HW2

Napon Jatusripitak

2/2/2018

1 Data Collection

1.1 Scraping

a. Install the xml2 and rvest packages. Use the read_html function to get the code of the ASEAN membership categories page here.

```
url <- 'https://en.wikipedia.org/wiki/Category:Member_states_of_the_Association_of_Southeast_Asian_Nati
asean <- read_html(url)
```

b. Use the SelectorGadget plugin and html_nodes to get the "nodes" for the links to the country pages.

```
nodes <- asean %>% html_nodes('.mw-category-group a')
nodes <- nodes[2:11]
```

c. Make a dataframe with the country names and the links to the pages (using html_text and html_attr)

```
country_names <- nodes %>% html_text()
country_links <- nodes %>% html_attr('href')

asean_members <- data.frame(
country_names, country_links,
stringsAsFactors = F) %>%
mutate(country_links = paste0("https://en.wikipedia.org", country_links))
knitr::kable(asean_members)
```

country_names	country_links
Brunei	https://en.wikipedia.org/wiki/Brunei
Cambodia	https://en.wikipedia.org/wiki/Cambodia
Indonesia	https://en.wikipedia.org/wiki/Indonesia
Laos	https://en.wikipedia.org/wiki/Laos
Malaysia	https://en.wikipedia.org/wiki/Malaysia
Myanmar	https://en.wikipedia.org/wiki/Myanmar
Philippines	https://en.wikipedia.org/wiki/Philippines
Singapore	https://en.wikipedia.org/wiki/Singapore
Thailand	https://en.wikipedia.org/wiki/Thailand
Vietnam	https://en.wikipedia.org/wiki/Vietnam

d. Using those links, write a loop that 1) gets the code for the page (using read_html), 2) extracts the text from the paragraphs on the page, 3) collapses it into a single string, and 4) saves it to the dataframe.

```
for(i in 1:nrow(asean_members)){
asean_members$member_pages[i] <- asean_members$country_links[i] %>%
read_html() %>%
html_nodes('p') %>%
html_text() %>%
paste(collapse = '\n\n')
}
```

```
head(substring(asean_members$member_pages, 500, 550))
```

```
## [1] "nto two parts by the Sarawak district of Limbang. B"
## [2] "he northwest, Laos to the northeast, Vietnam to the"
## [3] "th more than thirteen thousand islands.[12] At 1,90"
## [4] "n the heart of the Indochinese peninsula of Mainlan"
## [5] "ders with Singapore at the south, Vietnam at the no"
## [6] "ay of Bengal and the Andaman Sea. The country's 201"
```

2 Pre-Processing & Word Frequency Analysis

2.1 Pre-Process

a. Load the Trump tweets into R.

```
trumptweets <- read.csv("trumptweets.csv", quote = "", stringsAsFactors = FALSE)
trumptweets$doc_id <- seq.int(nrow(trumptweets))
trumptweets$time <- as.POSIXct(trumptweets$created_at, format="%m-%d-%Y %H:%M:%S")
trumptweets <- trumptweets %>% select(doc_id, text, source, created_at, retweet_count, favorite_count,
```

b. Pre-process these data using either tm or tidytext. (Discard punctuation, remove capitalization, remove stopwords, remove sparse terms to .01, tokenize, stem)

```
## TM
corp <- VCorpus(DataframeSource(trumptweets), readerControl=list(language="eng"))
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
removeamp <- function(x) gsub("&amp", "", x)
corp <- tm_map(corp, content_transformer(removeURL))
corp <- tm_map(corp, content_transformer(removeamp))
corp <- tm_map(corp, removePunctuation)
corp <- tm_map(corp, content_transformer(tolower))
corp <- tm_map(corp, removeWords, stopwords("english"))
corp <- tm_map(corp, stemDocument)
corp <- tm_map(corp, stripWhitespace)
```

c. Contruct a document-term matrix.

```
DTM <- DocumentTermMatrix(corp)
DTM <- removeSparseTerms(DTM,.99)
DTM.mat <- as.matrix(DTM)
```

d. Tidy the term matrix or otherwise standardize it for analysis.

```
tfidf <- tidy(DTM)
```

e. Create a tf-idf matrix.

```
total_words <- tfidf %>%
group_by(document) %>%
summarize(total = sum(count))
tfidf <- left_join(tfidf, total_words)
```

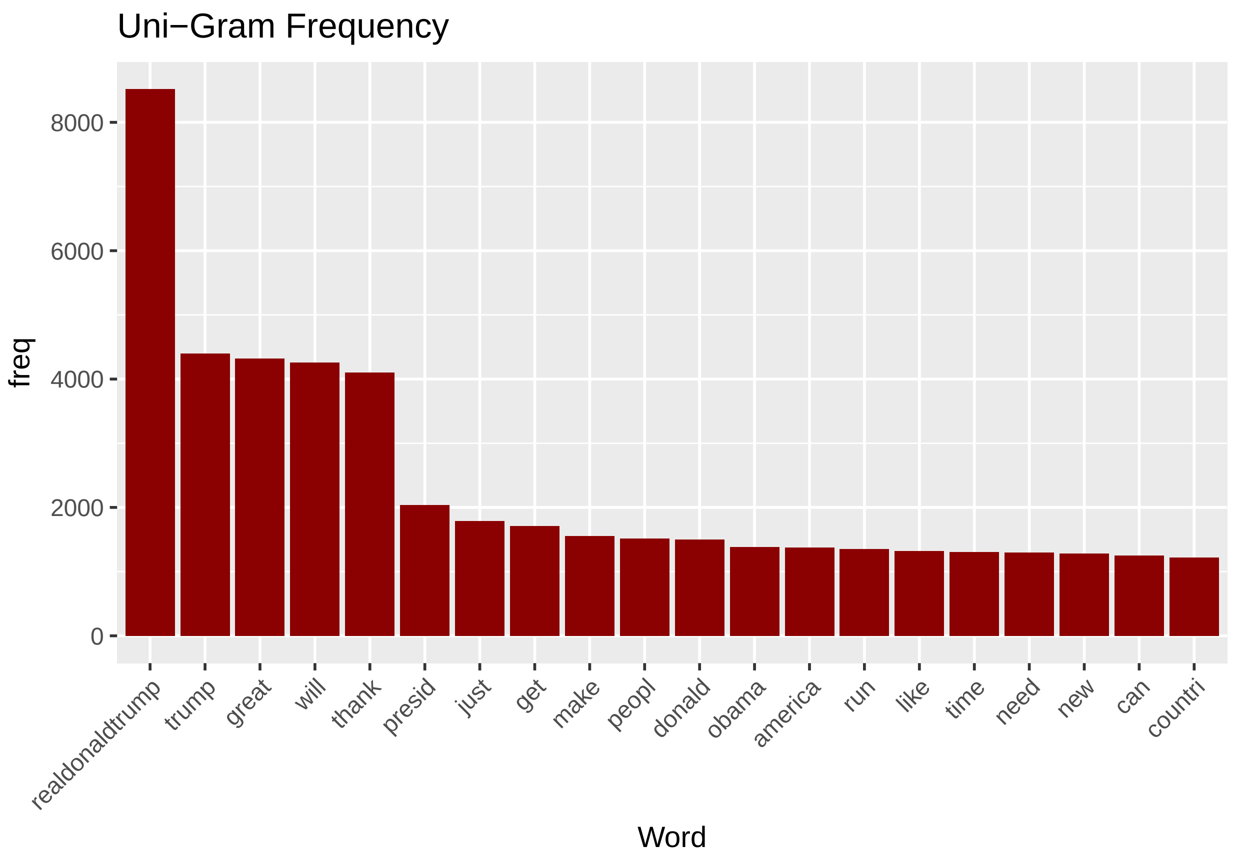
```
## Joining, by = "document"
```

```
tfidf <- tfidf %>% bind_tf_idf(document, term, count)
```

2.2 Word Frequency/Dictionary Methods

a. Plot the 20 most commonly occurring terms across the tweets.

```
freq2 <- sort(colSums(DTM.mat), decreasing=TRUE)
top20 <- head(freq2, 20)
wof <- data.frame(word=names(top20), freq=top20)
pl <- ggplot(wof, aes(reorder(word,-freq), freq))
pl +  geom_bar(stat="identity", fill="darkred") + theme(axis.text.x=element_text(angle=45, hjust=1)) +
```



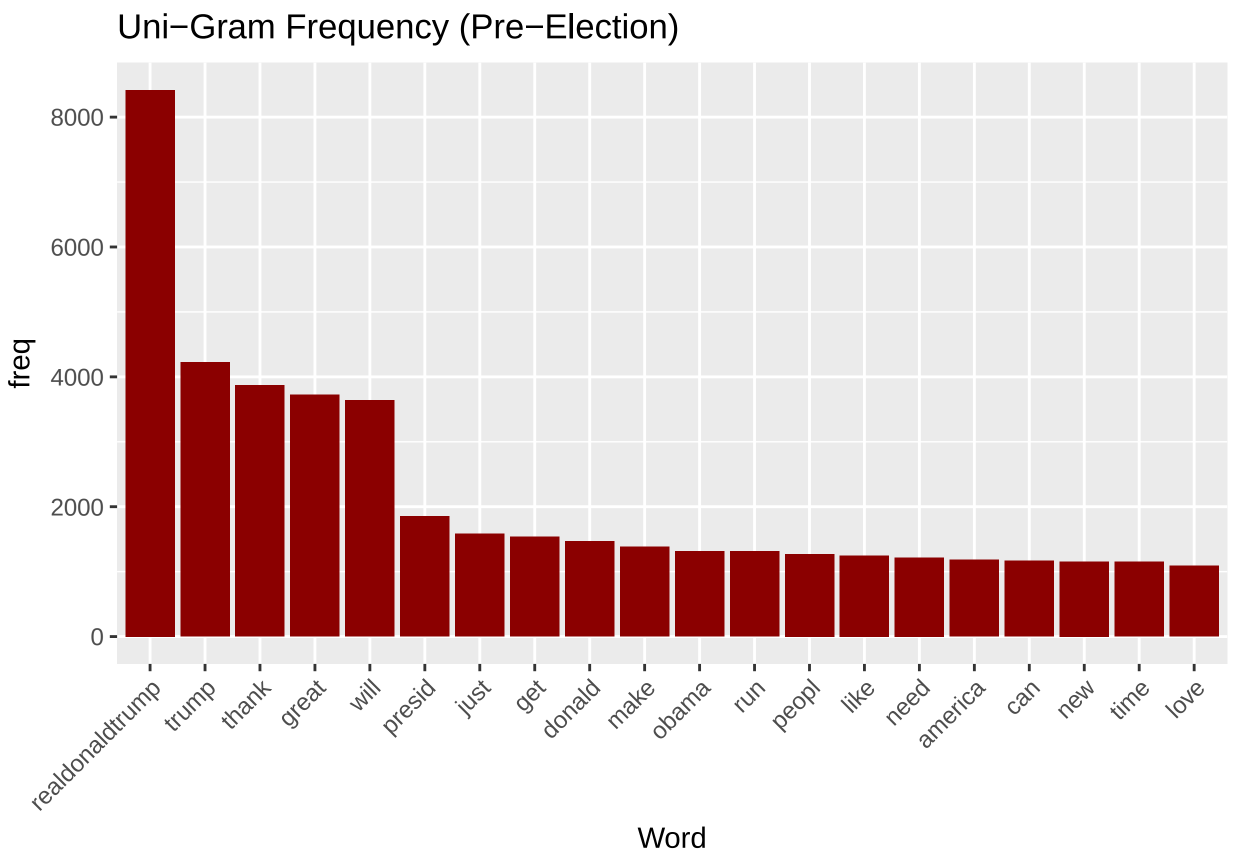
b. Split the data into pre/post-election sets. Now re-analyze and plot the 20 most common terms for each set. How do they differ?

First, we split the data into pre and post election.

```
# Split data
preelection <- meta(corp, "time") < "2016-11-08 00:00:00"
postelection <- meta(corp, "time") >= "2016-11-08 00:00:00"

precorp <- corp[preelection]
postcorp <- corp[postelection]

# Make sure that the data is in plain text
precorp <- tm_map(precorp, PlainTextDocument)
postcorp <- tm_map(postcorp, PlainTextDocument)
```

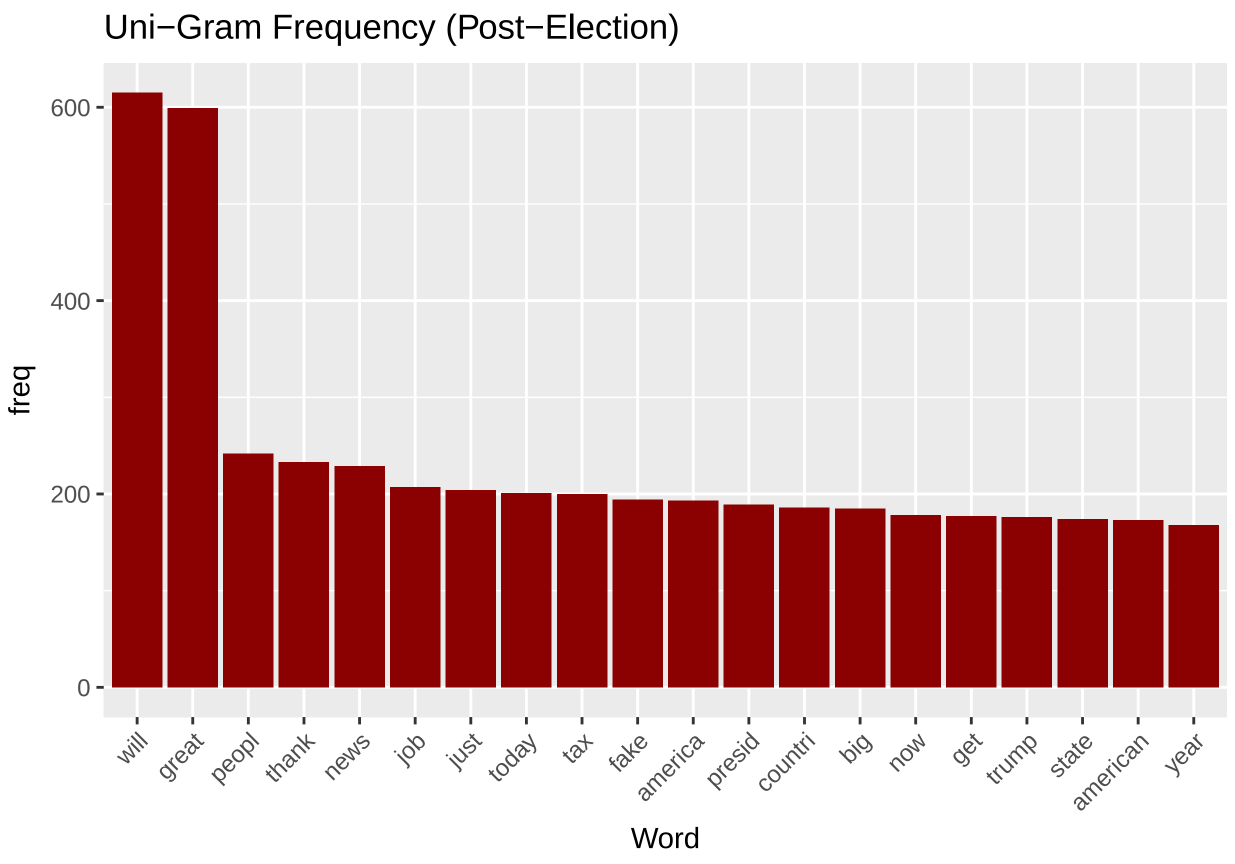
Analysis for pre-election

```r
# Doc Matrix for pre-election
DTM <- DocumentTermMatrix(precorp)
DTM <- removeSparseTerms(DTM,.99)
DTM.mat <- as.matrix(DTM)

# Plot
freq2 <- sort(colSums(DTM.mat), decreasing=TRUE)
top20 <- head(freq2, 20)

wof <- data.frame(word=names(top20), freq=top20)
pl <- ggplot(wof, aes(reorder(word,-freq), freq))
pl +  geom_bar(stat="identity", fill="darkred") + theme(axis.text.x=element_text(angle=45, hjust=1)) + g
```



Analysis for post-election

```r
# Doc Matrix for post-election
DTM <- DocumentTermMatrix(postcorp)
DTM <- removeSparseTerms(DTM,.99)
DTM.mat <- as.matrix(DTM)

# Plot
freq2 <- sort(colSums(DTM.mat), decreasing=TRUE)
top20 <- head(freq2, 20)

wof <- data.frame(word=names(top20), freq=top20)
pl <- ggplot(wof, aes(reorder(word,-freq), freq))
pl +  geom_bar(stat="identity", fill="darkred") + theme(axis.text.x=element_text(angle=45, hjust=1)) + g
```



Pre and post election sets differ in quite significant ways. First, since post-election set is significantly smaller than pre-election set, the most frequent term for post-election ("will") occurs only 615 times, whereas the most frequent term for pre-election ("realdonaldtrump") occurs 8417 times. Interestingly, prior to the election, "obama" occurs quite frequently, yet this term disappeared from the top 20 most frequent terms after the election.

c. Suppose now that you'd like to assess the frequency with which Trump uses specific hashtags. Notice that the # that signals a hashtag was removed in your preprocessing step that eliminated punctuation. Regret this immensely. Pre-process the data again to preserve only # and eliminate other punctuation (.,; etc.).

```
# https://pushpullfork.com/mining-twitter-data-tidy-text-tags/
reg_words <- "([^A-Za-z_\\d#@']|'(?![A-Za-z_\\d#@]))"
tidy_trump <- trumptweets %>%
  filter(!str_detect(text, "^RT")) %>%
  mutate(text = str_replace_all(text, "https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|&lt;|&gt;|RT
  unnest_tokens(word, text, token = "regex", pattern = reg_words) %>%
  filter(!word %in% stop_words$word,
         str_detect(word, "[a-z]"))
```

d. With your differently pre-processed DTM, evaluate the frequency only of hashtags Trump has used: what are the top 5 most-used over the entire time period?

```
org <- filter(tidy_trump, is_retweet == "false")
orgtweet <- org %>% count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n))

orgtweethashonly <- orgtweet %>% filter(grepl("^#", word))
top5 <- head(orgtweethashonly, 5)
```

```
top5
```

```
## # A tibble: 5 x 2
##   word                        n
##   <fct>                   <int>
## 1 #trump2016                861
## 2 #makeamericagreatagain    521
## 3 #celebapprentice          297
## 4 #celebrityapprentice      152
## 5 #maga                     128
```

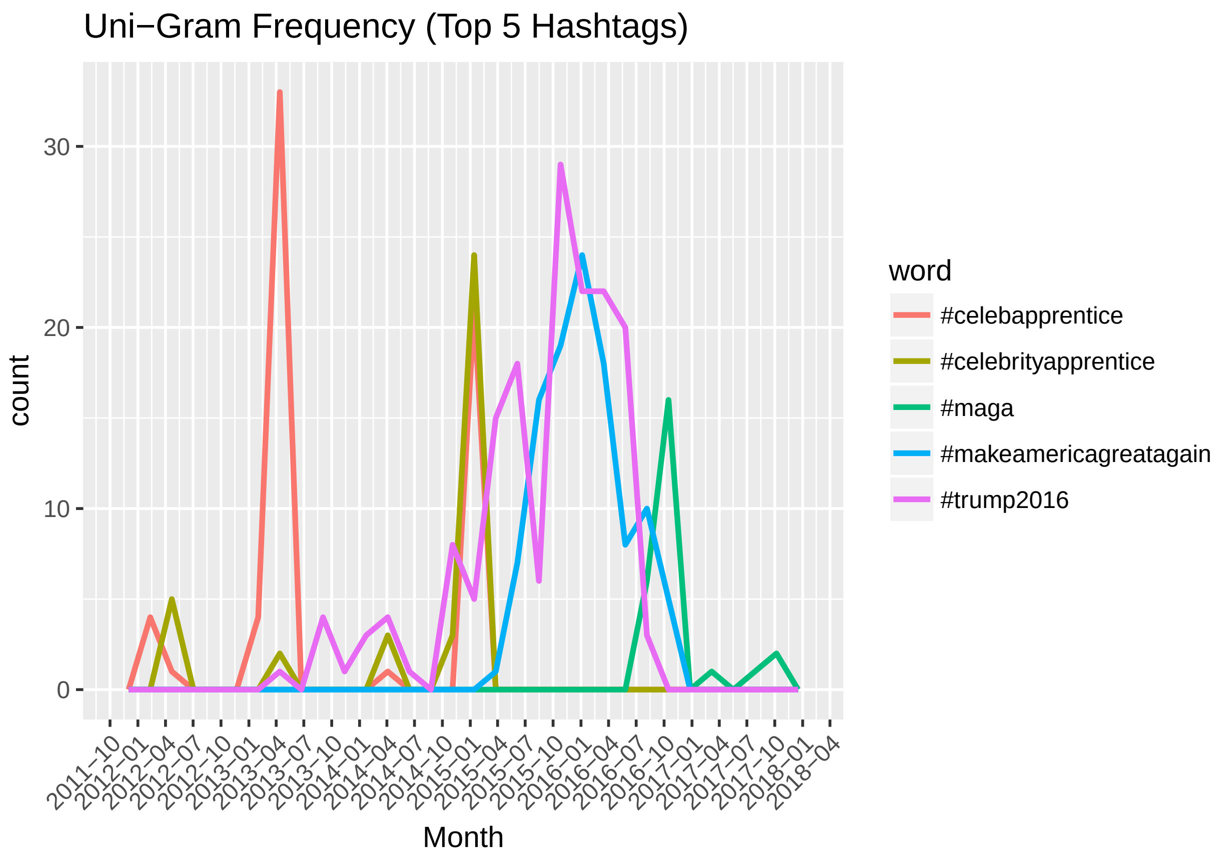
e. Plot the frequency of these top 5 hashtags over time using ggplot2.

```
tophash <- c("#trump2016", "#makeamericagreatagain", "#celebapprentice", "#celebrityapprentice", "#maga
tophashdat <- filter(org, word==tophash)
```

```
## Warning in word == tophash: longer object length is not a multiple of
## shorter object length
```

```
tophashdat %>%  ggplot(aes(x = as.Date(time), color = word)) + geom_freqpoly(size = 1, stat = "bin",
  position = "identity", show.legend = TRUE) + scale_x_date(date_breaks = "3 month", date_labels = "%Y-
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



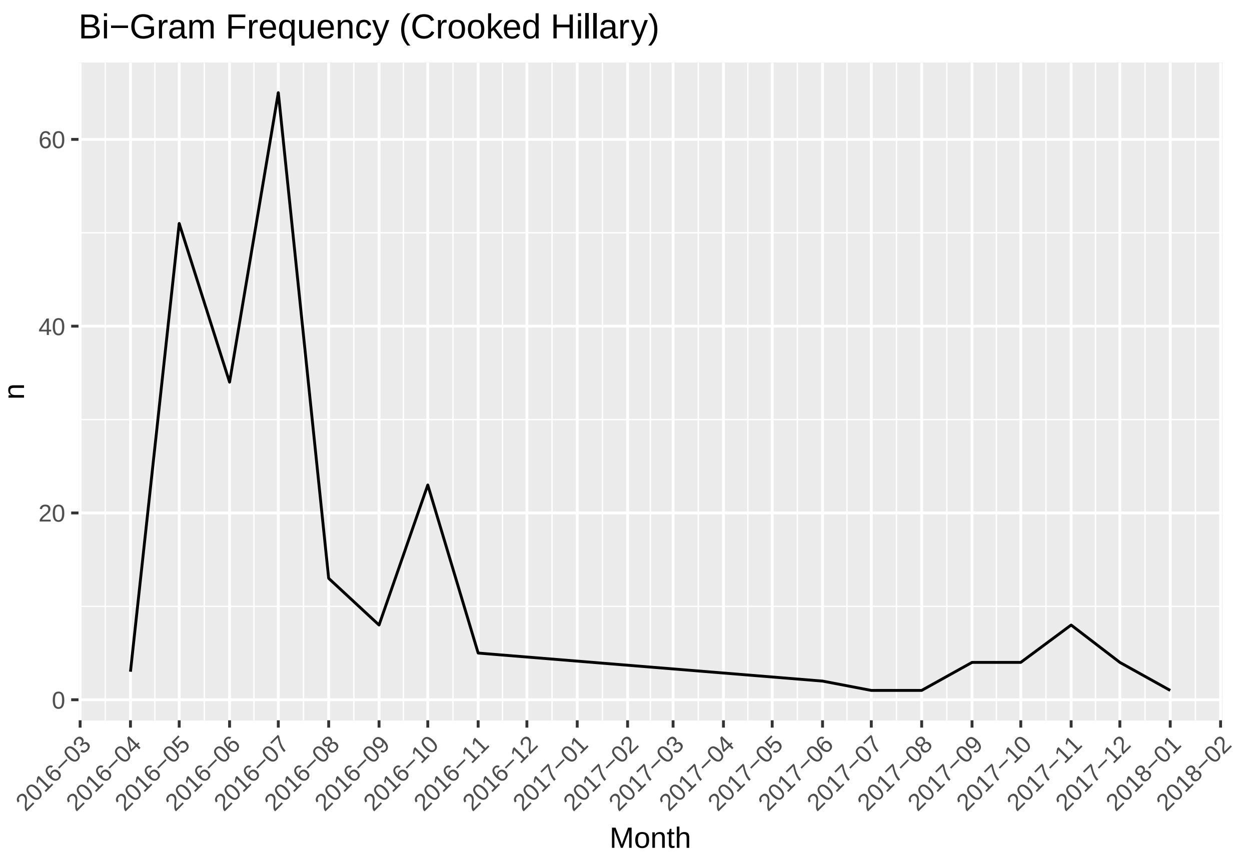
f. Using bigrams rather than unigrams, report the frequency with which Trump used the phrase "Crooked Hillary" over time (by month).

```
bigram.docs <- trumptweets %>%
unnest_tokens(ngram, text, token = "ngrams", n = 2) %>% count(month=floor_date(time, "month"), ngram, s
ungroup()
```

```r
crookedhillary <- filter(bigram.docs, ngram == "crooked hillary")
crookedhillary %>% arrange(ngram, month)
```

```
## # A tibble: 16 x 3
##    month               ngram               n
##    <dttm>              <chr>           <int>
##  1 2016-04-01 00:00:00 crooked hillary     3
##  2 2016-05-01 00:00:00 crooked hillary    51
##  3 2016-06-01 00:00:00 crooked hillary    34
##  4 2016-07-01 00:00:00 crooked hillary    65
##  5 2016-08-01 00:00:00 crooked hillary    13
##  6 2016-09-01 00:00:00 crooked hillary     8
##  7 2016-10-01 00:00:00 crooked hillary    23
##  8 2016-11-01 00:00:00 crooked hillary     5
##  9 2017-06-01 00:00:00 crooked hillary     2
## 10 2017-07-01 00:00:00 crooked hillary     1
## 11 2017-08-01 00:00:00 crooked hillary     1
## 12 2017-09-01 00:00:00 crooked hillary     4
## 13 2017-10-01 00:00:00 crooked hillary     4
## 14 2017-11-01 00:00:00 crooked hillary     8
## 15 2017-12-01 00:00:00 crooked hillary     4
## 16 2018-01-01 00:00:00 crooked hillary     1
```

```r
crookedhillary %>%  ggplot(aes(as.Date(month), n)) + geom_line() + scale_x_date(date_breaks = "1 month"
```

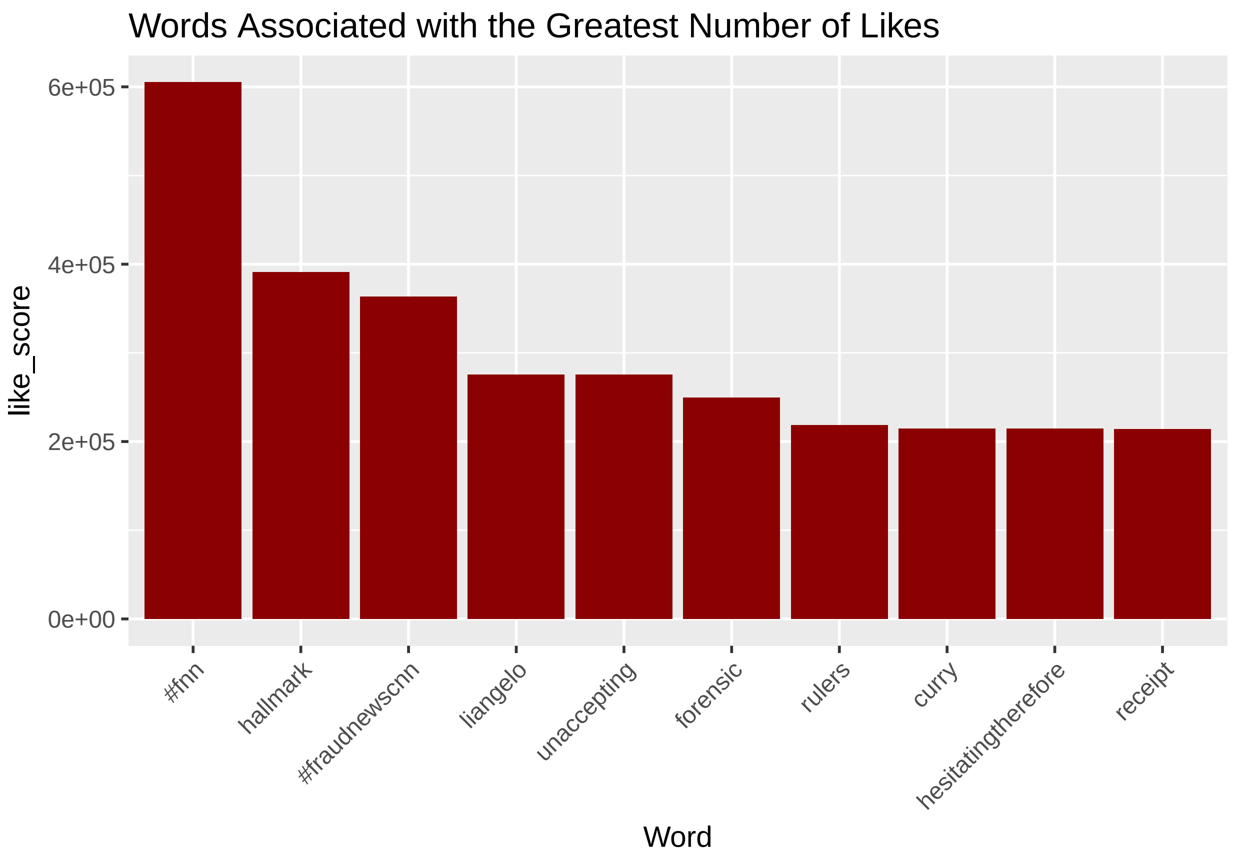


g. Suppose I want to know if the words associated with the greatest number of “likes” of a tweet are different from the words associated with the greatest number of retweets.

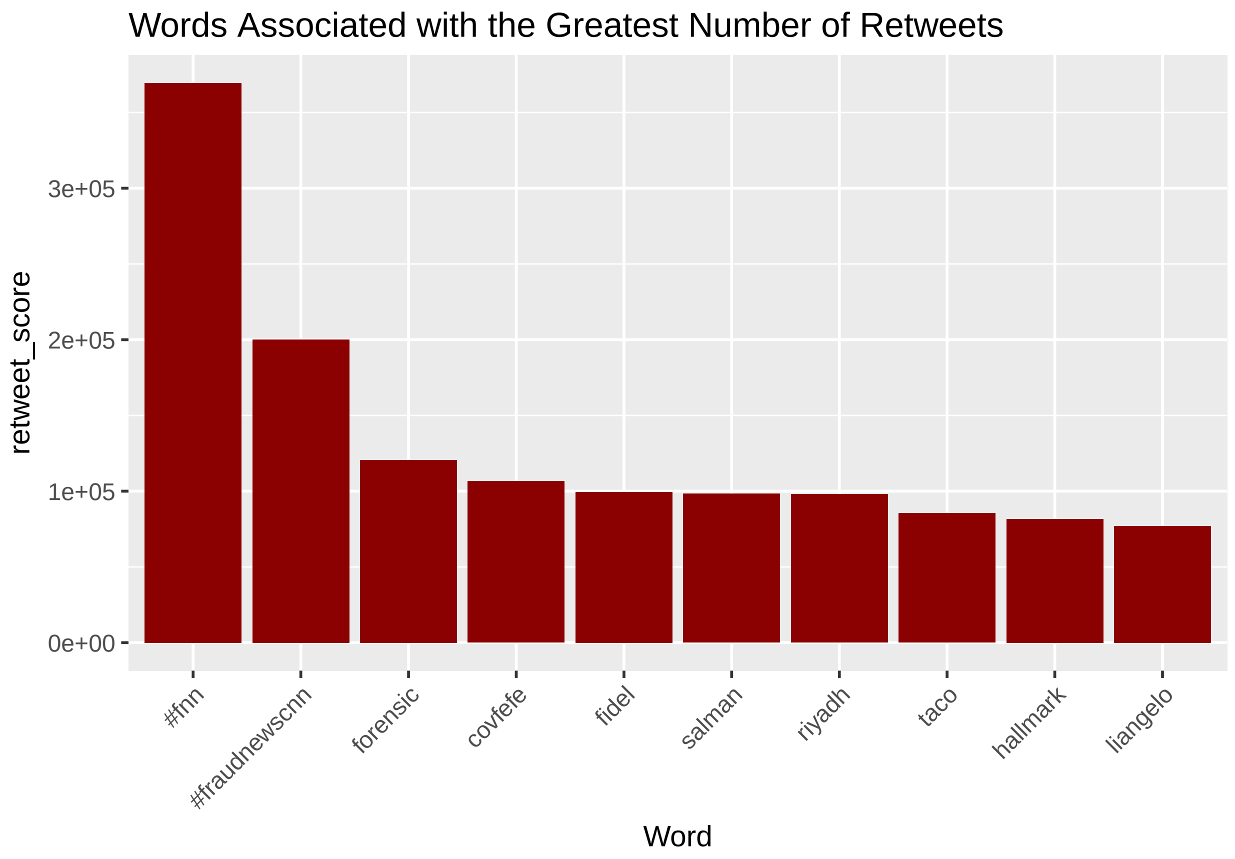
```r
# https://pushpullfork.com/mining-twitter-data-tidy-text-tags/
reg_words <- "([^A-Za-z_\\d#@']|'(?![A-Za-z_\\d#@]))"
tidy_trump <- trumptweets %>%
  mutate(text = str_replace_all(text, "https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|&lt;|&gt;|RT
  unnest_tokens(word, text, token = "regex", pattern = reg_words) %>%
  filter(!word %in% stop_words$word,
         str_detect(word, "[a-z]"))

freq  <- tidy_trump %>% count(word, sort = TRUE) %>% ungroup()

retweets <- tidy_trump %>% group_by(word) %>% summarize(retweet = sum(retweet_count)) %>% ungroup()
likes <- tidy_trump %>% group_by(word) %>% summarize(likes = sum(favorite_count)) %>% ungroup()

freq3 <- left_join(left_join(freq, likes), retweets)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

```r
freq4 <- freq3 %>% mutate(like_score = likes/n, retweet_score = retweet/n)

# Get words with greatest number of likes
toplikes <- head(arrange(freq4, desc(like_score)), 10)

pl <- ggplot(toplikes, aes(reorder(word, -like_score), like_score))
pl +  geom_bar(stat="identity", fill="darkred") + theme(axis.text.x=element_text(angle=45, hjust=1)) +
```



```
# Get words with greatest number of retweets
toptweets <- head(arrange(freq4, desc(retweet_score)), 10)
pl <- ggplot(toptweets, aes(reorder(word, -retweet_score), retweet_score))
pl +  geom_bar(stat="identity", fill="darkred") + theme(axis.text.x=element_text(angle=45, hjust=1)) +
```



The top 10 words associated with the greatest number of likes and the top 10 words associated with the greatest number of retweets are shown above. There are a few words which the two lists share in common (#fnn, hallmark, #fraudnewscnn, forensic, liangelo). Unfortunaely, I was not able to find a way to deal with words that appear more than once in a single tweet. In other words, these lists would be biased towards those words which appear more than once per tweet.