Basic Text Mining in R

- Loading Texts
- Preprocessing
  - Stage the Data
- Explore your data
  - Focus!
  - Word Frequency
  - Plot Word Frequencies
- Relationships Between Terms
  - Term Correlations
  - Word Clouds!
- Clustering by Term Similarity
  - Hierarchal Clustering
  - K-means clustering
- Just the Basics

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 websites about qualitative data analysis 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 because it most closely matches the data used in the QDA Mash-up class.

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.

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 your texts have loaded.
```

```
## [1] "ch 4.txt"    "excel.txt"   "nursing.txt" "wiki.txt"    "ws 1.txt"
## [6] "ws1.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)
docs <- Corpus(DirSource(cname))

summary(docs)
```

```
## A corpus with 6 text documents
##
## The metadata consists of 2 tag-value pairs and a data frame
## Available tags are:
##   create_date creator
## Available variables in the data frame are:
##   MetaID
```

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

```
inspect(docs[2])
```

In this case, you would call up only the second document you loaded. 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)
# inspect(docs[3]) # Check to see if it worked.
```

If necesasry, such as when working with 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]])
  }
# inspect(docs[1]) # You can check a document (in this case the first) to see if it worked.
```

Removing numbers:

```
docs <- tm_map(docs, removeNumbers)
# inspect(docs[3]) # 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)
# inspect(docs[3]) # 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("english"))
# inspect(docs[3]) # 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("department", "email"))
# Just replace "department" and "email" with words that you would like to remove.
```

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("qualitative research", "QDA", docs[[j]])
docs[[j]] <- gsub("qualitative studies", "QDA", docs[[j]])
docs[[j]] <- gsub("qualitative analysis", "QDA", docs[[j]])
docs[[j]] <- gsub("research methods", "research_methods", docs[[j]])
}
```

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.

```
  library(SnowballC)
docs <- tm_map(docs, stemDocument)
# inspect(docs[3]) # Check to see if it worked.
```

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)
# inspect(docs[3]) # 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
```

```
## A document-term matrix (6 documents, 2197 terms)
##
## Non-/sparse entries: 3867/9315
## Sparsity           : 71%
## Maximal term length: 40
## 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
```

```
## A term-document matrix (2197 terms, 6 documents)
##
## Non-/sparse entries: 3867/9315
## Sparsity           : 71%
## Maximal term length: 40
## Weighting          : term frequency (tf)
```

Explore your data

Organize terms by their frequency:

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

```
## [1] 2197
```

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

```
 If you prefer to export the matrix to Excel:
 m <- as.matrix(dtm)
 dim(m)
 write.csv(m, file="dtm.csv")
```

Focus!

Er, that is, you can focus on just the interesting stuff...

```
#  Start by removing sparse terms:
dtms <- removeSparseTerms(dtm, 0.1) # This makes a matrix that is 10% empty space, maximum.
inspect(dtms)
```

```
## A document-term matrix (6 documents, 24 terms)
##
## Non-/sparse entries: 144/0
## Sparsity           : 0%
## Maximal term length: 9
## Weighting          : term frequency (tf)
##
##       Terms
## Docs   analysi can categori compar contrast data develop emerg evalu
##   [1,]      61  23        2      8       11  111       8    11    33
##   [2,]       6   4       15      1        1   35       2     1     1
##   [3,]      25   9        1      8        2   47       7     3     1
##   [4,]      19   8        1      1        2   45       5     5     1
##   [5,]      13  25        4      5        1   51       3     4     1
##   [6,]       6   1        3      1        1   13       1     1     1
##       Terms
## Docs   exampl general group help interview new often particip process
##   [1,]     21       4    25    8         7  11    13       51      24
##   [2,]      7       1     6    1         3   2     3        4       3
##   [3,]      8       5     3    5         5  11     3        1      29
##   [4,]      3       1     5    6        13   9     4        7       1
##   [5,]     11       4     3    5         5  22     4        4       9
##   [6,]      2       1     3    1         3   1     1        1       2
##       Terms
## Docs   qualit question research studi use way
##   [1,]     48       32       13    11  22  19
##   [2,]      4       11        2     9  10   1
##   [3,]     28        5       28    23  21   8
##   [4,]     28        6       50    19  17   6
##   [5,]      7        7       10     3  21  25
##   [6,]     11        3        2     1   2   2
```

Word Frequency

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

```
freq[head(ord)]
```

```
##          absent     abstractfre abstractionedit         accentu
##               1               1               1               1
##        accommod       accompani
##               1               1
```

```
freq[tail(ord)]
```

```
## research   qualit      qda  analysi     code     data
##      105      126      128      130      203      302
```

Check out the frequency of frequencies.

```
head(table(freq), 20)
```

```
## freq
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 986 319 183 128  99  68  56  46  40  43  24  21  17  14  11  12  10   7
##  19  20
##  10   6
```

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 995 terms appear only once. There are also a lot of others that appear very infrequently.

```
tail(table(freq), 20)
```

```
## freq
##  49  50  52  53  56  58  61  63  64  66  68  70  73  93 105 126 128 130
##   1   1   3   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
## 203 302
##   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
```

```
##   analysi       can  categori    compar  contrast      data   develop
##       130        70        26        24        18       302        26
##     emerg     evalu    exampl   general     group      help interview
##        25        38        52        16        45        26        36
##       new     often  particip   process    qualit  question  research
##        56        28        68        68       126        64       105
##     studi       use       way
##        66        93        61
```

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)), decreasing=TRUE)
head(freq, 14)
```

```
##     data     code  analysi      qda   qualit research      use   differ
##      302      203      130      128      126      105       93       73
##      can particip  process    studi question   campus
##       70       68       68       66       64       63
```

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 data.
```

```
##  [1] "analysi"  "campus"   "can"      "code"     "data"     "differ"
##  [7] "exampl"   "may"      "method"   "might"    "new"      "one"
## [13] "particip" "process"  "qda"      "qualit"   "question" "research"
## [19] "set"      "studi"    "use"      "way"
```

Yet another way to do this:

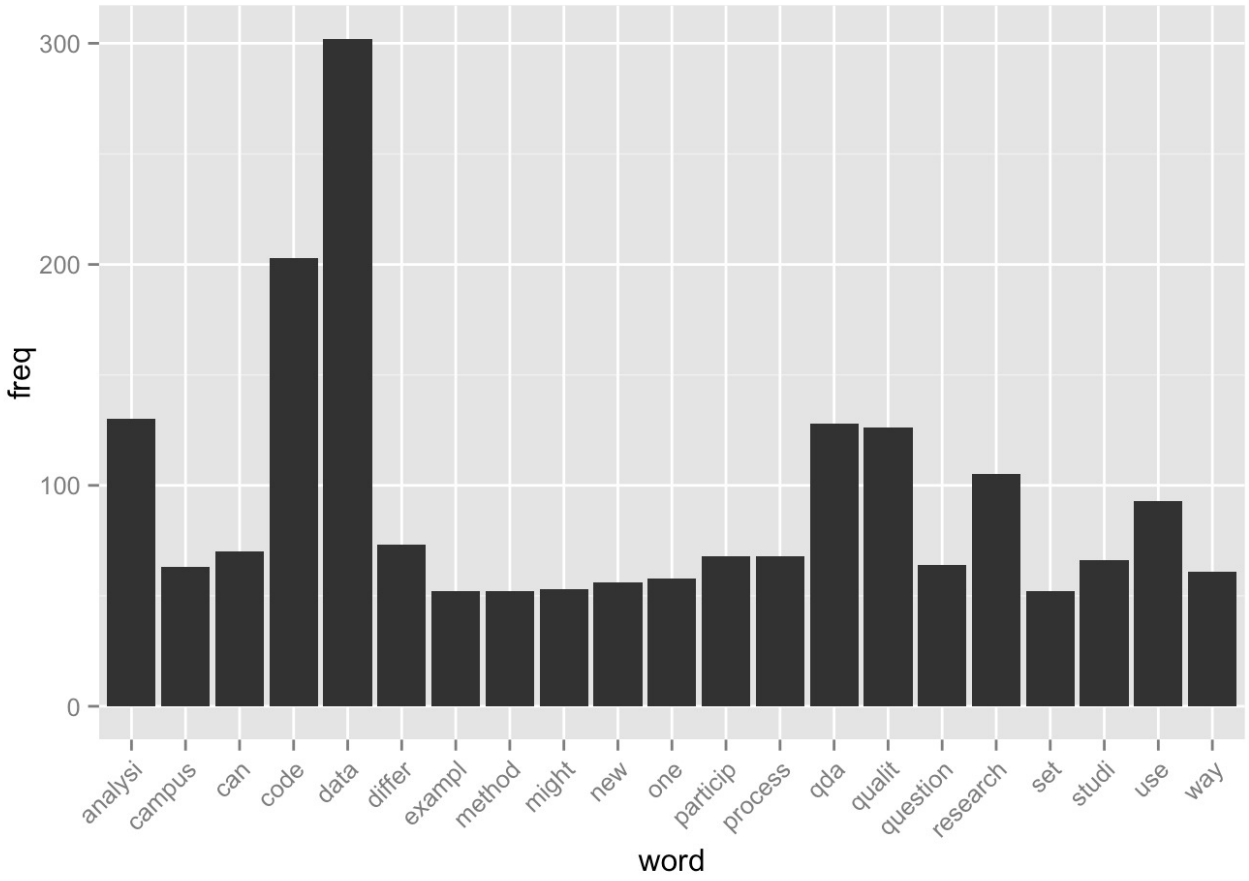
```
wf <- data.frame(word=names(freq), freq=freq)
head(wf)
```

```
##              word freq
## data         data  302
## code         code  203
## analysi   analysi  130
## qda           qda  128
## qualit     qualit  126
## research research  105
```

Plot Word Frequencies

Plot words that appear at least 50 times.

```
library(ggplot2)
p <- ggplot(subset(wf, freq>50), aes(word, freq))
p <- p + geom_bar(stat="identity")
p <- p + 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("question" , "analysi"), corlimit=0.98) # specifying a correlation limit of 0.98
```

```
##         question analysi
## across      0.99    0.99
## flow        0.99    0.99
## format      0.99    0.99
## group       0.99    0.99
## less        0.99    0.99
## matter      0.99    0.98
## multipl     0.99    0.98
## eight       0.98    0.98
## still       0.98    0.98
## time        0.98    0.98
```

In this case, “question” and “analysi” were highly correlated with numerous other terms. Setting `corlimit=` to 0.98 prevented the list from being overly long. Feel free to adjust the `corlimit=` to any value you feel is necessary.

```
findAssocs(dtms, "contrast", corlimit=0.90) # specifying a correlation limit of 0.95
```

```
##          contrast
## evalu        0.99
## particip     0.99
## group        0.98
## analysi      0.97
## often        0.97
## question     0.95
## data         0.93
## emerg        0.93
```

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.

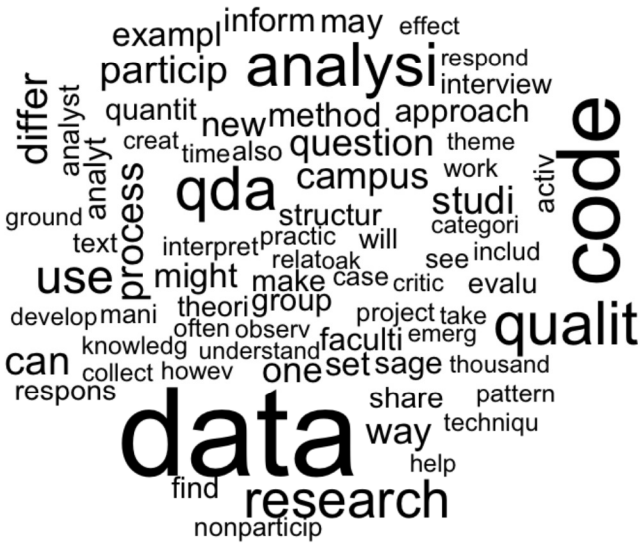
But first you will need to load the package that makes word clouds in R.

```
library(wordcloud)
```

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

Plot words that occur at least 25 times.

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



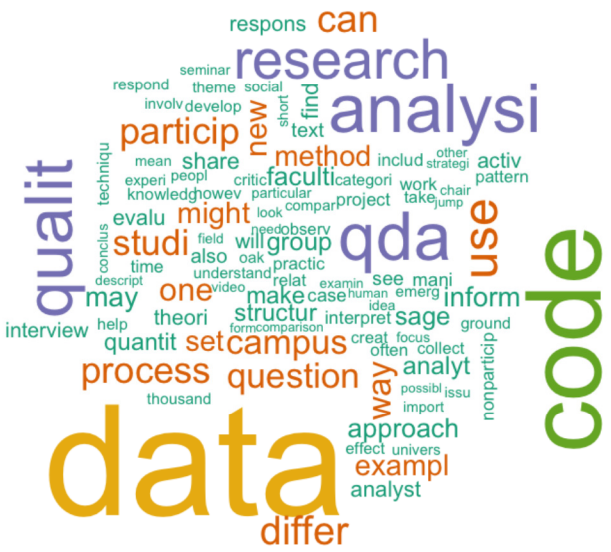
Note: The “set.seed() function just makes the configuration of the layout of the clouds consistent each time you plot them. You can omit that part if you are not concerned with preserving a particular layout.

Plot the 100 most frequently used words.

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

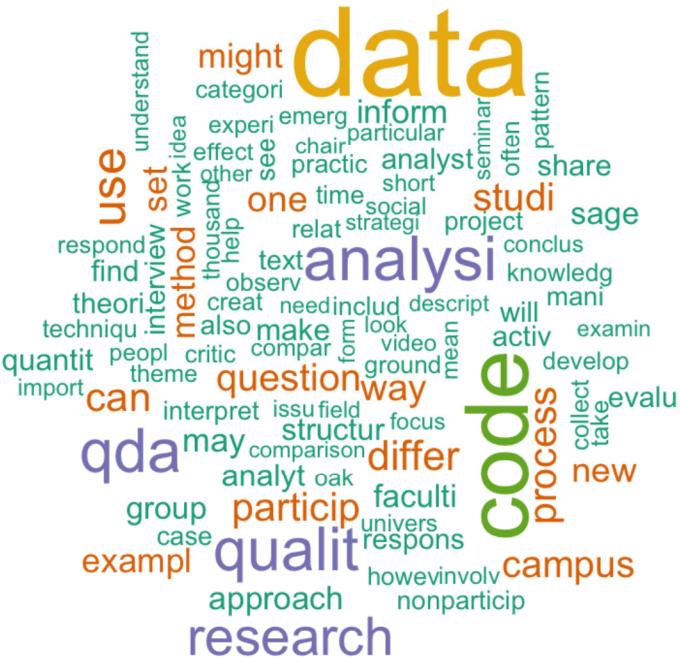
Add some color and plot words occurring at least 20 times.

```
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 makes a matrix that is only 15% empty space, maximum.
inspect(dtmss)
```

```
## A document-term matrix (6 documents, 24 terms)
##
## Non-/sparse entries: 144/0
## Sparsity           : 0%
## Maximal term length: 9
## Weighting          : term frequency (tf)
##
##       Terms
## Docs   analysi can categori compar contrast data develop emerg evalu
##   [1,]      61  23        2      8       11  111       8    11    33
##   [2,]       6   4       15      1        1   35       2     1     1
##   [3,]      25   9        1      8        2   47       7     3     1
##   [4,]      19   8        1      1        2   45       5     5     1
##   [5,]      13  25        4      5        1   51       3     4     1
##   [6,]       6   1        3      1        1   13       1     1     1
##       Terms
## Docs   exampl general group help interview new often particip process
##   [1,]     21       4    25    8         7  11    13       51      24
##   [2,]      7       1     6    1         3   2     3        4       3
##   [3,]      8       5     3    5         5  11     3        1      29
##   [4,]      3       1     5    6        13   9     4        7       1
##   [5,]     11       4     3    5         5  22     4        4       9
##   [6,]      2       1     3    1         3   1     1        1       2
##       Terms
## Docs   qualit question research studi use way
##   [1,]     48       32       13    11  22  19
##   [2,]      4       11        2     9  10   1
##   [3,]     28        5       28    23  21   8
##   [4,]     28        6       50    19  17   6
##   [5,]      7        7       10     3  21  25
##   [6,]     11        3        2     1   2   2
```

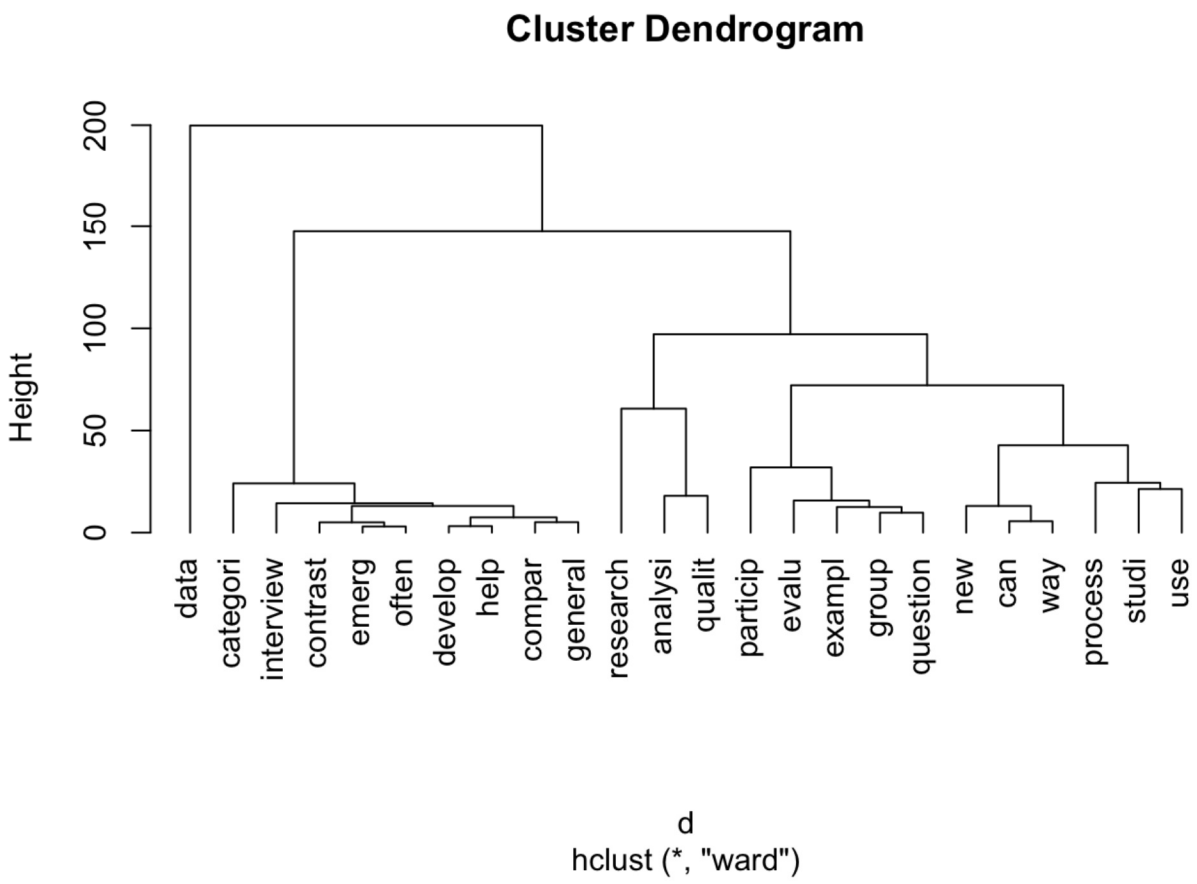
Hierarchal Clustering

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

```
library(cluster)
d <- dist(t(dtmss), method="euclidian")
fit <- hclust(d=d, method="ward")
fit
```

```
##
## Call:
## hclust(d = d, method = "ward")
##
## Cluster method   : ward
## Distance         : euclidean
## Number of objects: 24
```

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



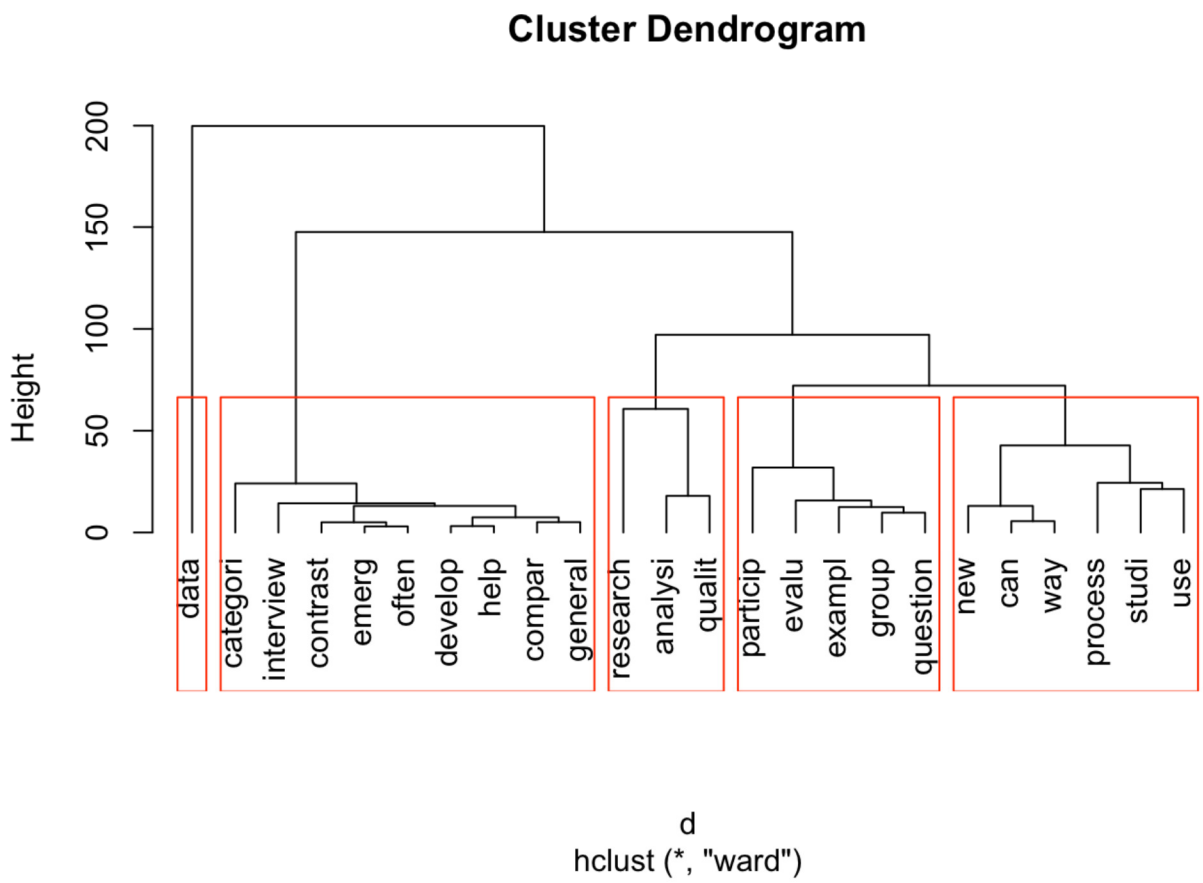
Some people find dendrograms to be fairly clear to read. Others simply find them perplexing. Here, we can see two, three, four, five, six, seven, or many more groups that are identifiable in the dendrogram.

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.

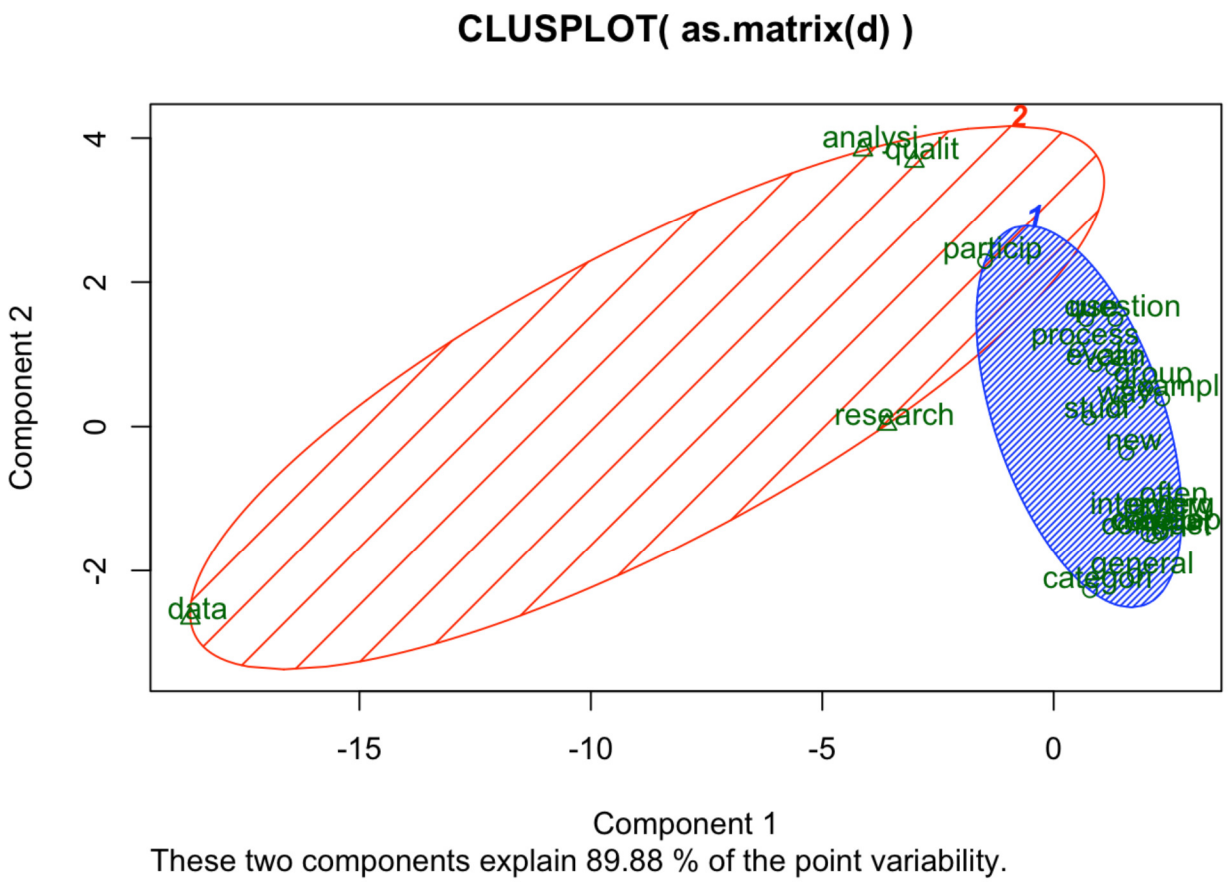
```
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=5)   # "k=" defines the number of clusters you are using
rect.hclust(fit, k=5, border="red") # draw dendogram with red borders around the 5 clusters
```



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, shade=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 (https://web-beta.archive.org/web/20160320075118/http://onepager.togaware.com/), author of Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery (https://web-beta.archive.org/web/20160320075118/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 comes almost directly from his “Text Mining” segment.

Back to the QDA Mashup Site (https://web-beta.archive.org/web/20160320075118/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 Texts                               #
################################################################################
#
#     Start by saving your text files in a folder titled:    "texts"
#     This will be the "corpus" (body) of texts you are mining.
#
#     Next, choose the type of computer you have...



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

```
################################################################################################
#                                  Start Your Analyses                                         #
################################################################################################
# **Load the R package for text mining and then load your texts into R.**
library(tm)
docs <- Corpus(DirSource(cname))
## Preprocessing
docs <- tm_map(docs, removePunctuation)   # *Removing punctuation:*
docs <- tm_map(docs, removeNumbers)      # *Removing numbers:*
docs <- tm_map(docs, tolower)   # *Converting to lowercase:*
docs <- tm_map(docs, removeWords, stopwords("english"))   # *Removing "stopwords"
library(SnowballC)
docs <- tm_map(docs, stemDocument)   # *Removing common word endings* (e.g., "ing", "es")
docs <- tm_map(docs, stripWhitespace)   # *Stripping whitespace
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 makes 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 how many words appear that frequently.
#
tail(table(freq), 20)
# Considering only the 20 greatest frequencies
#
# **View a table of the terms after removing sparse terms, as above.
freq <- colSums(as.matrix(dtms))
freq
# The above matrix was created using a data transformation we made earlier.
# **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 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=45, hjust=1))
p
#
## Relationships Between Terms
### Term Correlations
# See the description above for more guidance with correlations.
# If words always appear together, then correlation=1.0.
findAssocs(dtm, c("question" , "analysi"), corlimit=0.98) # specifying a correlation limit of 0.98
#
# Change "question" & "analysi" 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 frequencies
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 makes a matrix that is only 15% empty space.
library(cluster)
d <- dist(t(dtms), method="euclidian")   # First calculate distance between words
fit <- hclust(d=d, method="ward")
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=5)   # "k=" defines the number of clusters you are using
rect.hclust(fit, k=5, border="red") # draw dendogram 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, shade=T, labels=2, lines=0)
```

Back to the QDA Mashup Site (https://web-beta.archive.org/web/20160320075118/https://sites.google.com/site/miisqda/)