Text Mining and Word Clouds

Using R to make qualitative text analysis easier

Leah Figueroa, MA
Austin Community College
TAIR 2017

Why Word Clouds

Word clouds

- Simple, easy to use data visualization
- Reveals key points
- Provide an emotional connection to text
- Effective communication
- Provide clarity
- Fast
- Engaging

January 2011 State of the Union Address

Before we begin

- You will need
  - R
  - RStudio
    - If you don’t have these, there is an excellent walk through by University o Health System
      - https://www.hsl.virginia.edu/services/howdoi/how-do-i-install-r-and-rstudio
  - Text file(s) of the document (s) you wish to create word clouds from

Installing the appropriate packages

- In R, the fundamental unit of shareable code is the package. A package bundles together code, data, documentation, and tests, and is easy to share with others.
- For text mining, you will need the following packages
  - tm (text mining)
  - SnowballCC (collapses words to a common root to aid comparison of vocabulary)
  - RColorBrewer (Provides color schemes for maps (and other graphics))
  - ggplot2 (plotting system)
  - wordcloud (word cloud)
  - biclust (find biclusters of data in 2-D shape)
  - cluster (cluster analysis)
  - igraph (graphs)
  - fpc (flexible procedures for clustering)
  - Rcampdf (reads pdf) *** requires manual install

Installing the appropriate packages

- Start up Rstudio.
- Select Tools.
- Select Install Packages.
- Type NAME OF PACKAGE into Packages
- Make sure Install dependencies is checked.
- Select Install.
- If RStudio asks: “Do you want to install from sources the package which needs compilation? y/n:”Select y.
- Manually install Rcampdf (reads pdf).
    - install.packages(“Rcampdf”, repos = “http://datacube.wu.ac.at/”, type = “source”)

Creating a text file

- Copy and paste the text in a plain text file (e.g : lbjamericanpromise1965.txt)
- Save the file

http://www.lbjlibrary.org/lyndon-baines-johnson/speeches-films/president-johnsons-special-message-to-the-congress-the-american-promise

Loading Your Texts

Step 1:

getwd()

If your working directory is not correct, you will need to set your working directory. Mine is located below.

"C:/Users/l1719843/Documents/Documents for Qual Analysis/lbj/lbjspeech"

setwd(“ Fill in your own directory here, but remember to use / vs \ “)

getwd()

Step 2:

```
library(tm)
Loading required package: NLP
 #Create Corpus

docs <- Corpus(DirSource( Fill in your own directory here, but remember to use / vs \ “))
```

Loading Your Texts, Continued

Step 2:

```
library(tm)
Loading required package: NLP
 #Create Corpus

docs <- Corpus(DirSource( Fill in your own directory here, but remember to use / vs \ “))

docs <-Corpus(DirSource("C:/Users/l1719843/Documents/Documents for Qual Analysis/lbj/lbjspeech"))
```

Cleaning Your Text

- The *tm* package helps make data cleaning easier by using transformations
- To see the available transformations type *getTransformations()* at the R prompt:

  > getTransformations()

  [1] “removeNumbers” “removePunctuation” “removeWords” “stemDocument” “stripWhitespace”
- In addition to the prepackaged transformations, you will sometimes need to create a custom transformations.
  - In this case, LBJ’s speech uses “--” and that will need to be removed, otherwis

    the text document.

Cleaning Your Text - Custom Transformations

- In this example, we will build a content transformer, which will call *cleanText*

```
#create the cleanText content transformer
cleanText <- content_transformer(function(x, pattern) {return (gsub(pattern, ” “, x))})
```

- Now we can use this content transformer to eliminate hyphens:

```
docs <- tm_map(docs, cleanText, “-“)
```

- Be sure to inspect your document(s) after every transformation

```
#inspect a particular document
writeLines(as.character(docs[[1]])
```

Cleaning Your Text - Stripping Punctuation

```
#Remove punctuation – replace punctuation marks with ” “
docs <- tm_map(docs, removePunctuation)
```

Example of text after punctuation removed:

And so at the request of your beloved Speaker and the Senator from Montana the majority leader the Senator from Illinois the minority leader Mr McCulloch and other Members of both parties I came here tonight not as President Roosevelt came down one time in person to veto a bonus bill not as President Truman came down one time to urge the passage of a railroad bill but I came down here to ask you to share this task with me and to share it with the people that we both work for I want this to be the Congress Republicans and Democrats alike which did all these things for all these people

Cleaning Text, continued

For the next steps, we will do the following:

- **Convert the corpus, or document, to all lower case and;**
- **Remove all numbers**

```
#Transform to lower case (need to wrap in content_transformer)
docs <- tm_map(docs,content_transformer(tolower))

#Strip digits (std transformation, so no need for content_transformer)
docs <- tm_map(docs, removeNumbers)
```

Example text before:

I never thought then in 1928 that I would be standing here in 1965 It never even occurred to me in my fondest dreams that I might have the chance to help the sons and daughters of those students and to help people like them all over this country

Example text after:

i never thought then in that i would be standing here in it never even occurred to me in my fondest dreams that i might have the chance to help the sons and daughters of those students and to help people like them all over this country

Cleaning Text - Removing “Stop Words”

Stop words are common words that should be removed from the text.

Stop words include:

- articles (*a, an, the*)
- conjunctions (*and, or but* etc.)
- common verbs (*is*)
- qualifiers (*yet, however* etc)

The tm package includes a standard list of such *stop words* as they are referred to.

```
#remove stopwords using the standard list in tm
docs <- tm_map(docs, removeWords, stopwords(“english”))
```

Example of stop word removal

Example text before:

and so at the request of your beloved speaker and the senator from montana the majority leader the senator from illinois the minority leader mr mcculloch and other members of both parties i came here tonight not as president roosevelt came down one time in person to veto a bonus bill not as president truman came down one time to urge the passage of a railroad bill but i came down here to ask you to share this task with me and to share it with the people that we both work for i want this to be the congress republicans and democrats alike which did all these things for all these people

Example of stop word removal, continued

Example text after:

request beloved speaker senator montana majority leader senator illinois minority leader mr mcculloch members parties came tonight president roosevelt came one time person veto bonus bill president truman came one time urge passage railroad bill came ask share task share people work want congress republicans democrats alike things people

Cleaning Text - Removing White Space

Removing white space allows for a cleaner read

```
#Strip whitespace (cosmetic?)
docs <- tm_map(docs, stripWhitespace)
```

After completeing all transformations, depending on your version of R and/or R studio, you may need to run the following command:

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

Stemming

Stemming allows you to reduce related words to their common root.

An example is offer, offered, and offering. Stemming allows you to reduce all words to offer.

In this example, I have chosen not to stem words

Document Term Matrix (DTM)

The document term matrix is a matrix that lists all occurrences of words across the corpus (the entire body of work) by document.

A simple example might serve to explain the structure of the TDM more clearly. Assume we have a simple corpus consisting of two documents, *Doc1* and *Doc2*, with the following content:

Doc1: cats are fluffy

Doc2: dogs are fluffy

The DTM for this corpus would look like:

	cats	dogs	are	fluffy
Doc1	1	0	1	1
Doc2	0	1	1	1

Document Term Matrix (DTM), Continued

There are two important points when working with DTMs

- DTMs can be huge (number of documents x number of words in the corpus)
- Most words will only appear in a few documents
- As a result of these, a DTM is sparse

To create a DTM in R, use the following:

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

Document Term Matrix (DTM), Continued

You can call up the results from the dtm by typing the variable name into the console and hitting return.

```
dtm
```

<<DocumentTermMatrix (documents: 5, terms: 872)>>

Non-/sparse entries: 874/3486

Sparsity : 80%

Maximal term length: 18

Weighting : term frequency (tf)

This means that 80% of the rows are zero.

Mining the Corpus

Constructing the DTM allows us to turn the text into a mathematical object, which allows us to analyze it.

To get the frequency of occurrence of each word in the corpus, we sum over all rows to give column sums:

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

Inspect the freq.

```
#length should be total number of terms
length(freq)
[1] 872
```

Mining the Corpus

We can review the terms in *freq* by sorting in descending order.

```
#create sort order (descending)
ord <- order(freq,decreasing=TRUE)
```

We can then review both the most and least frequently occurring terms:

```
#inspect most frequently occurring terms
freq[head(ord)]
```

right	will	every	rights	must	american
31	24	19	18	17	16

```
#inspect least frequently occurring terms
freq[tail(ord)]
```

within	wonders	word	wounds	yonder	youth
1	1	1	1	1	1

Removing “Noise”

Words like “can” or “one” aren’t really helpful in a text analysis. Some of them have been removed earlier but we can enforce bounds when creating the DTM.

```
dtmr <-DocumentTermMatrix(docs, control=list(wordLengths=c(4, 20)))
```

This enforces lower and upper limit lengths to the words

If you have more than one document, you can further enforce bounds by picking only certain documents.

```
bounds = list(global = c(x, xx))))
```

Inspecting the new DTM:

```
dtmr
<<DocumentTermMatrix (documents: 5, terms: 831)>>

Non-/sparse entries: 833/3322

Sparsity           : 80%
```

Removing “Noise”

Inspecting the new DTM:

```
dtmr
<<DocumentTermMatrix (documents: 5, terms: 831)>>

Non-/sparse entries: 833/3322

Sparsity           : 80%

Maximal term length: 18

Weighting          : term frequency (tf)
```

Calculating frequencies

```
freqr <- colSums(as.matrix(dtmr))
#length should be total number of terms
length(freqr)
[1] 831
#create sort order (asc)
ordr <- order(freqr,decreasing=TRUE)
#inspect most frequently occurring terms
freqr[head(ordr)]
  right    will   every  rights    must american
     31      24      19      18      17      16
#inspect least frequently occurring terms
freqr[tail(ordr)]
within wonders    word  wounds  yonder   youth
     1       1       1       1       1       1
```

Calculating frequencies, Continued

While not all high frequency words are significant, you can get an idea of potential classifications.

We will find words that occur at least 5 times.

```
findFreqTerms(dtmr,lowfreq=6)
```

```
 [ [1] "american"     "americans"    "bill"         "came"
  [5] "cause"        "come"         "congress"     "constitution"
  [9] "country"      "deny"         "equal"        "every"
 [13] "free"         "government"   "great"        "history"
 [17] "issue"        "just"         "lives"        "many"
 [21] "must"         "nation"       "negroes"      "people"
 [25] "poverty"      "president"    "problem"      "right"
 [29] "rights"       "seek"         "selma"        "shall"
 [33] "time"         "tonight"      "vote"         "voting"
 [37] "want"         "white"        "will"         "world"
```

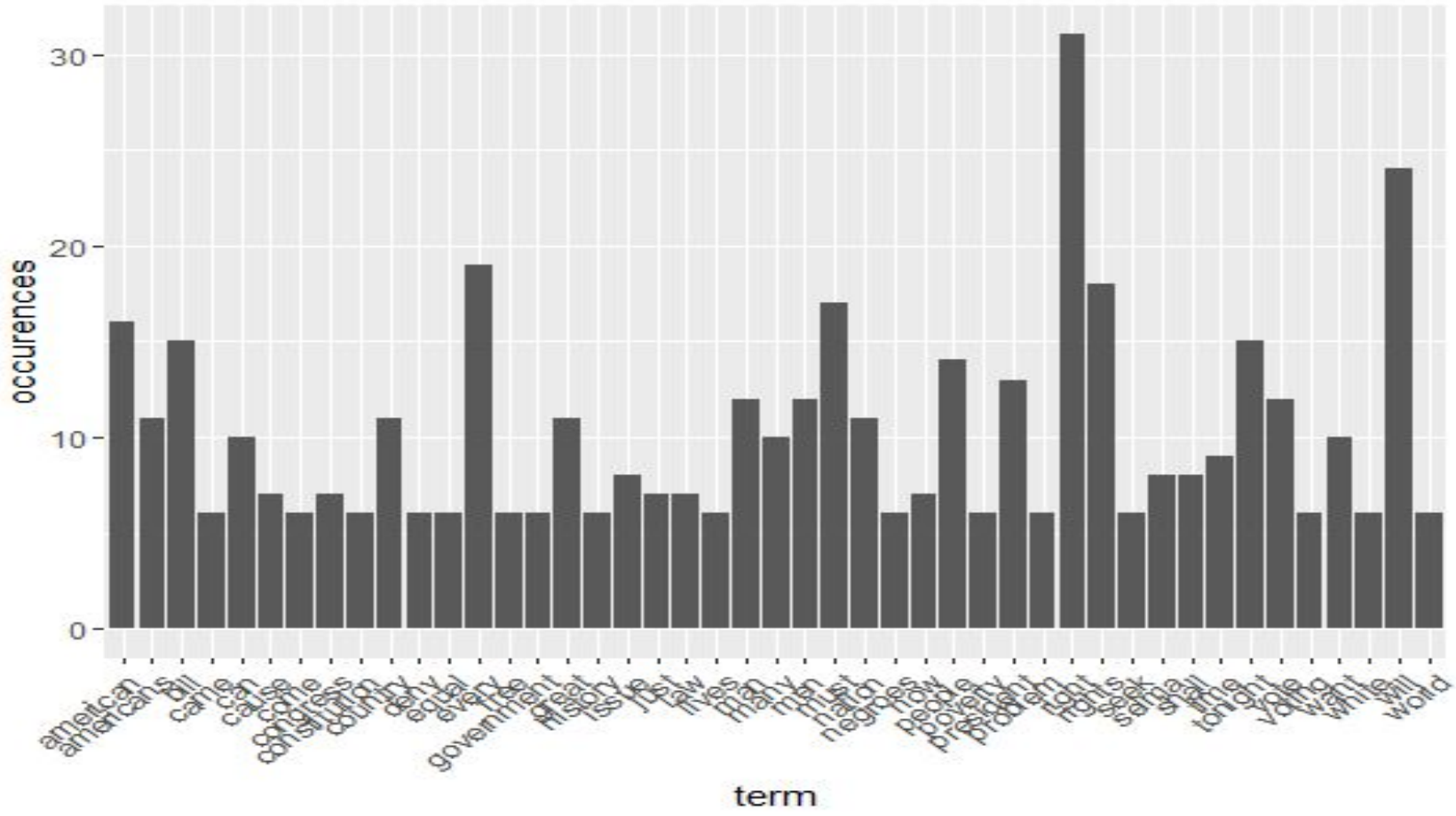
Basic Graphics

We will do a basic frequency histogram.

```
wf=data.frame(term=names(freqr),occurrences=freqr)
```

This creates a data frame, which allows us to plot the occurences of words

```
library(ggplot2)
p <- ggplot(subset(wf, freqr>5), aes(term, occurrences))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```


occurences
30
20
10
0
term
american
americans
bill
came
can
cause
come
congress
constitution
country
day
deny
equal
every
free
government
great
history
issue
just
law
lives
man
many
men
must
nation
negroes
now
people
poverty
president
problem
right
rights
seek
selma
shall
time
tonight
vote
voting
want
white
will
world

What You’ve Been Waiting For!

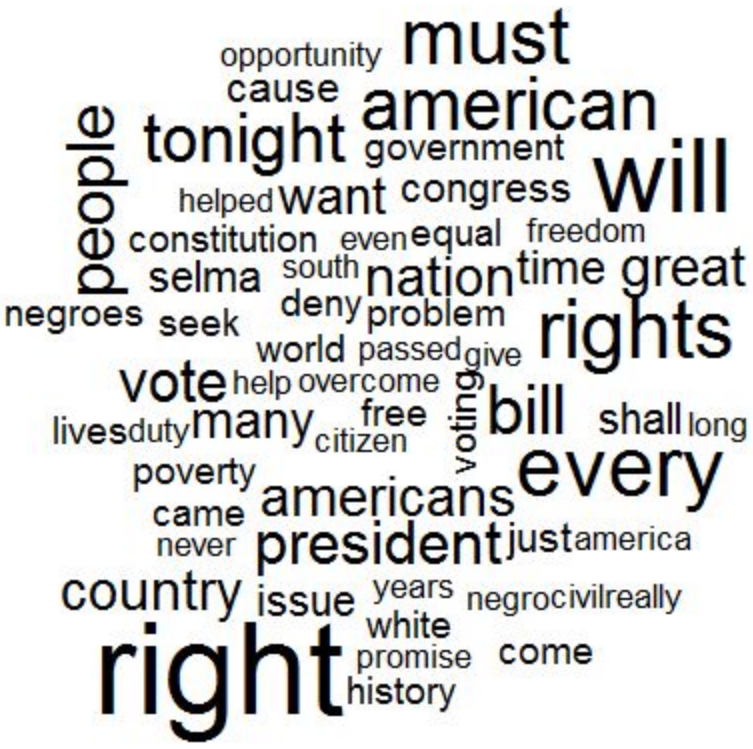
WORD CLOUDS!

Creating a word cloud

```
#wordcloud
library(wordcloud)
#setting the same seed each time ensures consistent look across clouds
set.seed(42)
```

Please note, setting the seed mean you get consistent results each time you run this.

```
#limit words by specifying min frequency
wordcloud(names(freqr),freqr, min.freq=5)
```

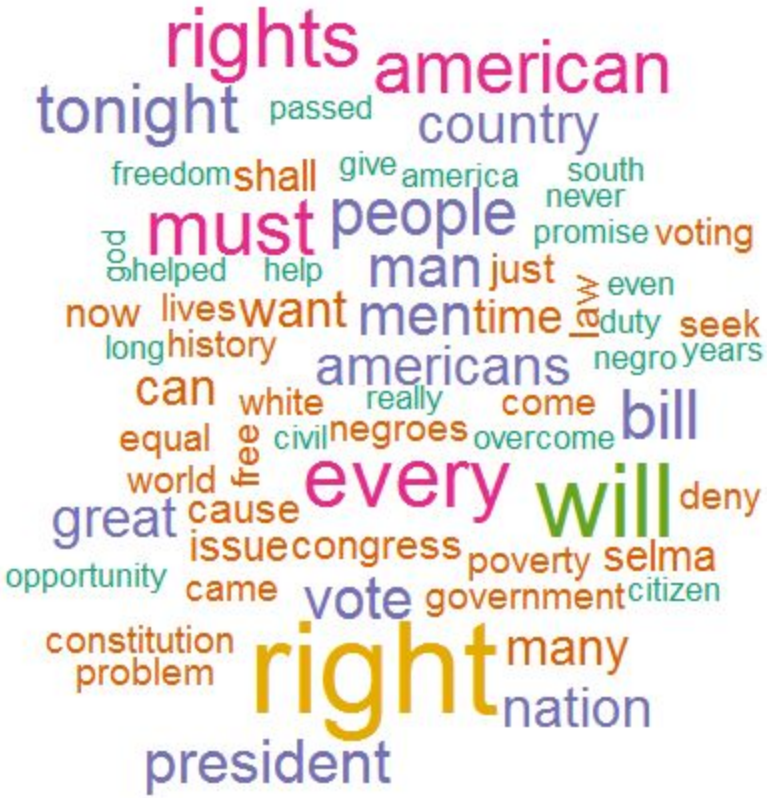

must
opportunity
cause
american
people
tonight
government
will
helped
want
congress
constitution
even
equal
freedom
selma
south
nation
time
great
negroes
seek
deny
problem
rights
world
passed
give
vote
help
overcome
voting
bill
shall
long
lives
duty
many
citizen
free
every
poverty
americans
came
president
just
america
never
country
issue
years
negro
civil
really
white
right
promise
come
history

Making the word cloud colorful!

```
#…add color
wordcloud(names(freqr),freqr,min.freq=70,colors=brewer.pal(6,”Dark2”))
```

Other colors are available:

https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf


rights
american
tonight
passed
country
freedom
shall
give
america
south
never
must
people
promise
voting
god
helped
help
man
just
law
even
now
lives
want
men
time
duty
seek
long
history
negro
years
americans
can
white
really
come
bill
equal
free
civil
negroes
overcome
every
will
world
deny
great
cause
issue
congress
poverty
selma
opportunity
came
vote
government
citizen
constitution
problem
right
many
nation
president

Questions or comments?

Email me at - leah.figueroa@austincc.edu

Thank you very much for attending.