Basics

If all you want to do is try this out, then use the abbreviated scripts below. You can always go back to read the material above if you want to better understand what you have done later.

Install needed R packages
You only need to do this once

```
# **To start,** install the packages you need to mine text
#      You only need to do this step once.

Needed <- c("tm", "SnowballCC", "RColorBrewer", "ggplot2", "wordcloud", "biclust", "cluster", "igraph", "fpc")
install.packages(Needed, dependencies=TRUE)

install.packages("Rcampdf", repos = "http://datacube.wu.ac.at/", type = "source")

# If you get the following message:
#       Update all/some/none? [a/s/n]:
#   enter "a" and press return
###############################################################################################
```

Tell your computer where to find the texts
Read this next part carefully. You need to to three unique things here:
1. Create a file named "texts" where you'll keep your data.
2. Save the file to a particular place
+ **Mac**: Desktop
+ **PC**: C: drive
3. Copy and paste the appropriate scripts below.

```
###############################################
###########################################
#                                  Loading Text
s                                          #
###############################################
###########################################
#
#     Start by saving your text files in a fold
er titled:    "texts"
#     This will be the "corpus" (body) of texts
you are mining.
#
#     Next, choose the type of computer you hav
e...



######
# **On a Mac**, save the folder to your *deskto
p* and use the following code chunk:
######
cname <- file.path("~", "Desktop", "texts")
cname
dir(cname)   # Use this to check to see that yo
ur texts have loaded.



######
# *On a PC*, save the folder to your *C: drive*
and use the following code chunk:
######
cname <- file.path("C:", "texts")
cname
dir(cname)
###############################################
###########################################
###############################################
###########################################
```

Run the Analyses

```
################################################
############################################
#                                Start Your Ana
lyses                                      #
################################################
############################################
# **Load the R package for text mining and then
load your texts into R.**
library(tm)
docs <- VCorpus(DirSource(cname))
## Preprocessing
docs <- tm_map(docs,removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, removeWords, stopwords("en
glish"))
docs <- tm_map(docs, removeWords, c("syllogism"
, "tautology"))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, PlainTextDocument)
# *This is the end of the preprocessing stage.*


### Stage the Data
dtm <- DocumentTermMatrix(docs)
tdm <- TermDocumentMatrix(docs)

### Explore your data
freq <- colSums(as.matrix(dtm))
length(freq)
ord <- order(freq)
m <- as.matrix(dtm)
dim(m)
write.csv(m, file="DocumentTermMatrix.csv")
### FOCUS - on just the interesting stuff...
#  Start by removing sparse terms:
dtms <- removeSparseTerms(dtm, 0.1) # This make
s a matrix that is 10% empty space, maximum.
### Word Frequency
head(table(freq), 20)
# The above output is two rows of numbers. The
top number is the frequency with which
# words appear and the bottom number reflects h
ow many words appear that frequently.
#
tail(table(freq), 20)
# Considering only the 20 greatest frequencies
#
# **View a table of the terms after removing sp
arse terms, as above.
```

```
freq <- colSums(as.matrix(dtms))
freq
# The above matrix was created using a data tra
nsformation we made earlier.
# **An alternate view of term frequency:**
# This will identify all terms that appear freq
uently (in this case, 50 or more times).
findFreqTerms(dtm, lowfreq=50)   # Change "50"
to whatever is most appropriate for your data.
#
#
#
### Plot Word Frequencies
# **Plot words that appear at least 50 times.**
library(ggplot2)
wf <- data.frame(word=names(freq), freq=freq)
p <- ggplot(subset(wf, freq>50), aes(word, freq
))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=4
5, hjust=1))
p
#
## Relationships Between Terms
### Term Correlations
# See the description above for more guidance w
ith correlations.
# If words always appear together, then correla
tion=1.0.
findAssocs(dtm, c("country" , "american"), corl
imit=0.85) # specifying a correlation limit of
0.85
findAssocs(dtms, "think", corlimit=0.70) # spec
ifying a correlation limit of 0.95
#
# Change "country" & "american", or "think" to
terms that actually appear in your texts.
# Also adjust the `corlimit= ` to any value you
feel is necessary.
#
#
### Word Clouds!
# First load the package that makes word clouds
in R.
library(wordcloud)
dtms <- removeSparseTerms(dtm, 0.15) # Prepare
the data (max 15% empty space)
freq <- colSums(as.matrix(dtm)) # Find word fre
quencies
dark2 <- brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100, rot
.per=0.2, colors=dark2)

### Clustering by Term Similarity

### Hierarchal Clustering
dtms <- removeSparseTerms(dtm, 0.15) # This mak
es a matrix that is only 15% empty space.
library(cluster)
d <- dist(t(dtms), method="euclidian")   # Firs
t calculate distance between words
fit <- hclust(d=d, method="complete")    # Also
try: method="ward.D"
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=6)   # "k=" defines the
number of clusters you are using
rect.hclust(fit, k=6, border="red") # draw dend
ogram with red borders around the 5 clusters

### K-means clustering
library(fpc)
library(cluster)
dtms <- removeSparseTerms(dtm, 0.15) # Prepare
the data (max 15% empty space)
d <- dist(t(dtms), method="euclidian")
kfit <- kmeans(d, 2)
clusplot(as.matrix(d), kfit$cluster, color=T, s
hade=T, labels=2, lines=0)
```

Back to the QDA Course Site (https://sites.google.com/site/qdaatmiis/) **Note: The QDA Course Site** is open only to students that are, or have been, registered for the Qualitative Data Analysis course at the Middlebury Institute of International Studies at Monterey. (http://miis.edu/)