Clustering Part 2

Homework

Prof. Bisbee

Due Date: 2023-04-10

Getting back to where we left off

We are going to start from the prepared `Trump_tweet_words.Rds` file from last time. Let's load it in, along with a bunch of useful packages for text analysis.

```
require(tidyverse)
require(tidytext)
require(scales)
tweet_words <- readRDS(file="../data/Trump_tweet_words.Rds") %>% # Note you can add data manipulations directly to the code that opens the file!
  mutate(PostPresident = Tweeting.date > "2016-11-03")
```

Comparing Word Use Pre/Post Using Log Odds Ratio

So far we have focused on frequency of word use, but another way to make this comparison is to look at the relative "odds" that each word is used pre/post presidency. After all, "Trump" is used by Trump both before and after his presidency so perhaps that is not a great indicator of content. We could instead consider the relative rate at which a word is used Post-Presidency relative to Pre-presidency.

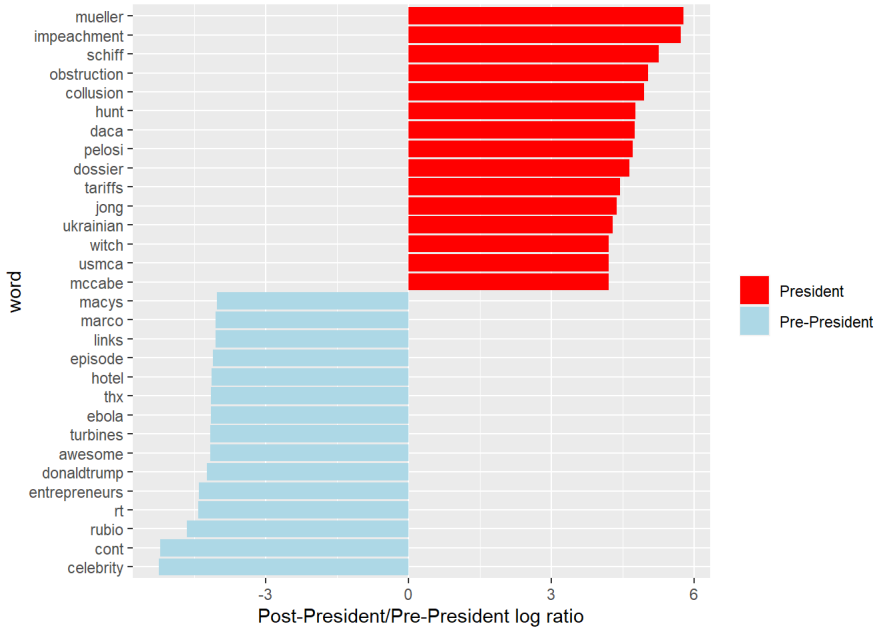
We are going to count each word stem use pre and post-presidency, then select only those words that were used at least 5 times, then `spread` the data so that if a word appears Pre-Presidency but not Post-Presidency (or visa-versa) we will create a matching word with the filled in value of 0, then we are going to ungroup the data so that the observation is now a word rather than a word-timing combination (look to see how the tibble changes before and after the ungroup() by running these code snippets separately to see). Then we are going to `mutate_each` to compute the fraction of times a word is used relative to all words (the `.` indicates the particular value of each variable – note that we are adding a `+ 1` to each of those values to avoid errors when taking the log later). We then compute the `ratio` by computing the relative frequency of each word used pre and post presidency and take the log of that ratio because of extreme outliers before arranging the tibble in decreasing value of ratio

So let's compute the log odds ratio for each word pre and post presidency.

```
prepost_ratios <- tweet_words %>%
  count(word, PostPresident) %>%
  filter(sum(n) >= 5) %>%
  spread(PostPresident, n, fill = 0) %>%
  ungroup() %>%
  mutate_each(funs((. + 1) / sum(. + 1)), -word) %>%
  mutate(ratio = `TRUE` / `FALSE`) %>%
  mutate(logratio = log(ratio)) %>%
  arrange(-logratio)
```

Now let's plot the top 15 most distinctive words (according to the log-ratio we just computed) that were tweeted before and after Trump was elected president.

```
prepost_ratios %>%
  group_by(logratio > 0) %>%
  top_n(15, abs(logratio)) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  ylab("Post-President/Pre-President log ratio") +
  scale_fill_manual(name = "", labels = c("President", "Pre-President"),
                    values = c("red", "lightblue"))
```



You could look at other splits. Pre-post his first impeachment? 2016 versus 2017? Note that the log-ratio is a comparison of a binary condition.

Sentiment Analysis

Everything we have done so far is “basic” in the sense that we are looking at frequencies and proportions. We have not done anything particularly fancy (apart from perhaps defining the log odds-ratio but even that is just applying a function to our data).

A prominent tool used to analyze text is called “sentiment analysis.” Unlike clustering algorithms that we discussed early that were unsupervised, “sentiment analysis” is an analysis that classifies the meaning/sentiment of text based on a dictionary of words that are assumed to be true. That is, we are using an object with assumed meaning to learn about the characteristics of a text in terms of concepts that are predefined and predetermined.

Sentiment analysis sounds like it is very complex, and it is because of the complexity of language and how the meaning/sentiment of words can change based on context.

Analyzing sentiment requires using a dictionary of predefined sentiment and then using the frequency (or a similar measure) of words with sentiment to classify a text. If we have information on the sentiment of a tweet (perhaps via crowdsourcing) then we can use the methods of prior classes to try to determine how close other tweets are to the tweets that have been identified as belong to each sentiment. i.e., we can use features of a tweet to classify it given the relationship of known tweets.

If we do not have this predefined tweet-level sentiment, we can characterize sentiment by counting the number of times a set of predefined words are used and then using the resulting distributions to interpret the “sentiment” of the text. Note that it is always preferrable to use the former to account for the contextual meaning of language, but characterizing the frequency of sentiments can also be of interest.

There are several dictionaries of sentiment, but we are going to use the NRC Word-Emotion Association (http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm) lexicon created by, and published in Saif M. Mohammad and Peter Turney. (2013),
` Crowdsourcing a Word-Emotion Association Lexicon.'' Computational Intelligence, 29(3): 436-465. This is available from the tidytext package, which associ
to the console.

```
library(tidytext)
library(textdata)
nrc <- get_sentiments("nrc")
nrc
```

```
## # A tibble: 13,901 × 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
```

Let’s get overall sentiment as a fraction of words used in tweet grouped by PostPresidency (could also group by `Tweeting.year` or `Tweeting.hour`).

Start by defining the relative frequency of each word being used pre/post presidency conditional on being tweeted at least 5 times.

```
word_freq <- tweet_words %>%
  group_by(PostPresident) %>%
  count(word) %>%
  filter(sum(n) >= 5) %>%
   mutate(prop = prop.table(n))
```

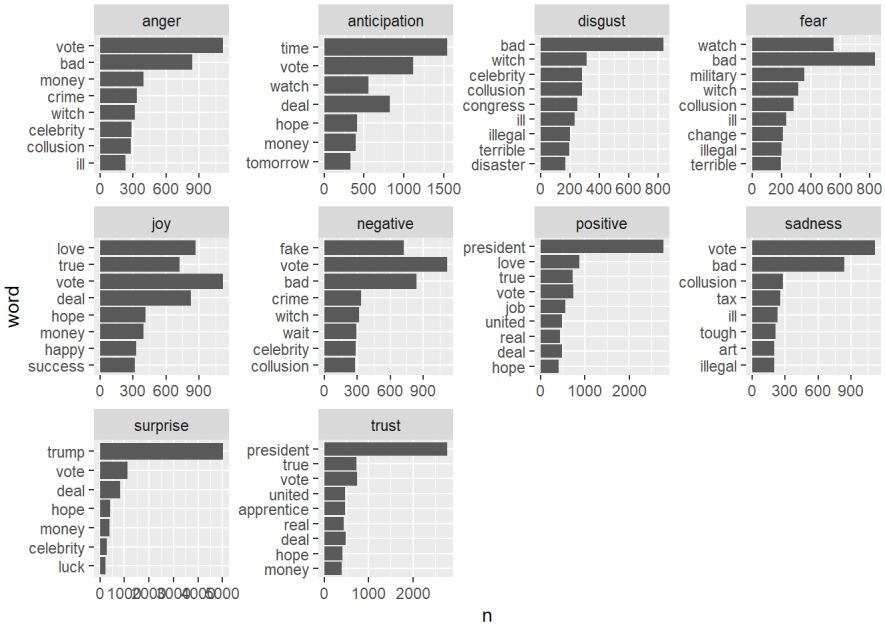
Now we use `inner_join` to select only the words in `tweet_words` that are contained in the NRC sentiment analysis word list. Note that `inner_join` keeps only the observations that are common to both `tweet_words` and `nrc` so it is going to keep just the words that have an associated sentiment. Note that if we stem the word tokens this would be problematic!

```
word_freq_sentiment <- word_freq %>%
    inner_join(nrc, by = "word")
```

Note that the proportion `prop` that we calculated above will no longer sum to 1 because: 1. we dropped words outside of the NRC word list with the `inner_join` , 2. each word can appear in multiple sentiments. That said, the proportion is still meaningful as a measure of the relative importance of each word, even if the interpretation of the value is somewhat problematic.

Now let's plot the top most typically used words in each sentiment.

```
word_freq_sentiment %>%
  group_by(sentiment) %>%
  top_n(10, n) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  facet_wrap(~ sentiment, scales = "free", nrow = 3) +
  geom_bar(stat = "identity") +
  coord_flip()
```



One way to measure sentiment is simply the number of positive sentiments - the number of negative sentiments.

Let's go back to our original tibble where each observation was a word, use `inner_join` to extract sentiments and then create a new tibble `tweet_sentiment_summary` that summarizes the sentiment of each tweet.

```
tweet_sentiment <- tweet_words %>%
    inner_join(nrc, by = "word")

tweet_sentiment_summary <- tweet_sentiment %>%
  group_by(PostPresident, sentiment) %>%
  count(document,sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

Note the use of `pivot_wider` here. This transforms a "long" tibble (many rows) into a "wide" tibble (many columns). Now each observation is a tweet (instead of a word in a tweet).

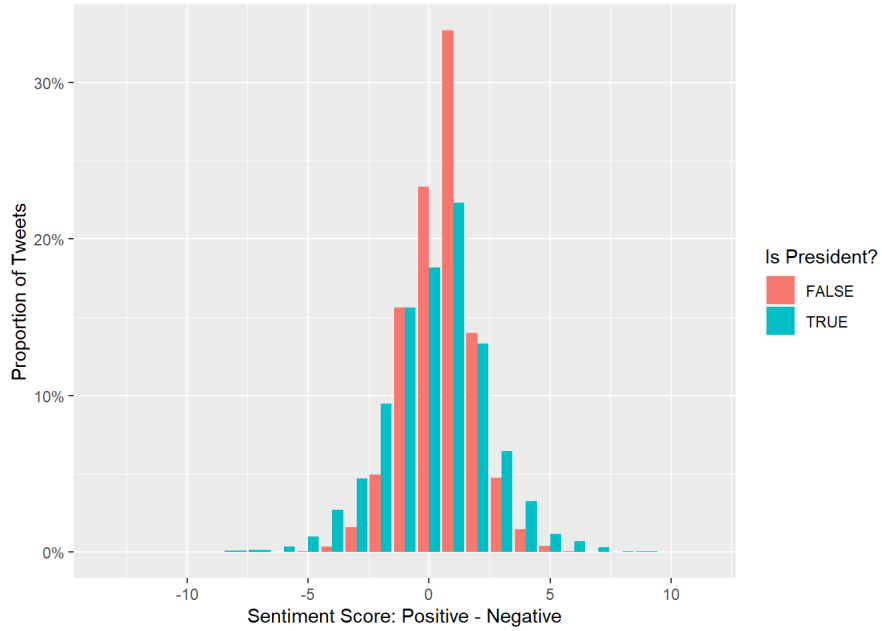
If we wanted to see how many total words were used in the all of the tweets we observed pre-post presidency we can summarize.

```
tweet_sentiment_summary  %>%
  group_by(PostPresident) %>%
  mutate(ntweet = 1) %>%
  summarize(across(-document, sum))
```

```
## # A tibble: 2 × 13
##   PostPresident anger anticipation disgust  fear   joy negative positive sadness
##   <lgl>         <int>        <int>   <int> <int> <int>    <int>    <int>   <int>
## 1 FALSE          8319        13781    5524  8195 12792    15307    28118    8185
## 2 TRUE           7115         6848    4897  6845  5562    12679    14982    5571
## # … with 4 more variables: surprise <int>, trust <int>, sentiment <int>,
## #   ntweet <dbl>
```

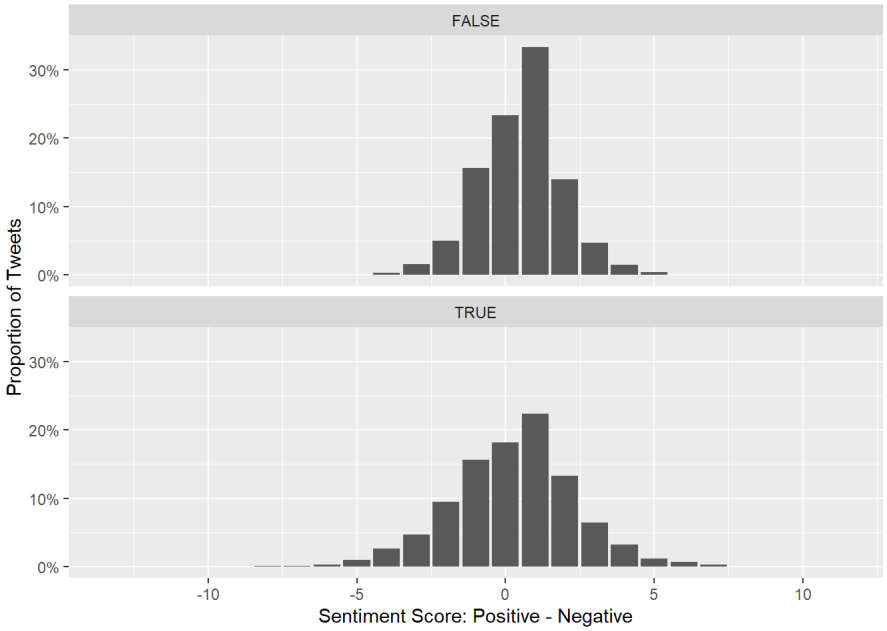
We can plot the distribution of sentiment scores pre/post presidency as follows. Note that the `position` controls where the bars for each `fill` are placed. (Try running it without `position` to see the default.) The `aes` of `geom_bar` is defined so as to plot the proportion rather than the frequency. Here we have told ggplot to calculate the proportion for us.

```
tweet_sentiment_summary %>%
ggplot(aes(x = sentiment, fill=PostPresident)) +
geom_bar(aes(y = ..prop..), position=position_dodge()) +
scale_y_continuous(labels = percent) +
labs(y= "Proportion of Tweets", x = "Sentiment Score: Positive - Negative", fill="Is President?")
```



As before, we could also use `facet_wrap` here. How to think about nrow here. Should it be 1 or 2? What is the comparison you want to make - comparing across x-axis or y-axis? Note also that we chose `scales="fixed"` this time. Why is this important?

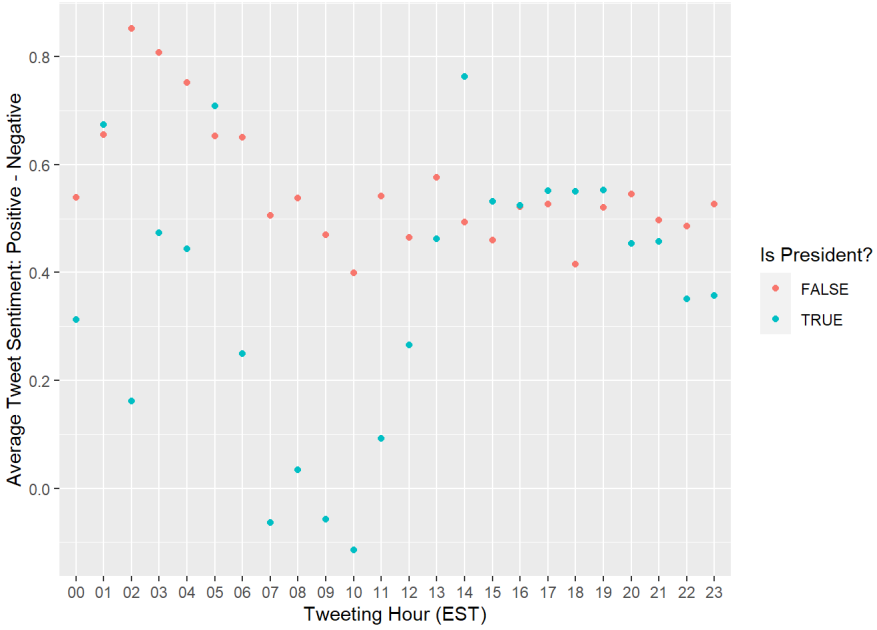
```
tweet_sentiment_summary %>%
  ggplot(aes(x = sentiment)) +
  facet_wrap(~ PostPresident, scales = "fixed", nrow = 2) +
  geom_bar(aes(y = ..prop..)) +
  scale_y_continuous(labels = percent) +
  labs(y= "Proportion of Tweets", x = "Sentiment Score: Positive - Negative")
```



Can also see how it varies by hour.

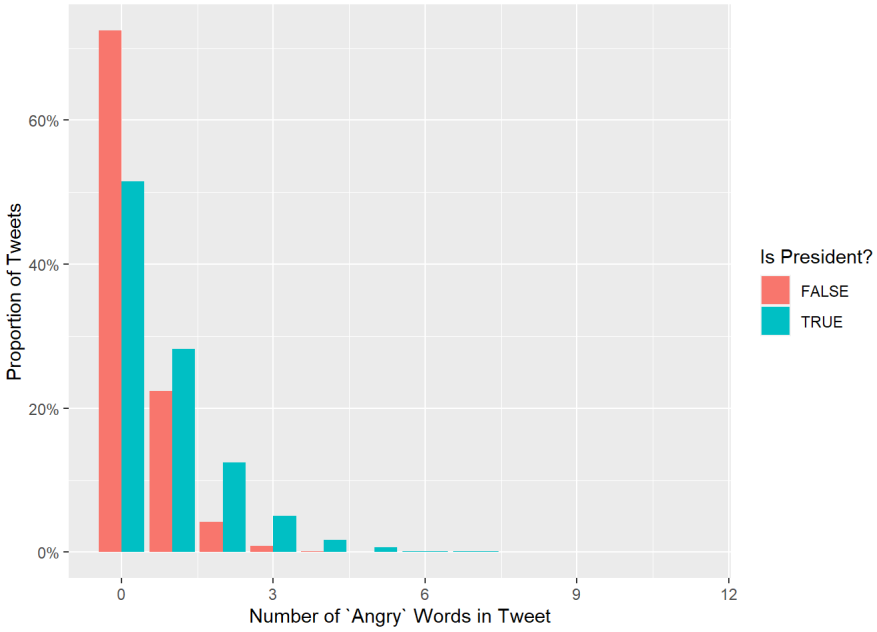
To do this we need to go back to the original tibble to `group_by Tweeting.hour`.

```
tweet_sentiment %>%
  group_by(PostPresident,Tweeting.hour,sentiment) %>%
  count(document,sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  summarize(AvgSentiment = mean(sentiment)) %>%
  ggplot(aes(y = AvgSentiment, x= Tweeting.hour, color=PostPresident)) +
  geom_point() +
  labs(x = "Tweeting Hour (EST)", y = "Average Tweet Sentiment: Positive - Negative", color = "Is President?")
```



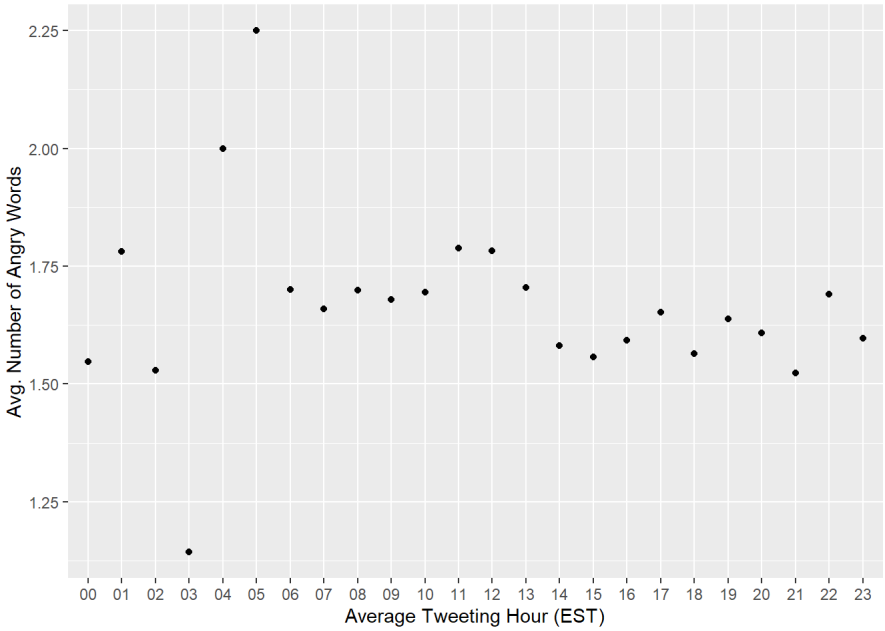
We can also dive in deeper and focus on particular sentiments. What is the distribution of the number of “angry” words being used in each tweet? Was Trump getting “angrier” over time?

```
tweet_sentiment_summary %>%
ggplot(aes(x = anger, fill=PostPresident)) +
geom_bar(aes(y = ..prop..), position=position_dodge()) +
scale_y_continuous(labels = percent) +
labs(y= "Proportion of Tweets", x = "Number of `Angry` Words in Tweet", fill="Is President?")
```



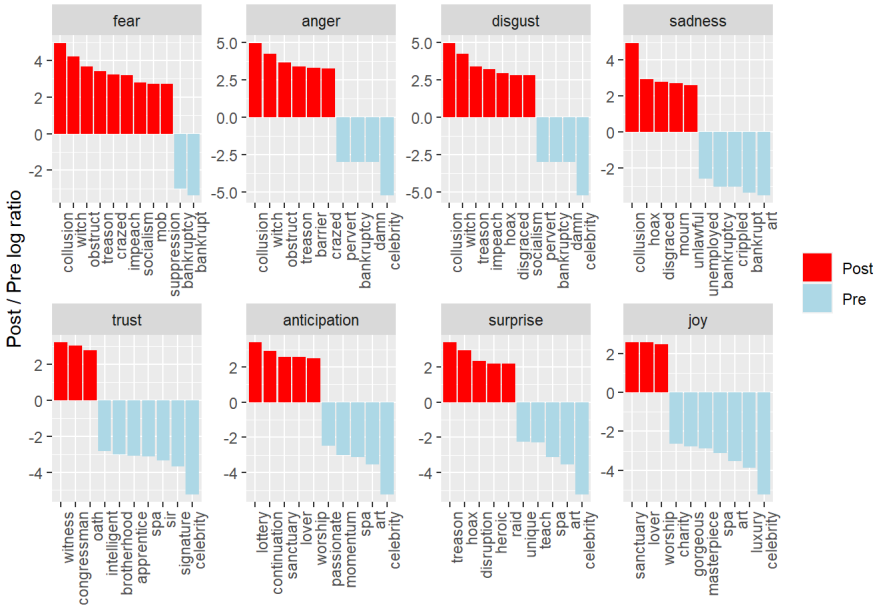
Or angrier based on the time of day post-presidency?

```
tweet_sentiment %>%
  filter(PostPresident==TRUE, sentiment == "anger") %>%
  group_by(Tweeting.hour) %>%
  count(document,sentiment) %>%
  summarize(AvgAnger = mean(n)) %>%
  ggplot(aes(y = AvgAnger, x= Tweeting.hour)) +
  geom_point() +
  labs(x = "Average Tweeting Hour (EST)", y = "Avg. Number of Angry Words")
```



And how about which words are most distinctively used in each sentiment (using log-odds ratio)?

```
prepost_ratios %>%
  inner_join(nrc, by = "word") %>%
  filter(!sentiment %in% c("positive", "negative")) %>%
  mutate(sentiment = reorder(sentiment, -logratio),
         word = reorder(word, -logratio)) %>%
  group_by(sentiment) %>%
  top_n(10, abs(logratio)) %>%
  ungroup() %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  facet_wrap(~ sentiment, scales = "free", nrow = 2) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(x = "", y = "Post / Pre log ratio") +
  scale_fill_manual(name = "", labels = c("Post", "Pre"),
                    values = c("red", "lightblue"))
```



Now you can blow the doors off of this!

How does the sentiment change over time? At various times of the day?

Note also that this is very basic stuff. Instead of defining sentiment at the token level and trying to aggregate up we can define it at the tweet level and then use the methods we talked about last time to try to classify tweet-level sentiments by determining how close each tweet is to the tweets classified according to each sentiment. This kind of "supervised" learning would require a tweet-level measure of sentiment that we could then predict using features as before (e.g., words, time of day, date). To do so we could build a prediction model and predict the tweet.

We can also try to use the most-frequently used sentiment to classify each tweet and see the most frequently "sentiment" associated with each tweet. If, for example, we were to classify the sentiment of each tweet using the modal sentiment (i.e., the most frequently appearing sentiment) we would get the following.

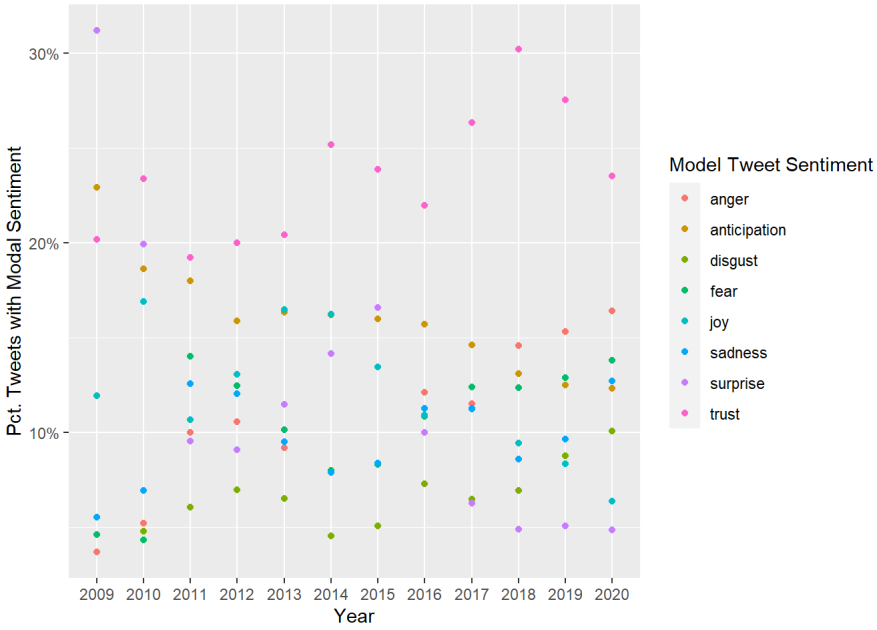
```
tweet_sentiment %>%
  filter(sentiment != "positive" & sentiment != "negative") %>%
  group_by(document) %>%
  count(sentiment) %>%
  top_n(1,n) %>%     # select the most frequently appearing sentiment
  group_by(sentiment) %>%
  count() %>%
  ungroup() %>%
  mutate(PctSentiment = n/sum(n))
```

```
## # A tibble: 8 × 3
##   sentiment        n PctSentiment
##   <chr>        <int>        <dbl>
## 1 anger         6530       0.105
## 2 anticipation  9548       0.154
## 3 disgust       3926       0.0633
## 4 fear          6488       0.105
## 5 joy           8092       0.130
## 6 sadness       5942       0.0958
## 7 surprise      6750       0.109
## 8 trust        14752       0.238
```

What do you think about this?

Now let's graph the proportion of tweets according to the modal sentiment over time.

```
tweet_sentiment %>%
  filter(sentiment != "positive" & sentiment != "negative") %>%
  group_by(Tweeting.year,document) %>%
  count(sentiment) %>%
  top_n(1,n) %>%
  group_by(Tweeting.year,sentiment) %>%
  count() %>%
  ungroup(sentiment) %>%
  mutate(PctSentiment = n/sum(n)) %>%
  ggplot(aes(x=Tweeting.year, y = PctSentiment, color = sentiment)) +
  geom_point() +
  labs(x = "Year", y = "Pct. Tweets with Modal Sentiment", color = "Model Tweet Sentiment") +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



You can literally do a thousand descriptive things with sentiment!