

Text Mining in R

Jarrod Griffin

1/14/2021

Importing Data

Tweets from the CCIDM Twitter page (<https://twitter.com/CCIDM>) were downloaded using the `snsraper` python package. Data was in the 'json' format, so we need to use the 'rjson' package to import it. All hashtags and mentions were removed manually. I will also generate IDs for each tweet.

```
#install.packages("rjson")
#install.packages('tidyverse')
#install.packages('tidytext')
library('rjson')
library('tidyverse')
library('tidytext')

tweets <- fromJSON(file = 'CCIDM_tweets.json') %>%
  as_tibble() %>%      #tidytext package uses tibbles
  cbind(tweet_id = 31:1) #generate IDs (tweets are in reverse chronological order)

head(tweets)
```

```
##
## 1           We would like to thank CCIDM alumnus @WilliamAtienza2 for joining Center members
## 2 We would like to thank everyone who participated in our workshops, attended our events, and helped
## 3
## 4
## 5           Our full, six part Virtual R Workshop Series
## 6           Join the CCIDM, AMI, and Insights Association on November 11th
## 6           Thank you to those who attended!

##  tweet_id
## 1      31
## 2      30
## 3      29
## 4      28
## 5      27
## 6      26
```

Tidy Text Format and Tokenization

The tidy text format takes after Hadley Wickham's definition of tidy data, which is that:

- Each variable is a column
- Each observation is a row

- Each type of observational unit is a table

Tidy text is defined as a **table with one token per row**.

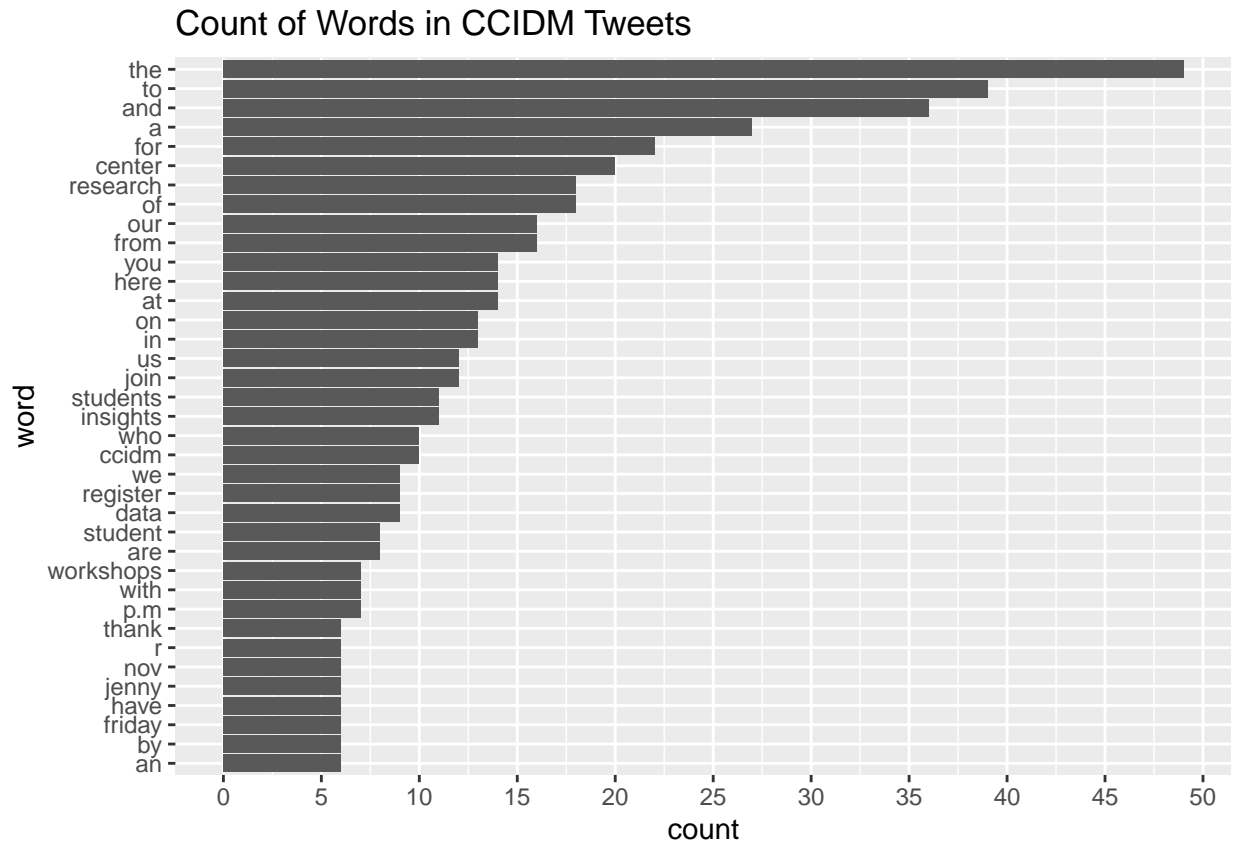
A token is defined as a **meaningful unit of text such as a word, sentence, paragraph or n-gram**.

The process of splitting the text up into tokens is called **tokenization**, and can be done by using the `unnest_tokens()` function.

```
tokenized_tweets <- unnest_tokens(tweets, input = 'tweet', output = 'word')
head(tokenized_tweets)
```

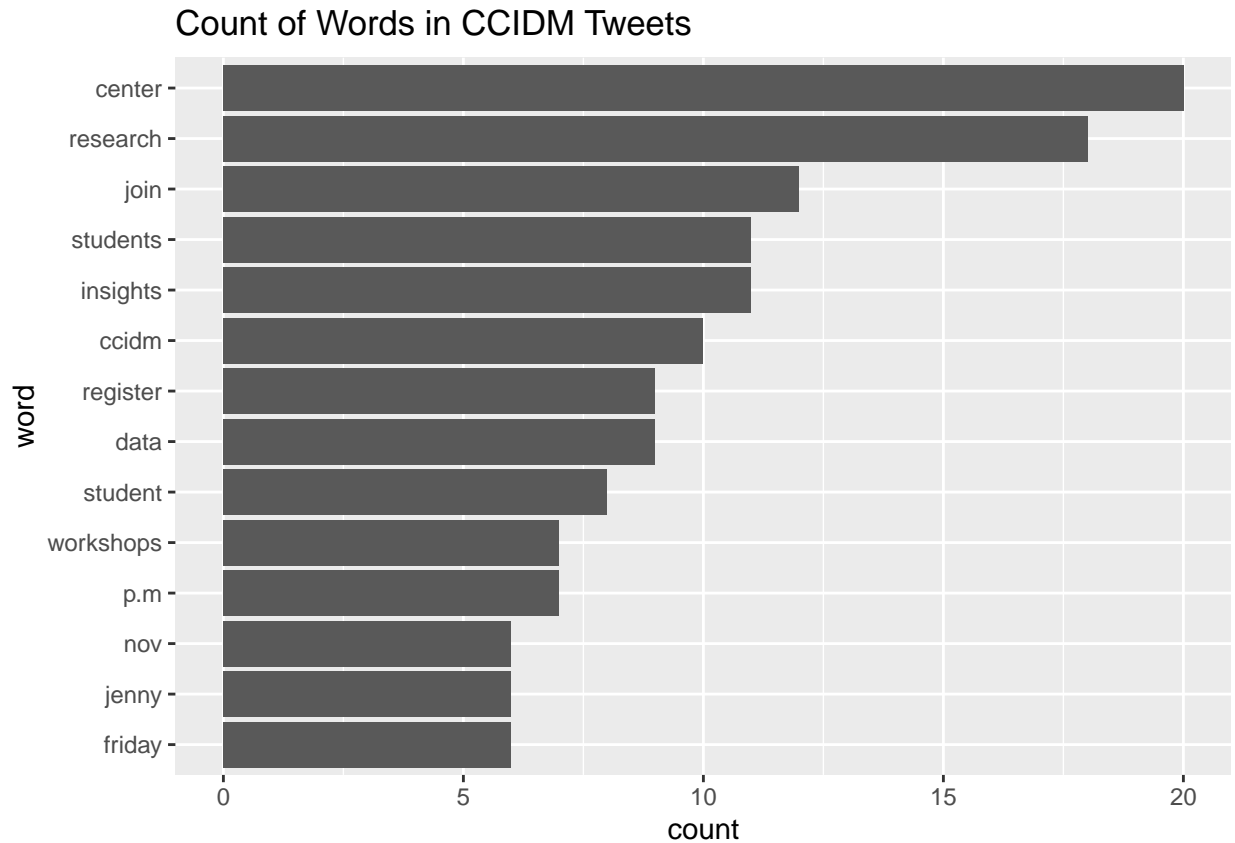
```
##      tweet_id word
## 1          31  we
## 1.1        31 would
## 1.2        31  like
## 1.3        31   to
## 1.4        31 thank
## 1.5        31 ccidm
```

```
tokenized_tweets %>%
  count(word, sort = TRUE) %>%
  rename(count = n) %>%
  filter(count > 5) %>%
  mutate(word = reorder(word, count)) %>%
  ggplot(aes(x = count, y = word)) +
    geom_col() +
    labs(title = "Count of Words in CCIDM Tweets") +
    scale_x_continuous(breaks = seq(0, 50, 5))
```



As you can see from the graph above, many of the words do not add value to our analysis. Words like “the”, “and”, or “to” are known as **stop words**. We will remove these stop words by calling the `anti_join(stop_words)` line of code. As you can see from the graph below, we have less words, but the words are much more interesting.

```
tokenized_tweets %>%
  anti_join(stop_words) %>% #finds where tweet words overlap with predefined stop words, and removes th
  count(word, sort = TRUE) %>%
  rename(count = n) %>%
  filter(count > 5) %>%
  mutate(word = reorder(word, count)) %>%
  ggplot(aes(x = count, y = word)) +
    geom_col() +
    labs(title = "Count of Words in CCIDM Tweets") +
    scale_x_continuous(breaks = seq(0, 50, 5))
```



There are many ways to visualize word counts, including word clouds as seen below.

```
#install.packages('ggwordcloud')
library('ggwordcloud')

tokenized_tweets %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(n > 4) %>%
  ggplot(aes(label = word, size = n, color = n)) +
    geom_text_wordcloud() +
    scale_size_area(max_size = 15)
```

