Text Analysis with R

36-600

Fall 2021

Let's Return to Where We Were...

Let's process the Hillary Clinton speech from 2016...

```r
library(stopwords)
lines            <- readLines("http://www.stat.cmu.edu/~pfreeman/clinton.txt")
speech           <- tolower(unlist(strsplit(lines,split="[ ,!\\.]")))
speech           <- speech[speech!=""]
stopword.logical <- speech %in% stopwords::stopwords("en")
clinton          <- data.frame("word"=speech[stopword.logical==FALSE])
head(clinton,2)
```

```
##    word
## 1 thank
## 2  much
```

...and throw in Trump's 2016 speech too, because, why not?...

```r
library(stopwords)
lines            <- readLines("http://www.stat.cmu.edu/~pfreeman/trump.txt")
speech           <- tolower(unlist(strsplit(lines,split="[ ,!\\.]")))
speech           <- speech[speech!=""]
stopword.logical <- speech %in% stopwords::stopwords("en")
trump            <- data.frame("word"=speech[stopword.logical==FALSE])
head(trump,2)
```

```
##        word
## 1   friends
## 2 delegates
```

In both cases, we remove (some) punctuation, remove empty strings, and remove stopwords according to one stopwords dictionary. We then list the words as single columns in data frames.

tidytext

In these notes, we will illustrate some basic text mining methods using functions in the `tidytext` package, and much of the code you will see has been constructed using the examples in the book Text Mining with R by Silge & Robinson (along with some examples of `dplyr` function usage found on, e.g., StackOverflow).

Silge & Robinson define the "tidy text" format as a table with one token per row. (A token is one unit of the text we will analyze. A token could be a single word, contiguous pairs of words, paragraphs, etc. For us, for now, it is a single word...hence the way we stored the words from the Clinton and Trump speeches on the previous page.)

Below, we reformat the data so as to combine the words from Clinton and from Trump into a single data frame:

```r
suppressMessages(library(tidyverse))
library(tidytext)

clinton %>% mutate(speaker="HC") -> clinton
trump   %>% mutate(speaker="DT") -> trump
tidy_speech  <- rbind(clinton,trump)
speech_words <- tidy_speech %>% count(speaker,word,sort=TRUE) # our workhorse dataframe
head(speech_words)
```

```
##   speaker    word  n
## 1      HC  people 37
## 2      DT country 31
## 3      HC      us 31
## 4      DT   going 28
## 5      HC     can 26
## 6      HC    just 26
```

Most-Commonly Used Words

Below we show the 10 most-commonly used words in each speech.

```
speech_words %>% filter(.,speaker=="HC") %>% head(.,10)
```

```
##    speaker      word  n
## 1       HC    people 37
## 2       HC        us 31
## 3       HC       can 26
## 4       HC      just 26
## 5       HC   country 25
## 6       HC   america 23
## 7       HC      know 22
## 8       HC president 20
## 9       HC      make 19
## 10      HC       now 19
```

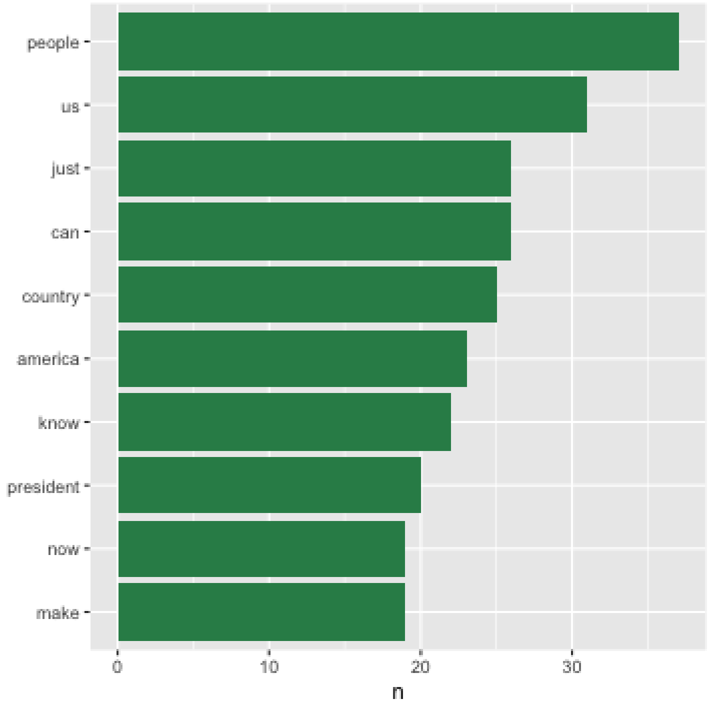
```
speech_words %>% filter(.,speaker=="DT") %>% head(.,10)
```

```
##    speaker      word  n
## 1       DT   country 31
## 2       DT     going 28
## 3       DT   america 25
## 4       DT    people 23
## 5       DT         - 20
## 6       DT       one 17
## 7       DT     every 15
## 8       DT     great 14
## 9       DT     trade 14
## 10      DT americans 13
```

Most-Commonly Used Words

Here we visualize the 10 most-commonly used words in the Clinton speech. Note that if we leave the call to `mutate()` out, then the word order gets shifted into alphabetical order in the plot.

```
speech_words %>% filter(.,speaker=="HC") %>% head(.,10) %>%
  mutate(.,word = reorder(word,n)) %>%
  ggplot(.,mapping=aes(n,word)) + geom_col(fill="seagreen") + labs(y=NULL)
```



Sentiment Analysis

What is the emotional content of a document? Is the film review positive, or negative? What about the tweet? What about the speech? One way to mine information about emotional content is to apply methods of sentiment analysis.

There are a multitude of ways to classify or quantify the sentiment of a word. The `tidytext` package interfaces with three general-purpose sentiment lexicons:

- `AFINN`, which assigns each word a score of -5 (most negative) to 5 (most positive)
- `bing`, which categorizes a subset of words as positive or negative (other words are ignored)
- `nrc`, which extracts words categorized as positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, or trust (see example below)

Sentiment analysis is, of course, going to be imperfect (for instance, the phrase "not good" may ultimately be categorized as positive because the "not" is ignored as a stopword), but it gives some notion of what sentiment underlies a piece of writing.

Sentiment Analysis: Example

As a first example, let's run the speeches through the `bing` lexicon to answer the question "who's more negative...Clinton or Trump?"

```
dt_sentiment <- speech_words %>% filter(.,speaker=="DT") %>%
  inner_join(.,get_sentiments("bing")) %>% suppressMessages(.)
table(dt_sentiment$sentiment)
```

```
##
## negative positive
##      112      110
```

```
hc_sentiment <- speech_words %>% filter(.,speaker=="HC") %>%
  inner_join(.,get_sentiments("bing")) %>% suppressMessages(.)
table(hc_sentiment$sentiment)
```

```
##
## negative positive
##       91      116
```

We see that Trump is more associated with negativity. However...

```
x = matrix(c(110,116,112,91),nrow=2) # first column: positive scores / second column: negative scores
prop.test(x)$p.value
```

```
## [1] 0.2118861
```

...we cannot reject the null hypothesis that the proportion of negative words is equal for both speakers!

Sentiment Analysis: Example

What about anger? Who's more angry?

```r
library(textdata)
speech_words %>% filter(.,speaker=="DT") %>%
  inner_join(.,get_sentiments("nrc")) %>% suppressMessages(.) %>%
  filter(.,sentiment=="anger") -> trump.anger
trump.anger %>% head(.) ; trump.anger %>% select(.,n) %>% sum(.)/nrow(trump)
```

```
##   speaker      word  n sentiment
## 1      DT  opponent 11     anger
## 2      DT  violence 11     anger
## 3      DT terrorism  9     anger
## 4      DT     crime  6     anger
## 5      DT   illegal  6     anger
## 6      DT       bad  4     anger
```

```
## [1] 0.04952741
```

```r
speech_words %>% filter(.,speaker=="HC") %>%
  inner_join(.,get_sentiments("nrc")) %>% suppressMessages(.) %>%
  filter(.,sentiment=="anger") -> clinton.anger
clinton.anger %>% head(.) ; clinton.anger %>% select(.,n) %>% sum(.)/nrow(clinton)
```

```
##   speaker     word n sentiment
## 1      HC     fear 5     anger
## 2      HC    money 5     anger
## 3      HC    words 5     anger
## 4      HC fighting 4     anger
## 5      HC      gun 4     anger
## 6      HC     shot 3     anger
```

```
## [1] 0.0303712
```

tf-idf

In order to apply statistical learning methods to text data, we must map them from an unstructured state (their original format) to a structured state (a data table). One common means of doing this is to measure *term frequency* (tf), the proportion of time a token appears in a document and *inverse-document frequency* (idf), defined as follows

$$\log\left(\frac{n_{\text{documents overall}}}{n_{\text{documents in which term appears}}}\right).$$

and multiplying them together. Intuition: if a term appears in all documents, the idf is zero, so the tf-idf score is zero. Higher scores mean the term was used frequently…but only in a subset of the documents.

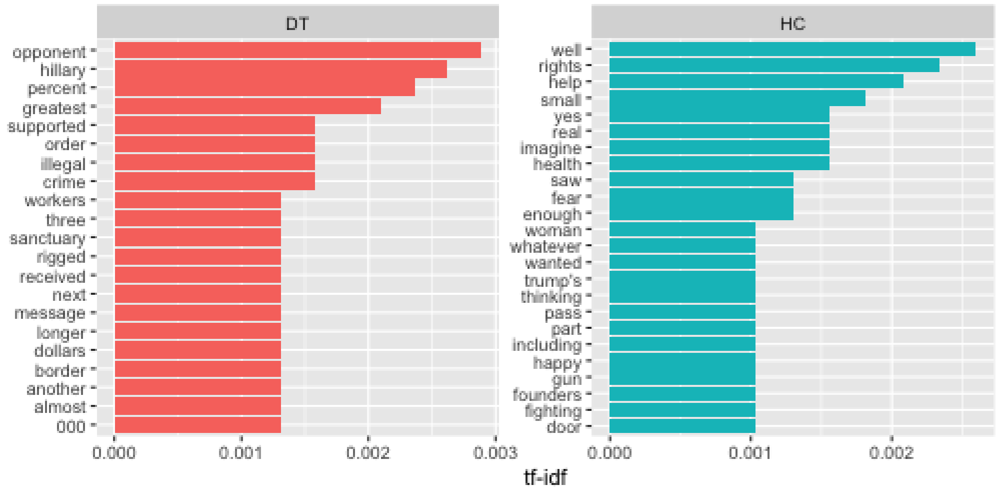
Computing and Visualizing tf-idf Scores

To compute tf-idf scores, we use the `tidytext` function `bind_tf_idf()`.

```
speech_words %>% bind_tf_idf(.,word,speaker,n) -> speech_tf_idf
speech_tf_idf %>% arrange(.,desc(tf_idf)) %>% head(.,1)
```

```
##   speaker     word  n         tf       idf      tf_idf
## 1      DT opponent 11 0.00415879 0.6931472 0.002882654
```

```
library(forcats)
speech_tf_idf %>%
  group_by(.,speaker) %>% slice_max(.,tf_idf,n=15) %>% ungroup(.) %>%
  ggplot(.,aes(tf_idf,fct_reorder(word,tf_idf),fill=speaker)) +
    geom_col(show.legend=FALSE) + facet_wrap(~speaker,ncol=2,scales="free") +
    labs(x="tf-idf",y=NULL)
```



Correlation

However, before we leave, we can use the term-frequency column in the `speech_tf_idf` data frame to ask to question of how correlated the two speeches are. We note that the number by itself is not as interesting as comparing correlations…for instance, would the correlation between an Obama speech and a Clinton speech be higher than that between an Obama speech and a Trump speech? (But remember…you really should perform a hypothesis test on the back end to test whether the two correlations are indeed statistically significantly different.)

```r
# The following code was shamelessly stolen from StackOverflow (and subsequently modified)
tf <- pivot_wider(speech_tf_idf,names_from=speaker,values_from=tf) %>%
  select(.,-n,-idf,-tf_idf) %>%
  gather(.,var,val,-word,na.rm=TRUE) %>%
  group_by(.,word,var) %>%
  distinct(.,val) %>%
  spread(.,var,val) %>%
  replace_na(.,list(HC=0,DT=0))
cor(tf$HC,tf$DT)
```

```
## [1] 0.5320991
```