Sentiment Analysis in R

Jenn Schilling

10/21/2021

Introduction

This file contains the code for webinar: Sentiment Analysis in R (https://www.airweb.org/collaborate-learn/calendar/2021/10/20/event/sentiment-analysis-in-r), presented for the Association for Institutional Research, October 20 & 22 2021.

Webinar Details
This webinar will teach participants how to complete a text analysis in R, including data processing and cleaning, visualization of word frequencies, and sentiment analysis. Sentiment analysis is useful for finding patterns in text data from open response questions on surveys and course evaluations as well as evaluating social media posts. This series is ideal for higher education professionals who have some experience in R and want to add text analysis to their R skills.

As a result of this webinar, participants will be able to: - Prepare data for a text analysis in R. - Conduct text mining in R. - Complete sentiment analysis in R.

Materials developed by Jenn Schilling.

Setup

Load libraries and tweet data.

```
library(here) # working directory
library(tidyverse) # data processing and plotting
library(tidytext) # text analysis
library(wordcloud) # word cloud
library(scales) # number formatting

# Read data
text_data <- read_csv(here("data", "uarizona_tweets.csv"), show_col_types = FALSE)
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

Data Processing

The first step is to understand and process the data. This particular dataset is a subset of what is pulled from the `rtweet` package. The full data includes more details about the tweet and engagement with it, but this subset includes the date, user, tweet text, source, and a few other metrics that may be of interest.

Now maybe you do not want to look at tweets, the same process we are going to walk through in this webinar could be used for any type of text data. I pulled Twitter data to use publicly accessible data that is relevant to my institution, using hashtags that are related to my university, so that I would have a good demonstration dataset. But this same process would work with any text dataset, you would just adjust the code to read the data in the “Setup” chunk above.

One important part of text analysis is having an identification column. Since we will eventually be creating a data frame of individual words, we want to be able to tie those words back to the original text (in this case, tweet). Having an identification column allows us to tie back to the original text. In this case, we will use the `status_id` field to identify each tweet.

Being able to tie back the analyzed data to the original text can be useful to complete the analysis. For example, if you were analyzing course evaluations and you wanted to evaluate the sentiment of students by academic year in a course, or if you were analyzing an open-ended survey question and wanted to investigate the sentiment of students in different departments, you could use the identification column to join your final analyzed data back to the original text data and determine these segments. It is not in this particular dataset, but if we had the location data of the tweets, then we could look at the words used or sentiment of the tweets by zip code to get an idea of the impression of the university in different locations.

```
# First let's look at the data
View(text_data)

# Check the number of rows and columns
dim(text_data)
```

```
## [1] 577   9
```

```
# View the data types of the columns
glimpse(text_data)
```

```
## Rows: 577
## Columns: 9
## $ status_id      <dbl> 1.440821e+18, 1.440818e+18, 1.440798e+18, 1.440809e+18,~
## $ created_at     <dttm> 2021-09-22 23:30:14, 2021-09-22 23:18:39, 2021-09-22 2~
## $ screen_name    <chr> "MeggsHahn", "NortonTutors", "NortonTutors", "NortonTut~
## $ text           <chr> "i don’t care what frat you’re in two men on a vespa is~
## $ source         <chr> "Twitter for iPhone", "Norton Tutors", "Norton Tutors",~
## $ is_quote       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ is_retweet     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ favorite_count <dbl> 0, 0, 0, 0, 0, 0, 0, 6, 1, 0, 38, 0, 0, 3, 0, 0, 3, 2, ~
## $ retweet_count  <dbl> 0, 1, 2, 0, 2, 1, 0, 0, 0, 0, 2, 0, 1, 1, 0, 2, 1, 0, 0~
```

Once we have a basic understanding of the data, we need to process it. We will first make all the text lowercase. Then we need to expand contractions, remove special characters and emojis, and remove extra whitespace. We also want to make the `status_id` column a character column instead of a numeric column.

```
text_data_processed <- text_data %>%

# Lowercase
  mutate(text = str_to_lower(text)) %>%

# Expand contractions
  mutate(text = gsub("n't|n’t", " not", text),
         text = gsub("'ll|’ll", " will", text),
         text = gsub("'re|’re", " are", text),
         text = gsub("'ve|’ve", " have", text),
         text = gsub("'m|’m", " am", text),
         text = gsub("'d|’d", " would", text),
         text = gsub("it's|it’s", "it is", text),
         text = gsub("'s|’s", "", text)) %>%

# Remove emojis
  mutate(text = gsub("\U0001", "", text)) %>%

# Remove links
  mutate(text = gsub("(https:|http:).*", "", text)) %>%

# Remove special characters
  mutate(text =  gsub("[^a-zA-Z0-9 ]", " ", text)) %>%

# Remove ampersand notation
  mutate(text = gsub("amp", "", text)) %>%

# Remove extra whitespace
  mutate(text = str_squish(text)) %>%

# Make identification column a character
  mutate(status_id = as.character(status_id))
```

Now that the text processing is complete, let’s take another look at the data.

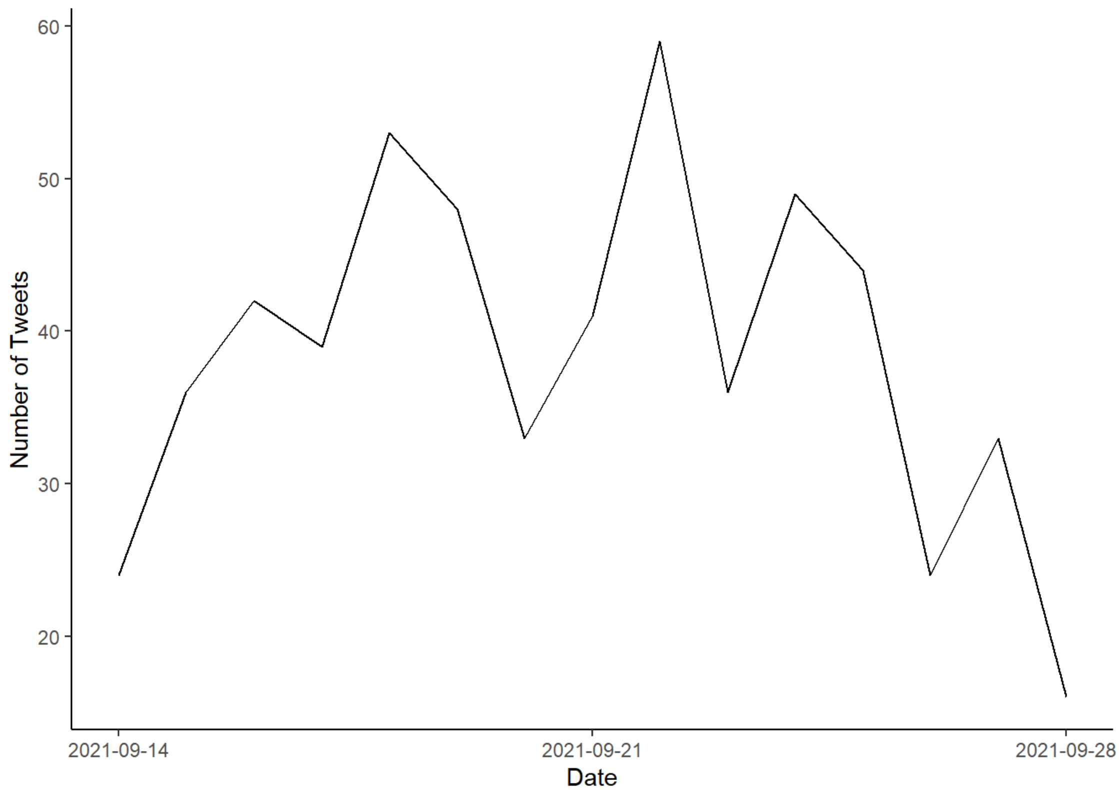
```
# View the columns and a few records
glimpse(text_data_processed)
```

```
## Rows: 577
## Columns: 9
## $ status_id      <chr> "1440820766086340608", "1440817851074420736", "14407984~
## $ created_at     <dttm> 2021-09-22 23:30:14, 2021-09-22 23:18:39, 2021-09-22 2~
## $ screen_name    <chr> "MeggsHahn", "NortonTutors", "NortonTutors", "NortonTut~
## $ text           <chr> "i do not care what frat you are in two men on a vespa ~
## $ source         <chr> "Twitter for iPhone", "Norton Tutors", "Norton Tutors",~
## $ is_quote       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ is_retweet     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ favorite_count <dbl> 0, 0, 0, 0, 0, 0, 0, 6, 1, 0, 38, 0, 0, 3, 0, 0, 3, 2, ~
## $ retweet_count  <dbl> 0, 1, 2, 0, 2, 1, 0, 0, 0, 0, 2, 0, 1, 1, 0, 2, 1, 0, 0~
```

```
# Plot number of tweets over time
tweets_time <- text_data_processed %>%
  mutate(created_at_date = as.Date(created_at, "%m/%d/%Y")) %>%
  group_by(created_at_date) %>%
  summarise(n = n(),
            .groups = "drop")
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%m/%d/%Y'
```

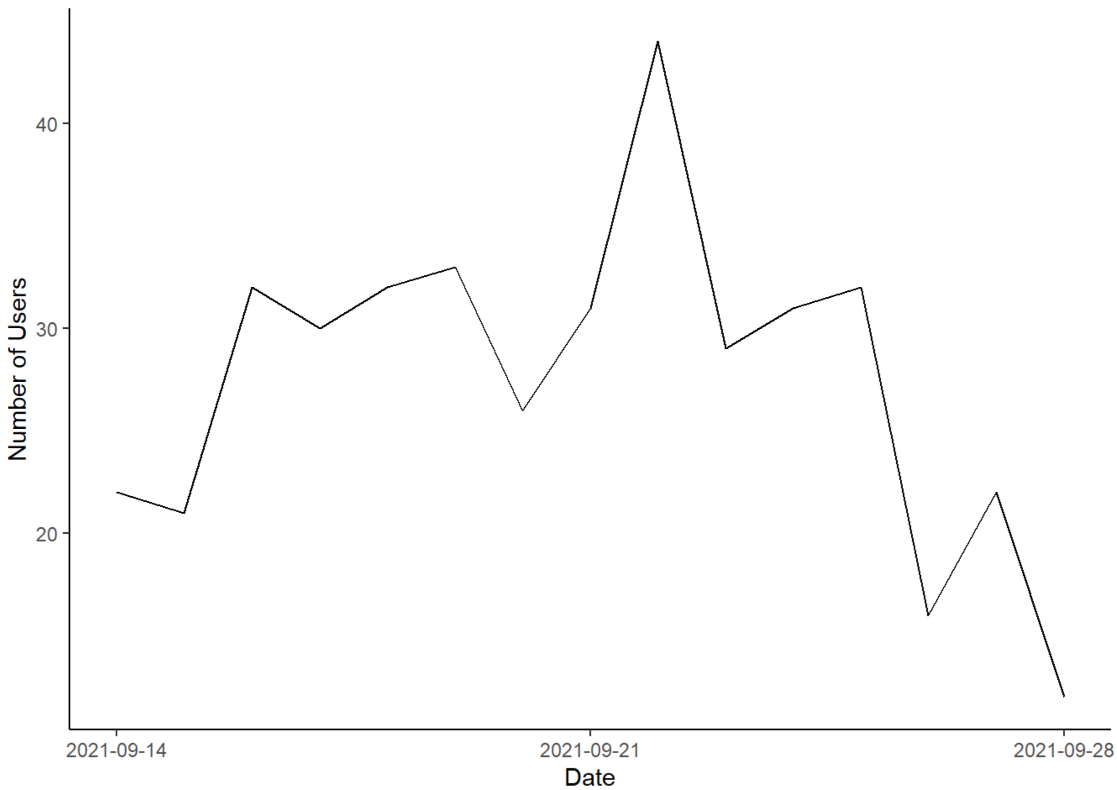
```
ggplot(data = tweets_time,
       mapping = aes(x = created_at_date,
                     y = n,
                     group = 1)) +
  geom_line() +
  scale_x_continuous(breaks = seq(min(tweets_time$created_at_date),
                                  max(tweets_time$created_at_date),
                                  "weeks")) +
  labs(x = "Date",
       y = "Number of Tweets") +
  theme_classic()
```



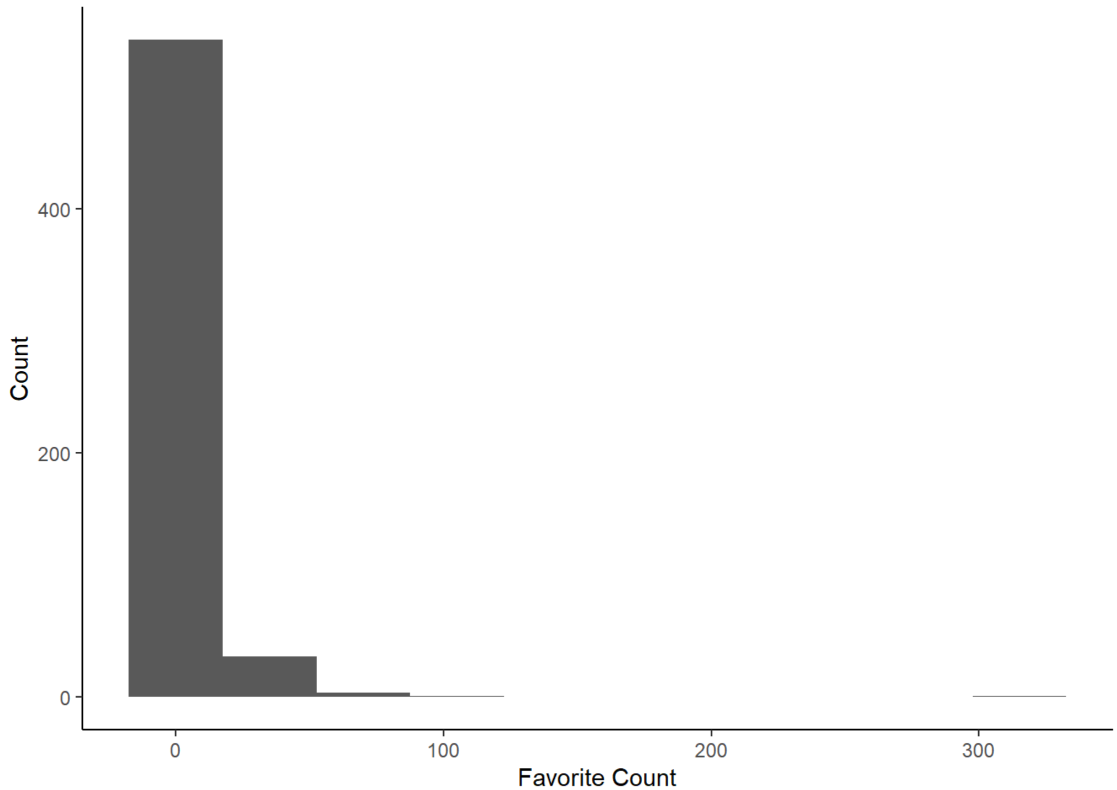

```
# Plot number of screen names over time
tweets_user <- text_data_processed %>%
  mutate(created_at_date = as.Date(created_at, "%m/%d/%Y")) %>%
  select(created_at_date, screen_name) %>%
  unique(.) %>%
  group_by(created_at_date) %>%
  summarise(n = n(),
            .groups = "drop")
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%m/%d/%Y'
```

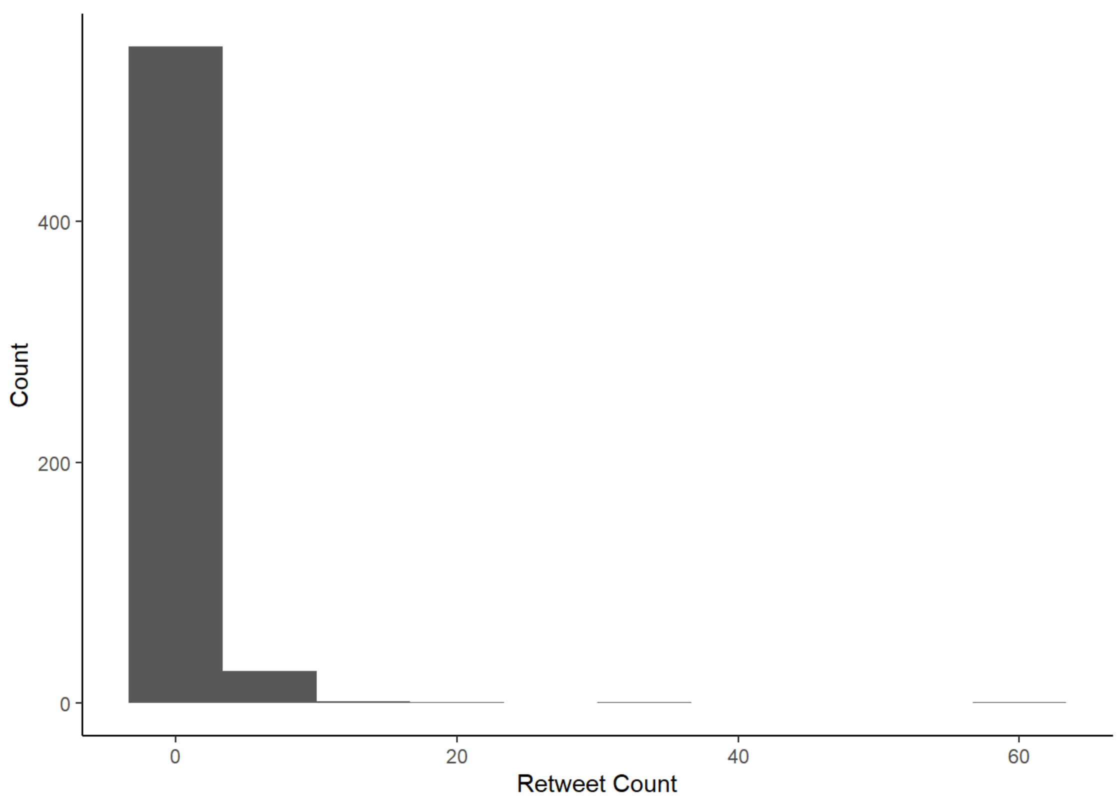
```
ggplot(data = tweets_user,
       mapping = aes(x = created_at_date,
                     y = n,
                     group = 1)) +
  geom_line() +
  scale_x_continuous(breaks = seq(min(tweets_user$created_at_date),
                                  max(tweets_user$created_at_date),
                                  "weeks")) +
  labs(x = "Date",
       y = "Number of Users") +
  theme_classic()
```




```
# Plot histograms of favorite count and retweet count
ggplot(data = text_data_processed,
       mapping = aes(x = favorite_count)) +
  geom_histogram(bins = 10) +
  labs(x = "Favorite Count",
       y = "Count") +
  theme_classic()
```




```
ggplot(data = text_data_processed,
       mapping = aes(x = retweet_count)) +
  geom_histogram(bins = 10) +
  labs(x = "Retweet Count",
       y = "Count") +
  theme_classic()
```



Tokenizing

After processing the data, the next step in a text analysis is to get individual words. This is called tokenizing. Tokenizing involves splitting a long text string, such as a tweet, document, or open-ended responses, into individual words or sets of words. A token is a meaningful unit of text, such as a word. For text analysis, we can create a tidy text data table or data frame which contains one token per row.

There is a simple function in the `tidytext` that will complete the tokenizing process for us called `unnest_tokens()`. This will create a new data frame with one row for each word in the text and a new column with each word.

After tokenizing, we will remove stop words, which are common words in the English language that are not useful for text analysis. These are words such as “and”, “the”, “can”, “a”, etc.

```
# Tokenize the text data to get each individual word
text_tokens <- text_data_processed %>%
  unnest_tokens(word, text)

# Let's see what the new table looks like
View(text_tokens)

dim(text_tokens)
```

```
## [1] 12450     9
```

```
# Let's take a look at the stop words list
View(stop_words)

# Now remove the stop words
text_tokens <- anti_join(text_tokens, stop_words, by = "word")

# Look at the tokenized data frame again
View(text_tokens)

dim(text_tokens) # notice the drop in the row count now that stop words are gone
```

```
## [1] 7369    9
```

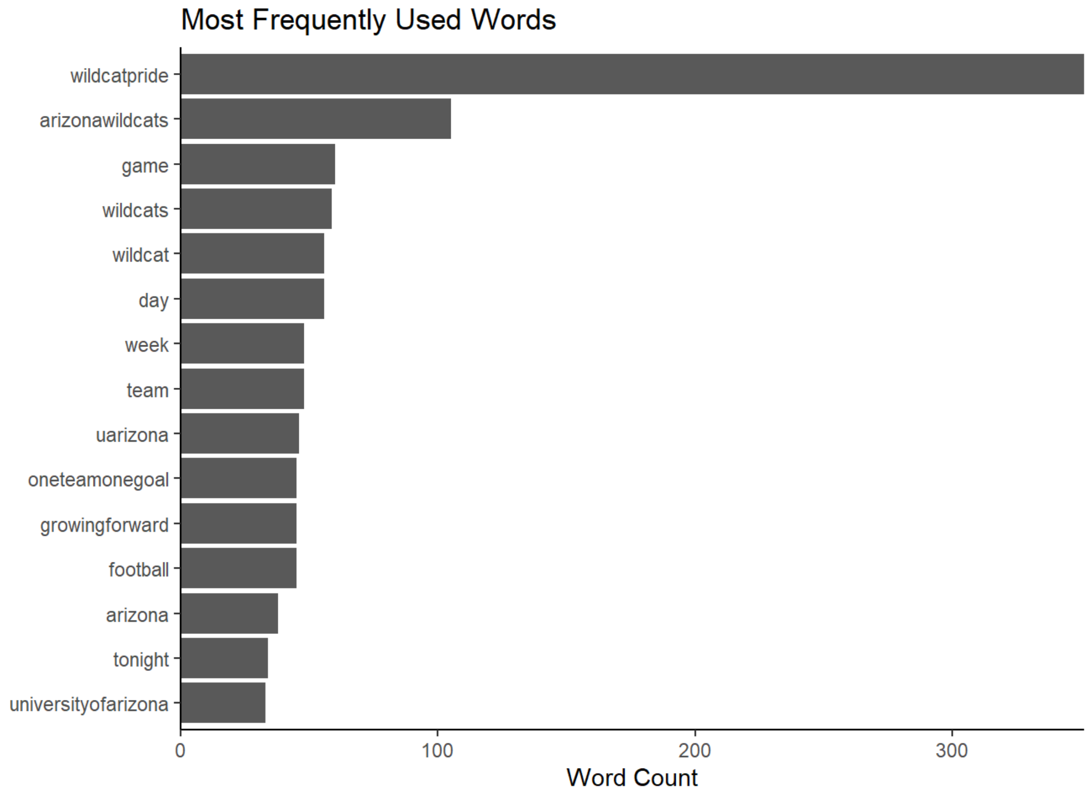
Word Frequencies

Now that we have a data frame with the individual words, we can begin to do some analysis of the text and the words used. We can examine word frequencies and look at the most used words. We can look at word frequencies over time or by user. We can also investigate the number of words in each tweet.

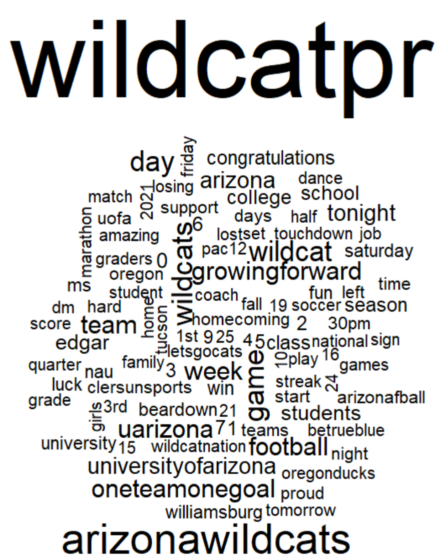
First, we are going to create a few different word count data frames. Then we will make some plots to look at the words used in the data.

```r
# Count number of times a word appears in each tweet
tweet_words <- text_tokens %>%
  count(status_id, word, sort = TRUE)

View(tweet_words)

# Count the number of words in each tweet
total_words <- tweet_words %>%
  group_by(status_id) %>%
  summarise(total_words = sum(n))

View(total_words)

# Put the counts together
total_tweet_words <- left_join(tweet_words, total_words, by = "status_id")

View(total_tweet_words)

# Count word frequencies overall
tweet_word_freq <- text_tokens %>%
  count(word, sort = TRUE)

View(tweet_word_freq)

# View top words
ggplot(data = tweet_word_freq %>%
         top_n(15),
       mapping = aes(y = reorder(word, n),
                     x = n)) +
  geom_col() +
  labs(title = "Most Frequently Used Words",
       x = "Word Count",
       y = "") +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```

```
## Selecting by n
```




```
# Create word cloud
tweet_word_freq %>%
  with(wordcloud(word, n, max.words = 100))
```

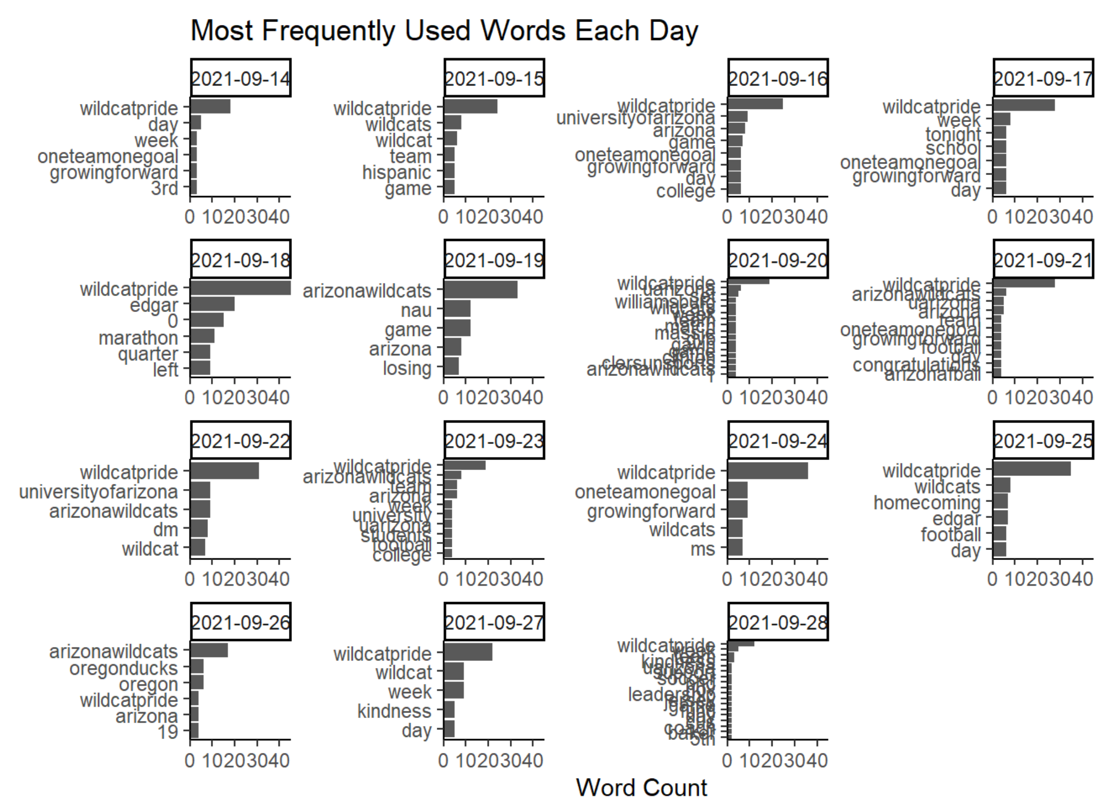


```
# Word frequency by day
day_words <- text_tokens %>%
  mutate(created_at_date = as.Date(created_at, "%m/%d/%Y")) %>%
  count(created_at_date, word, sort = TRUE)
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%m/%d/%Y'
```

```
View(day_words)

# Plot word frequencies by day
ggplot(data = day_words %>%
         group_by(created_at_date) %>%
         top_n(5) %>%
         ungroup %>%
         mutate(created_at_date = as.factor(created_at_date),
                word = reorder_within(word, n, created_at_date)),
       mapping = aes(y = word,
                     x = n)) +
  geom_col() +
  facet_wrap(~created_at_date,
             scales = "free") +
  labs(title = "Most Frequently Used Words Each Day",
       x = "Word Count",
       y = "") +
  scale_y_reordered() +
  scale_x_continuous(limits = c(0, 45),
                     expand = c(0, 0)) +
  theme_classic()
```

```
## Selecting by n
```

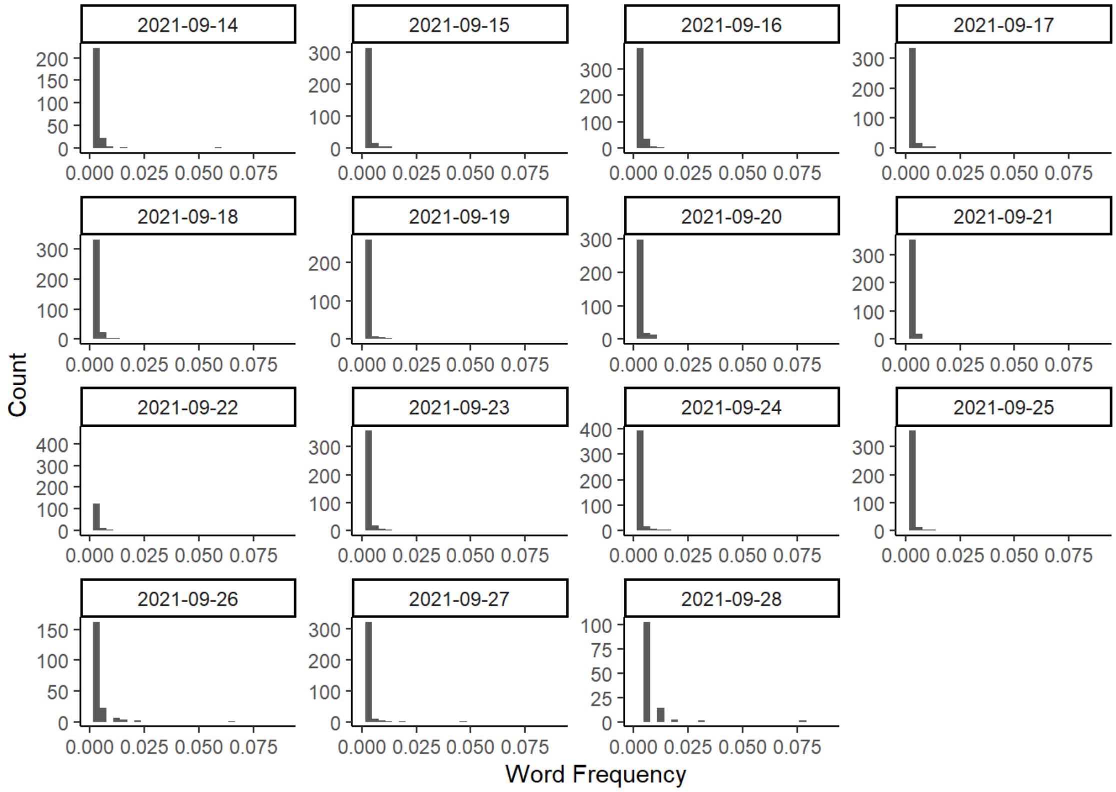


Term Frequency - Inverse Document Frequency

Another way to look at word frequencies is to compare the number of times a word is used with the number of documents it is used in. The inverse document frequency gives more weight to words that are not used as much across the documents. In our example, a document is a tweet. So a higher value for inverse document frequency means that a word is not used across the tweets while a lower value means that a word is used in many tweets. The *tf-idf* is the multiplication of the term (or word) frequency and the inverse document frequency - it can tell us how important a word is in the collection of documents (or in this case tweets).

```
View(total_tweet_words)

# Compute frequency of word by tweet
total_tweet_words <- total_tweet_words %>%
  mutate(freq = n / total_words)

View(total_tweet_words)

# Because there are so many individual tweets, we will first look at total words
# each day and the frequency of each word
total_day_words <- day_words %>%
  group_by(created_at_date) %>%
  summarise(total_words = sum(n),
            .groups = "drop")

total_day_words <- left_join(day_words, total_day_words, by = "created_at_date") %>%
  mutate(freq = n / total_words)

View(total_day_words)

# Create a plot of the total words in each day
ggplot(data = total_day_words,
       mapping = aes(x = freq)) +
  geom_histogram(bins = 30) +
  facet_wrap(~created_at_date,
              scales = "free") +
  scale_x_continuous(limits = c(0, 0.09)) +
  labs(x = "Word Frequency",
       y = "Count") +
  theme_classic()
```

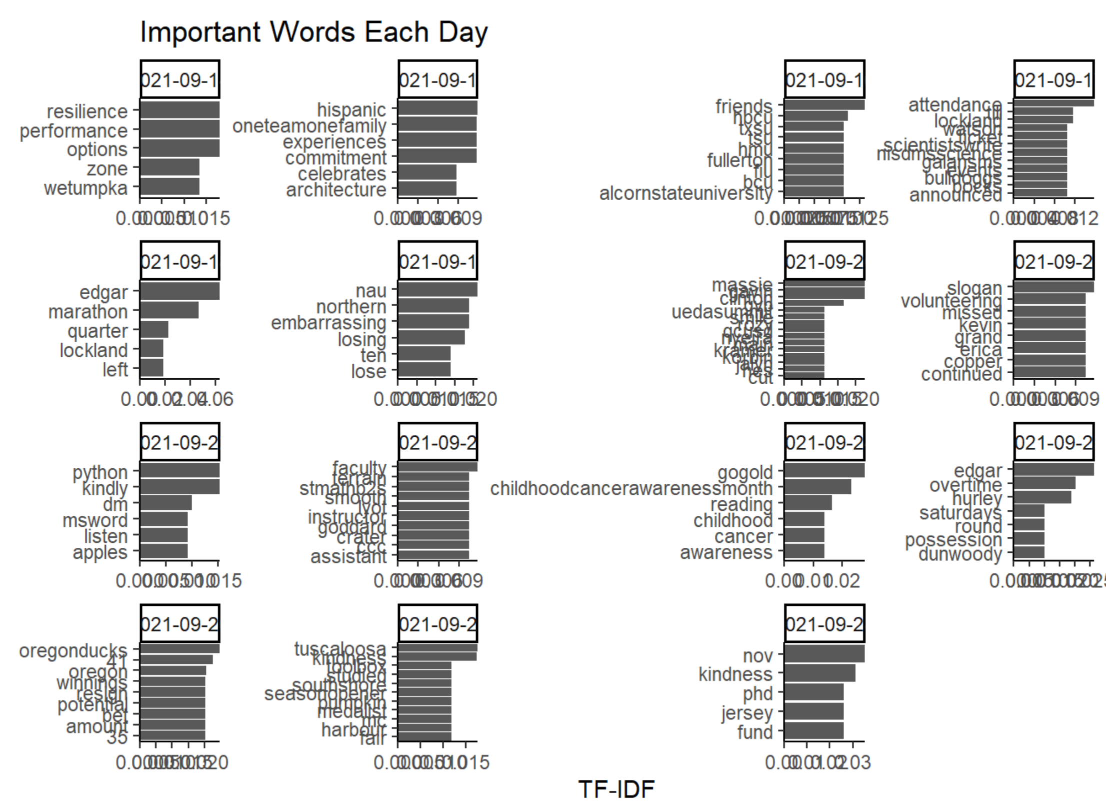
```
## Warning: Removed 30 rows containing missing values (geom_bar).
```




```
# Notice that there are only a few words that occur frequently and most words occur rarely

# Usually, you would do this analysis for each document (i.e. tweet), but since
# we have a large number of tweets, I grouped the data together by day instead.
# When analyzing data from a survey, the "documents" might be each open-ended
# question; for course evaluations, the "documents" might be each class. For
# other text analysis applications, the documents could be individual books,
# speeches, websites, etc.
# For tweets, we could also group the data into documents for each user
# or we could use a hashtag to create a document.

# Next we will use the bind_tf_idf() function to compute the tf-idf statistic
# We will keep the data grouped into "documents" of days, and this analysis
# will tell us which words are important each day but not too common across
# the days.

# Compute tf, idf, and tf-idf
day_tf_idf <- day_words %>%
  bind_tf_idf(word, created_at_date, n)

View(day_tf_idf)
# tf = term frequency
# idf = inverse document frequency
# tf_idf = term frequency * inverse document frequency

# View words with high tf-idf
day_tf_idf %>%
  arrange(-tf_idf)
```

```
## # A tibble: 5,175 x 6
##    created_at_date word             n     tf   idf tf_idf
##    <date>          <chr>        <int>  <dbl> <dbl>  <dbl>
##  1 2021-09-18      edgar           20 0.0314  2.01 0.0634
##  2 2021-09-18      marathon        11 0.0173  2.71 0.0468
##  3 2021-09-28      nov              2 0.0130  2.71 0.0352
##  4 2021-09-28      kindness         3 0.0195  1.61 0.0314
##  5 2021-09-24      gogold           6 0.0102  2.71 0.0276
##  6 2021-09-25      edgar            7 0.0130  2.01 0.0262
##  7 2021-09-28      fund             2 0.0130  2.01 0.0262
##  8 2021-09-28      jersey           2 0.0130  2.01 0.0262
##  9 2021-09-28      phd              2 0.0130  2.01 0.0262
## 10 2021-09-26      oregonducks      6 0.0225  1.10 0.0247
## # ... with 5,165 more rows
```

```
# Plot words with high tf-idf by day
ggplot(data = day_tf_idf %>%
         group_by(created_at_date) %>%
         top_n(5, tf_idf) %>%
         ungroup %>%
         mutate(created_at_date = as.factor(created_at_date),
                word = reorder_within(word, tf_idf, created_at_date)),
       mapping = aes(y = word,
                     x = tf_idf)) +
  geom_col() +
  facet_wrap(~created_at_date,
             scales = "free") +
  labs(title = "Important Words Each Day",
       x = "TF-IDF",
       y = "") +
  scale_y_reordered() +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```



Sentiment Analysis

The data set of words used in each tweet or document can also be used to evaluate the sentiment or opinion/feeling of the text. Certainly an algorithmic approach to determining the emotions behind a piece of text is not as accurate as human coding, but using a lexicon (i.e. dictionary) of words and their corresponding sentiments, we can match up the feelings of certain words. Once we have the sentiment of individual words, we can estimate the sentiment of the whole tweet or document.

There are three general lexicons in the `{tidytext}` package, and all three use single words.

- AFINN assign words with a score between -5 (most negative) and 5 (most positive); negative scores have negative sentiment while positive scores have positive sentiment
- bing categorizes words into categories of positive or negative
- nrc categorizes words into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust

The function `get_sentiments()` with the name of the lexicon will retrieve the desired lexicon.

You can add your own words to the sentiment dictionary if you'd like; this can be particularly helpful if there are keywords relevant to the topic of the text you are analyzing.

```
# View each lexicon
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

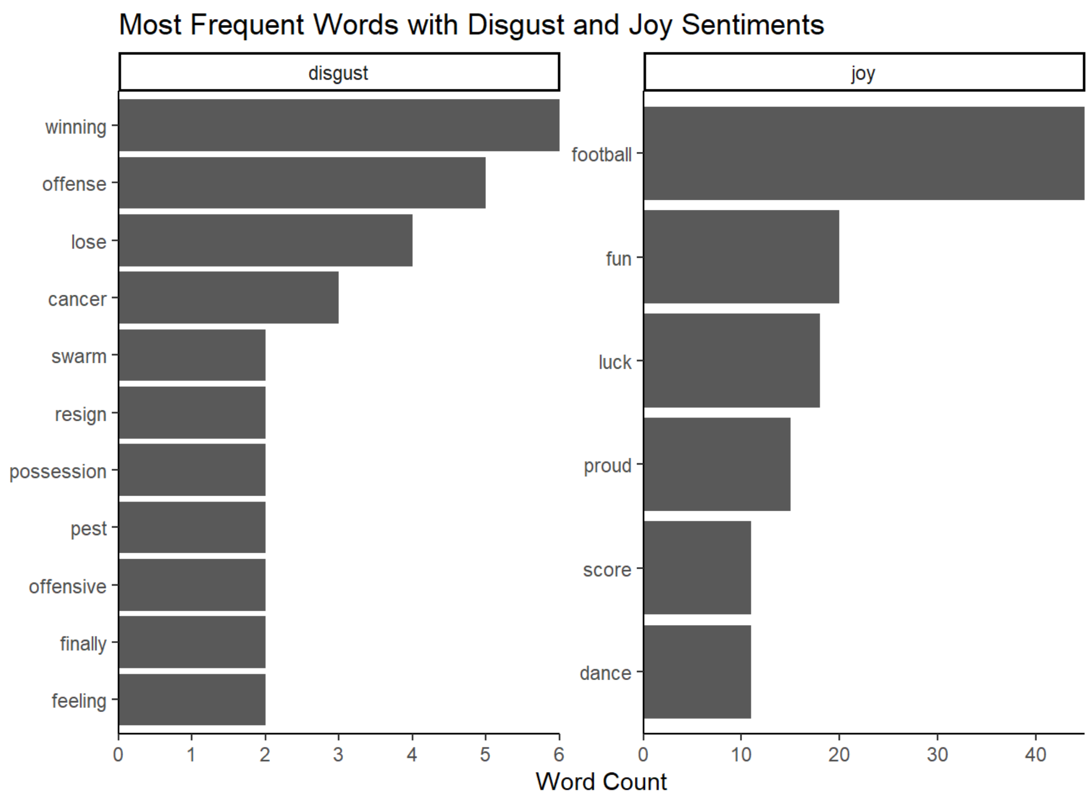
```
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # ... with 6,776 more rows
```

```
get_sentiments("nrc") # note that words can have multiple sentiments in nrc
```

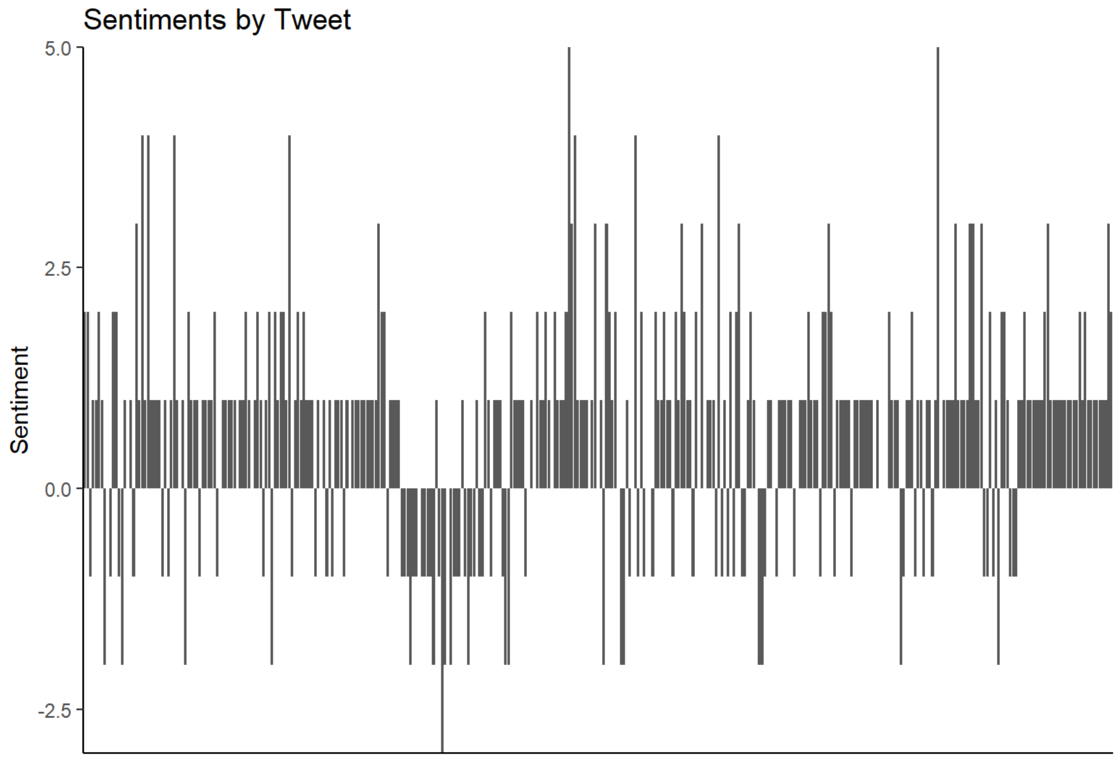
```
## # A tibble: 13,901 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # ... with 13,891 more rows
```

```
# Find common joy words in the tweets

nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

View(nrc_joy)

tweet_words_joy <- inner_join(tweet_word_freq,
                              nrc_joy,
                              by = "word")

View(tweet_words_joy)

# Find common disgust words in the tweets

nrc_disgust <- get_sentiments("nrc") %>%
  filter(sentiment == "disgust")

View(nrc_disgust)

tweet_words_disgust <- inner_join(tweet_word_freq,
                                  nrc_disgust,
                                  by = "word")

View(tweet_words_disgust)

# Plot joy and disgust together
tweet_words_joy_disgust <- bind_rows(tweet_words_joy, tweet_words_disgust) %>%
  group_by(sentiment) %>%
  top_n(5, n) %>%
  mutate(word = reorder_within(word, n, sentiment))

ggplot(data = tweet_words_joy_disgust,
       mapping = aes(x = n,
                     y = word)) +
  geom_col() +
  facet_wrap(~ sentiment,
             scales = "free") +
  labs(title = "Most Frequent Words with Disgust and Joy Sentiments",
       x = "Word Count",
       y = "") +
  scale_y_reordered() +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```



Most Frequent Words with Disgust and Joy Sentiments
disgust
joy
winning
offense
lose
cancer
swarm
resign
possession
pest
offensive
finally
feeling
football
fun
luck
proud
score
dance
0
1
2
3
4
5
6
0
10
20
30
40
Word Count

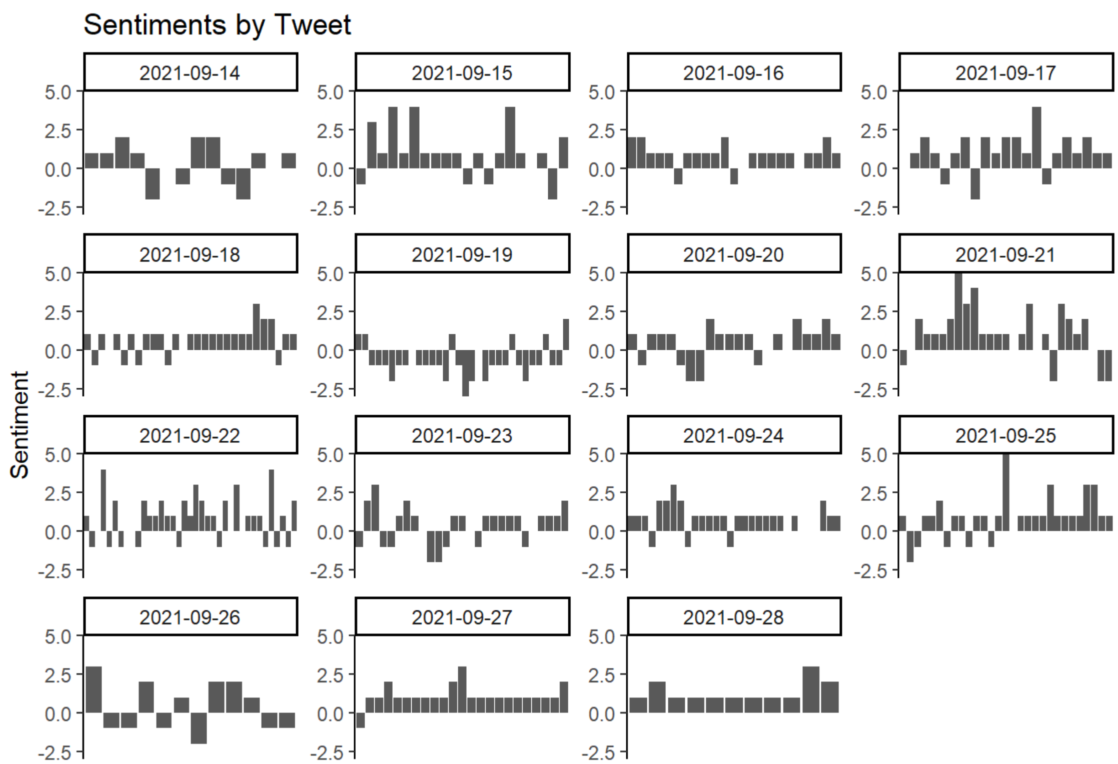
```r
# Looking and only positive and negative sentiment by word, we can evaluate the
# sentiment of each tweet or of a day of tweets.

# First, let's get the sentiment of each word
bing <- get_sentiments("bing")

sentiment_tweet_words <- inner_join(tweet_words,
                                    bing,
                                    by = "word")

View(sentiment_tweet_words)

# Next, we will count the number of positive and negative words per tweet
pos_neg_per_tweet <- sentiment_tweet_words %>%
  count(status_id, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n) %>%
  mutate(positive = ifelse(is.na(positive), 0 , positive),
         negative = ifelse(is.na(negative), 0, negative),
         sentiment = positive - negative)

View(pos_neg_per_tweet)

# Let's plot the sentiments by tweet
ggplot(data = pos_neg_per_tweet,
       mapping = aes(x = status_id,
                     y = sentiment)) +
  geom_col() +
  labs(title = "Sentiments by Tweet",
       x = "",
       y = "Sentiment") +
  scale_y_continuous(expand = c(0, 0)) +
  theme_classic() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



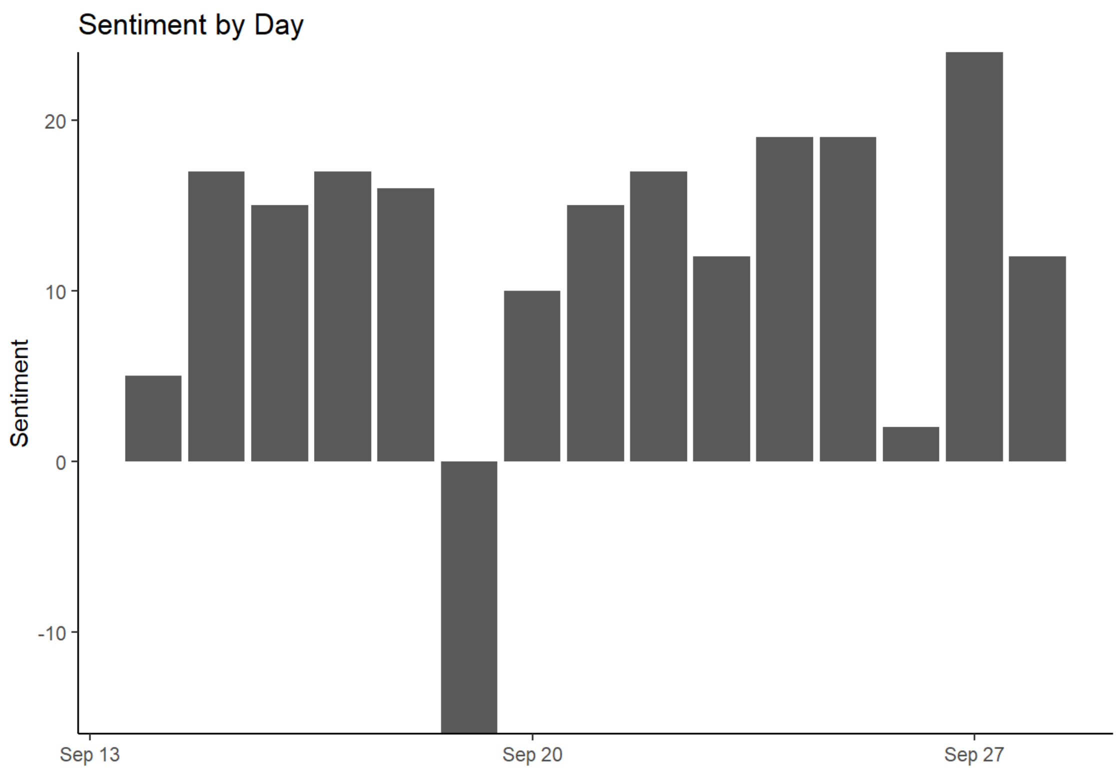

```
# We can also look at each tweet over time
pos_neg_per_tweet_date <- text_data_processed %>%
  mutate(created_at_date = as.Date(created_at, "%m/%d/%Y")) %>%
  select(status_id, created_at_date) %>%
  right_join(pos_neg_per_tweet, by = "status_id")
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%m/%d/%Y'
```

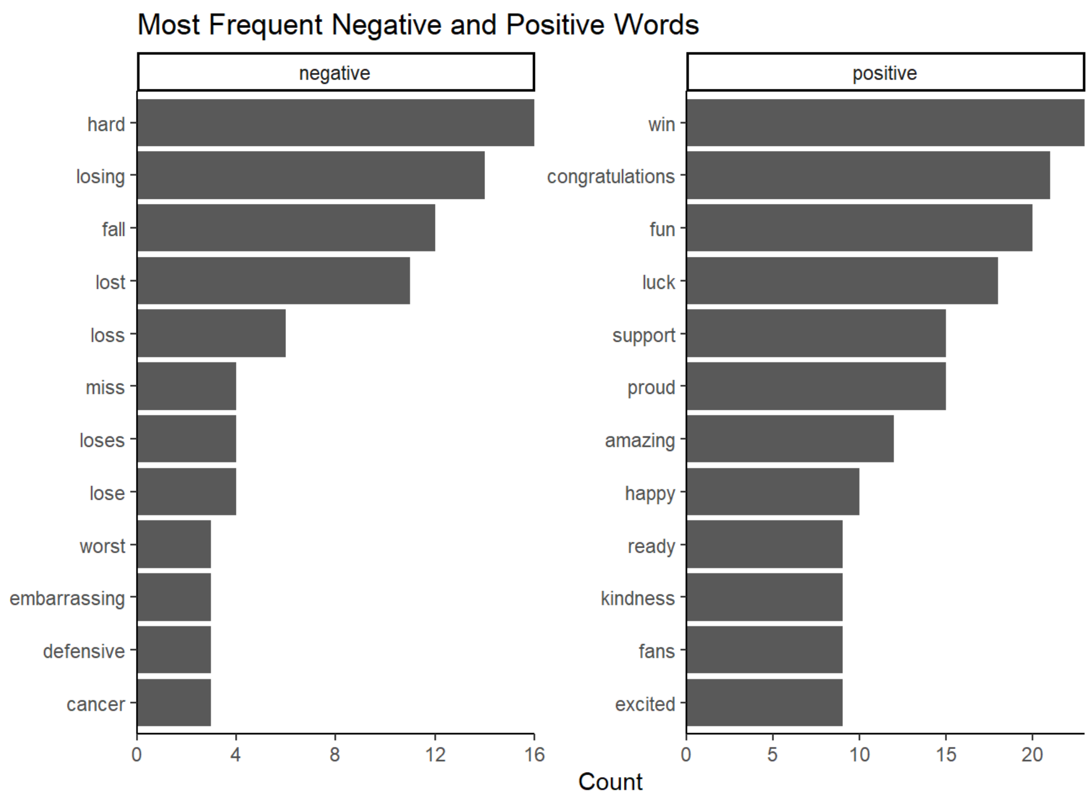
```
# Now let's get the general sentiment each day
ggplot(data = pos_neg_per_tweet_date,
       mapping = aes(x = status_id,
                     y = sentiment)) +
  geom_col() +
  facet_wrap(~created_at_date,
             scales = "free") +
  labs(title = "Sentiments by Tweet",
       x = "",
       y = "Sentiment") +
  scale_y_continuous(limits = c(-3, 5),
                     expand = c(0, 0)) +
  theme_classic() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.line.x = element_blank())
```




```
# Finally, let's find the general positive or negative sentiment for each day
sentiment_day_words <- inner_join(day_words,
                                  bing,
                                  by = "word")

View(sentiment_day_words)

pos_neg_per_day <- sentiment_day_words %>%
  count(created_at_date, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n) %>%
  mutate(positive = ifelse(is.na(positive), 0 , positive),
         negative = ifelse(is.na(negative), 0, negative),
         sentiment = positive - negative)

View(pos_neg_per_day)

# Plot sentiment by day
ggplot(data = pos_neg_per_day,
       mapping = aes(x = created_at_date,
                     y = sentiment)) +
  geom_col() +
  labs(title = "Sentiment by Day",
       x = "",
       y = "Sentiment") +
  scale_y_continuous(expand = c(0, 0)) +
  theme_classic() +
  theme()
```




```
# Next let's look at the most common positive and negative words overall
sentiment_words <- sentiment_tweet_words %>%
  group_by(sentiment, word) %>%
  summarise(n = sum(n),
            .groups = "drop")

View(sentiment_words)

# Plot the top 10
sentiment_words_top <- sentiment_words %>%
  group_by(sentiment) %>%
  top_n(10, n) %>%
  ungroup() %>%
  mutate(word = reorder_within(word, n, sentiment))


ggplot(data = sentiment_words_top,
       mapping = aes(x = n,
                     y = word)) +
  geom_col() +
  facet_wrap(~ sentiment,
             scales = "free") +
  labs(title = "Most Frequent Negative and Positive Words",
       x = "Count",
       y = "") +
  scale_y_reordered() +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```




```
# Let's compare the lexicons
afinn <- get_sentiments("afinn")

nrc <- get_sentiments("nrc")

table(bing$sentiment)
```

```
##
## negative positive
##     4781     2005
```

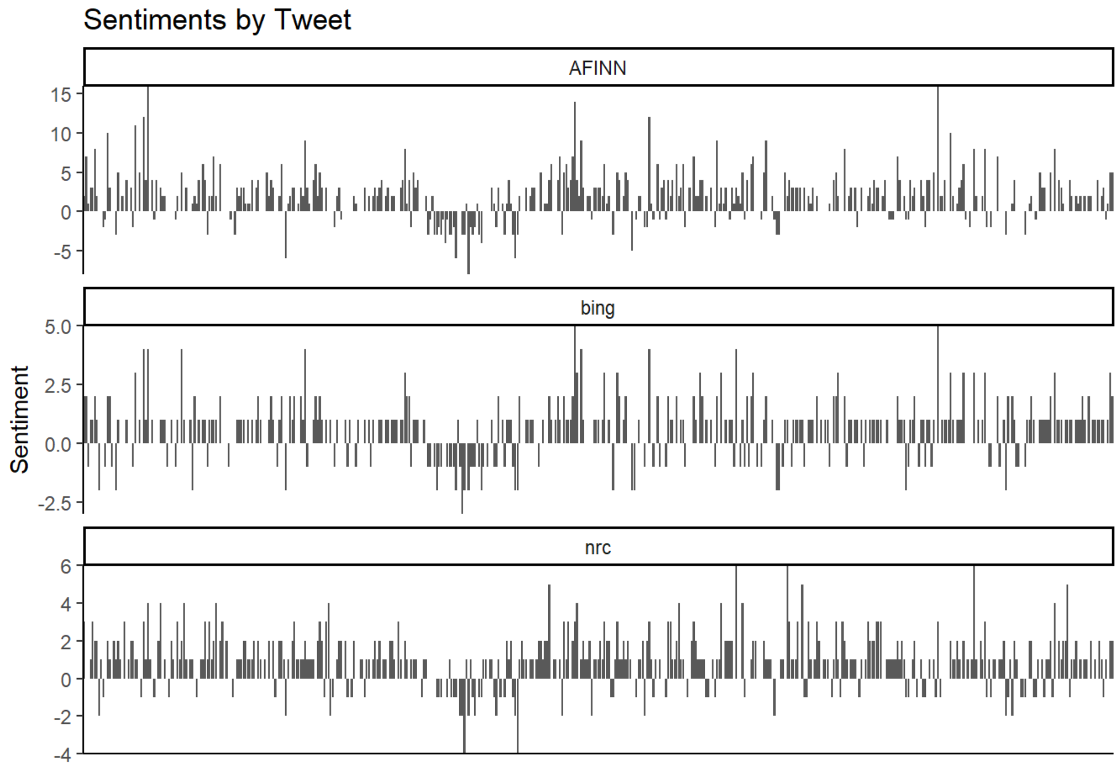
```
table(nrc$sentiment)
```

```
##
##        anger anticipation      disgust         fear          joy     negative
##         1247          839         1058         1476          689         3324
##     positive      sadness     surprise        trust
##         2312         1191          534         1231
```

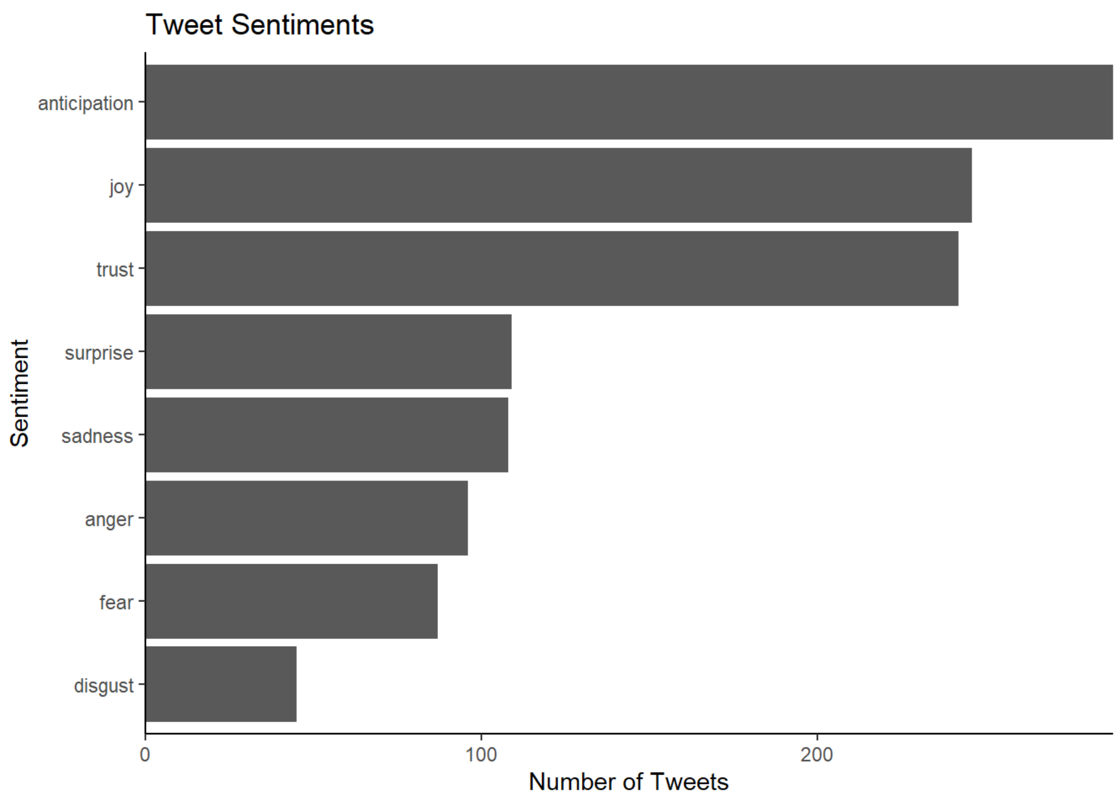
```
table(afinn$value)
```

```
##
##   -5   -4   -3   -2   -1    0    1    2    3    4    5
##   16   43  264  966  309    1  208  448  172   45    5
```

```
tweet_words_afinn <- inner_join(afinn, tweet_words, by = "word") %>%
  group_by(status_id) %>%
  summarise(sentiment = sum(value),
            .groups = "drop") %>%
  mutate(method = "AFINN") %>%
  select(status_id, sentiment, method)

tweet_words_nrc <- inner_join(nrc %>% filter(sentiment %in% c("positive", "negative")),
                              tweet_words, by = "word") %>%
  count(status_id, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n) %>%
  mutate(positive = ifelse(is.na(positive), 0 , positive),
         negative = ifelse(is.na(negative), 0, negative),
         sentiment = positive - negative,
         method = "nrc") %>%
  select(status_id, sentiment, method)

tweet_words_bing <- inner_join(bing, tweet_words, by = "word") %>%
  count(status_id, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n) %>%
  mutate(positive = ifelse(is.na(positive), 0 , positive),
         negative = ifelse(is.na(negative), 0, negative),
         sentiment = positive - negative,
         method  = "bing") %>%
  select(status_id, sentiment, method)

tweet_words_all <- bind_rows(tweet_words_afinn,
                             tweet_words_nrc,
                             tweet_words_bing)

View(tweet_words_all)

ggplot(data = tweet_words_all,
       mapping = aes(x = status_id,
                     y = sentiment)) +
  geom_col() +
  labs(title = "Sentiments by Tweet",
       x = "",
       y = "Sentiment") +
  facet_wrap(~ method,
             scale = "free_y",
             ncol = 1) +
  scale_y_continuous(expand = c(0, 0)) +
  theme_classic() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



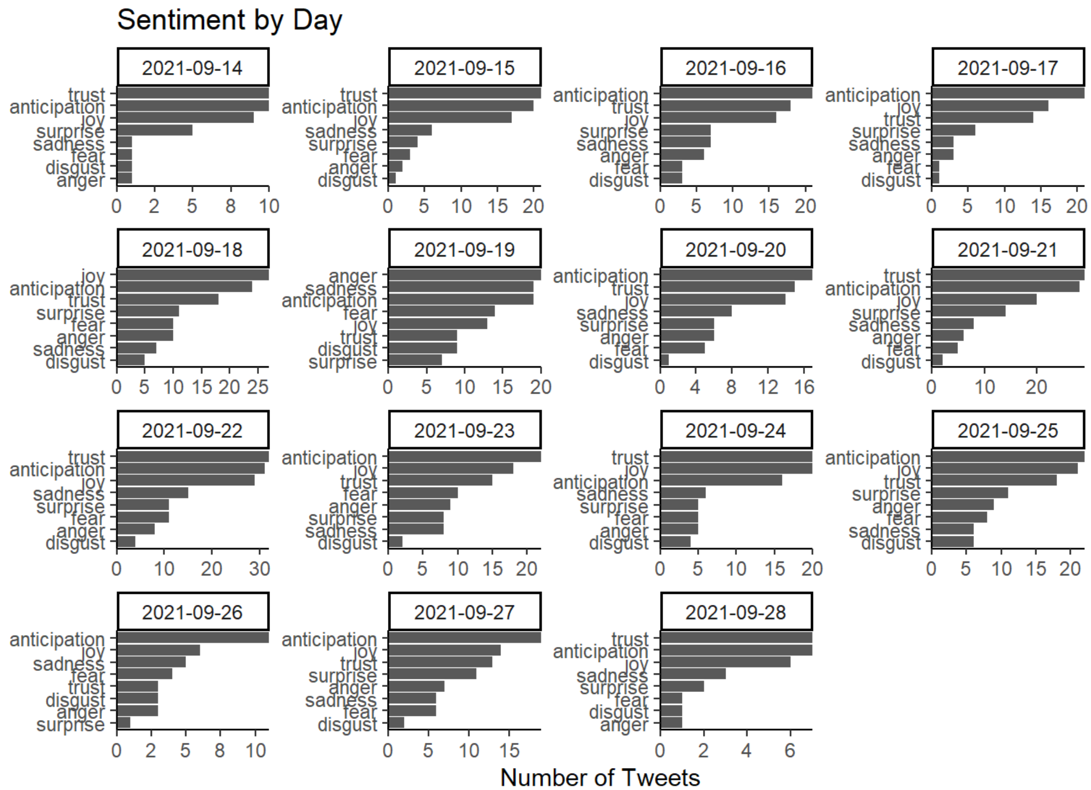

```
# Finally, let's look at the different feelings of each tweet
tweet_words_nrc_all <- inner_join(nrc %>% filter(sentiment != "positive" & sentiment != "negative"),
                                  tweet_words, by = "word") %>%
  count(status_id, sentiment)

View(tweet_words_nrc_all)

tweet_words_nrc_all_total <- tweet_words_nrc_all %>%
  count(sentiment) %>%
  mutate(sentiment = reorder(sentiment, n))

ggplot(data = tweet_words_nrc_all_total,
       mapping = aes(x = n,
                     y = sentiment)) +
  geom_col()  +
  labs(title = "Tweet Sentiments",
       x = "Number of Tweets",
       y = "Sentiment") +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```




```
# Let's look at tweet sentiments by day
tweet_words_nrc_all_date <- text_data_processed %>%
  mutate(created_at_date = as.Date(created_at, "%m/%d/%Y")) %>%
  select(status_id, created_at_date) %>%
  right_join(tweet_words_nrc_all, by = "status_id") %>%
  count(created_at_date, sentiment) %>%
  mutate(sentiment = reorder_within(sentiment, n, created_at_date))
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%m/%d/%Y'
```

```
ggplot(data = tweet_words_nrc_all_date,
       mapping = aes(x = n,
                     y = sentiment)) +
  geom_col() +
  facet_wrap(~ created_at_date,
             scales = "free") +
  labs(title = "Sentiment by Day",
       x = "Number of Tweets",
       y = "") +
  scale_y_reordered() +
  scale_x_continuous(expand = c(0, 0),
                     labels = number_format(accuracy = 1)) +
  theme_classic()
```



Moving Beyond Words

So far we have only looked at individual words. We may want to consider sets of words or whole sentences. We can do this by either breaking the text into sections using some sort of consistent header (for example, "chapter" in a book) or by using a fixed grouping of the words. In our example, we have already looked at sentiment by tweet because each tweet was its own document, but there may be cases in a text analysis where there are multiple entries in each document. For example, each document might be a survey question so there would be multiple responses within each document. It might be useful to analyze the responses together, or there might be an interest in breaking them into individual responses or some other categorization.

Another reason we might want to do this is that single words on their own may hide meaning that words together may show. For example, "I am not feeling good today" would end up with positive sentiment due to "good" and "feeling", but it is not really a positive statement. In this section, we will look at how to create n-grams, or sets of words together.

```
# Get sets of two words
text_ngrams <- text_data_processed %>%
  unnest_tokens(ngram, text, token = "ngrams", n = 2)

View(text_ngrams) # note that words are duplicated

dim(text_ngrams)
```

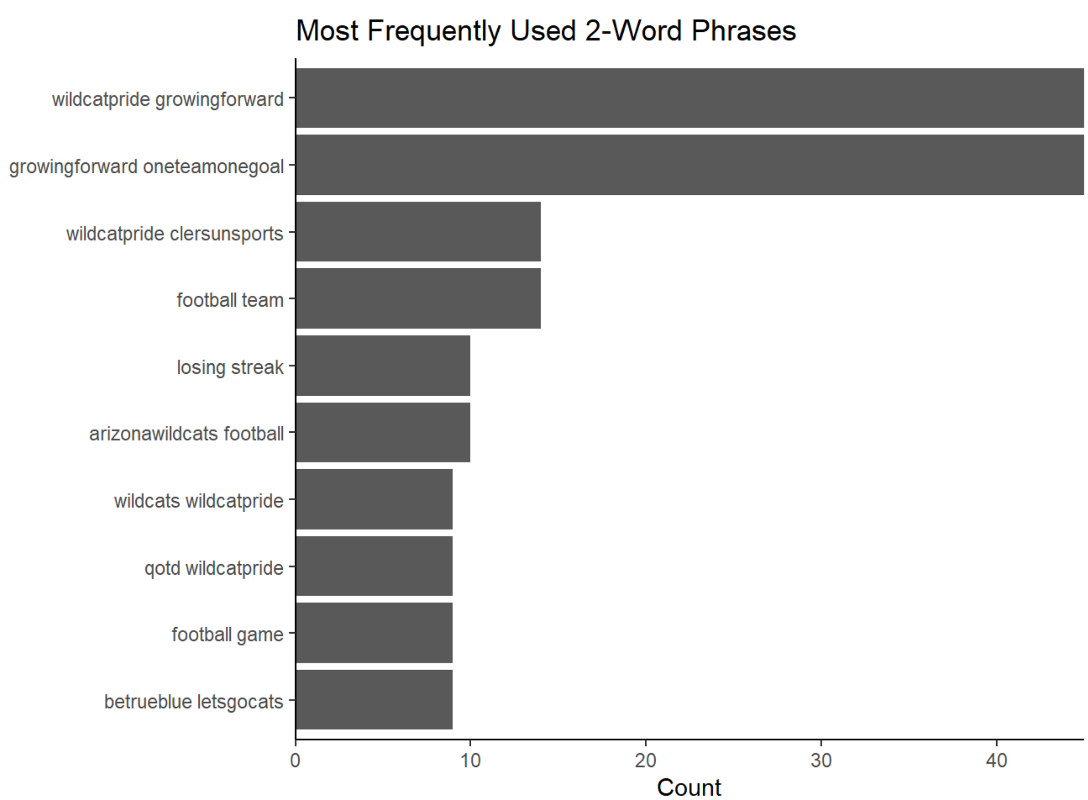
```
## [1] 11873     9
```

```
# Remove stop words
text_ngrams <- text_ngrams %>%
  separate(ngram, c("word_1", "word_2"), sep = " ") %>%
  anti_join(stop_words, by = c("word_1" = "word")) %>%
  anti_join(stop_words, by = c("word_2" = "word")) %>%
  unite(ngram, word_1, word_2, sep = " ")

View(text_ngrams) # note that words are duplicated

dim(text_ngrams)
```

```
## [1] 4360    9
```

```
# Get counts of n-grams
text_ngrams_freq <- text_ngrams %>%
  count(ngram, sort = TRUE)

View(text_ngrams_freq)

# Plot top n-grams
ggplot(data = text_ngrams_freq %>%
         top_n(10),
       mapping = aes(y = reorder(ngram, n),
                     x = n)) +
  geom_col() +
  labs(title = "Most Frequently Used 2-Word Phrases",
       x = "Count",
       y = "") +
  scale_x_continuous(expand = c(0, 0)) +
  theme_classic()
```

```
## Selecting by n
```



```
# Create word cloud
text_ngrams_freq %>%
  with(wordcloud(ngram, n, max.words = 25))
```

```
## Warning in wordcloud(ngram, n, max.words = 25): wildcatpride growingforward
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(ngram, n, max.words = 25): growingforward oneteamonegoal
## could not be fit on page. It will not be plotted.
```