Topic 10: Text, Tweets, Sentiment, Part 4

Josh Clinton

2022-07-12

Tweets, Texts, and Sentiments, oh my!

Lots of press attention (and academic/commercial research) to social media communication. Trying to characterize (in the hopes of predicting) behavior and "sentiment." Many are trying to do this at scale – analyzing the behavior of large numbers of indvidual users – including using clustering methods to uncover different "types" of users that can be used to understand and predict human opinion and behavior across a wide range of activities and concepts.

Perhaps no individual has been more scrutinized than President Trump – who was arguably defined by his use of twitter as a medium of communication, agenda-setting, and mobilization. Multiple news stories and academic papers have focused on the corpus of Trump Tweets.

For example, the NY Times did an expose here whereby they had reporters read every single tweet and classify it. (See the methodology here. Remarkably, this was work that was done by hand. Hopefully by the end of class you can see how you could do something similar using R! More similar to what we are going to do is the work by fivethirtyeight.

We used to be able to read in Trump data via the Twitter API, but since that has been deactivated we are going to use data that was collected and posted here.

Note that you could also look at news coverage using data that people have collected on the scrolling chryons at the bottom of the screen here.

As an example of a recent publication in *Nature: Humanities and Social Sciences Communications* using sentiment analysis see: Sentiments and emotions evoked by news headlines of coronavirus disease (COVID-19) outbreak.

Let's load the packages we are going to us into memory. You may need to install some of these/

```
library(readr)
library(tidyverse)
library(lubridate)
library(scales)
```

Just to give you a sense of the preprocessing, here I read in a csv file and did some manipulations

```
trumptweets <- read_csv("~/GitHub/vandy_ds_1000_materials/Lectures/Topic10_Clustering/data/trumptweets.c
```

```
## Rows: 41122 Columns: 9-- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): link, content, mentions, hashtags
## dbl  (3): id, retweets, favorites
## lgl  (1): geo
## dttm (1): date
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#View(trumptweets)
glimpse(trumptweets)
```

```
## Rows: 41,122
## Columns: 9
## $ id        <dbl> 1698308935, 1701461182, 1737479987, 1741160716, 1773561338, ~
## $ link      <chr> "https://twitter.com/realDonaldTrump/status/1698308935", "ht~
## $ content   <chr> "Be sure to tune in and watch Donald Trump on Late Night wit~
## $ date      <dttm> 2009-05-04 20:54:25, 2009-05-05 03:00:10, 2009-05-08 15:38:~
## $ retweets  <dbl> 500, 33, 12, 11, 1399, 27, 14, 18, 14, 20, 38, 64, 18, 29, 8~
## $ favorites <dbl> 868, 273, 18, 24, 1965, 26, 16, 25, 8, 48, 67, 102, 17, 53, ~
## $ mentions  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ hashtags  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ geo       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
tweets <- trumptweets %>%
  select(id, content, date, retweets, favorites) %>%
  mutate(date = with_tz(date, "EST"),
         Tweeting.hour = format(date, format = "%H"),
         Tweeting.date = format(date, format = "%Y-%m-%d"),
         Tweeting.date = as.Date(Tweeting.date),
         Tweeting.year = as.factor(format(date, format = "%Y")))
```

First thing we always do is to see what we have so we know what we are working with and what we have to do to answer the questions we want to answer. Nese we select the variables we want and practice creating some time objects for future use. (We are using the `lubridate` package for the date manipulations.) Note that `date` had the time of the tweet in UTC so we used the `with_tz` function from `lubridate` package to change the time zone to Eastern Standard Time (as that is where Washington, DC, Bedminster, NJ, and Mar Lago, FL are located) – note that dates are also changed if the timezone change crosses days (a benefit of saving the object as a date object!).

Because our data is larger than usual, we want to keep an eye on how much is loaded into memory. Since we no longer need `trumptweets` let us remove it via `rm()`.

```
rm(trumptweets)
```

Let's focus on asking the question of how Trump's Twitter behavior changed over time. This is a broad question, so let's break it up into a few specific questions to tackle using the tools we have talked about thus far to help demonstrate that based on the concepts and code we have talked about in class you are able to do quite a bit.

1. How did the frequency of Trump's tweets change over time – and, in particular, once he was elected President.
2. When did Trump tweet? Was "Executive Time" a change to his behavior or did he always have a self-defined "Executive Time"?
3. Which of his tweets were most impactful? (Measures via Rewtweets and Favorites).

All of this can be done at the level of the tweet using tools we have previously used and discussed in class – i.e., no text analysis required. So we will start there. Sometimes simple analyses will get you what you need and complexity for the sake of complexity is not a virtue.

After using the data – and conditional means – to answer these descriptive questions we can then pivot to analyze what was being tweeted about using several tools that will get into the analysis of the content of the text being tweeted.

4. What were the topics that Trump was tweeting about before and after becoming president? (`kmeans`)
5. What were the dominant "sentiments" being expressed by Trump and how did that change after becoming president. (Sentiment Analysis)

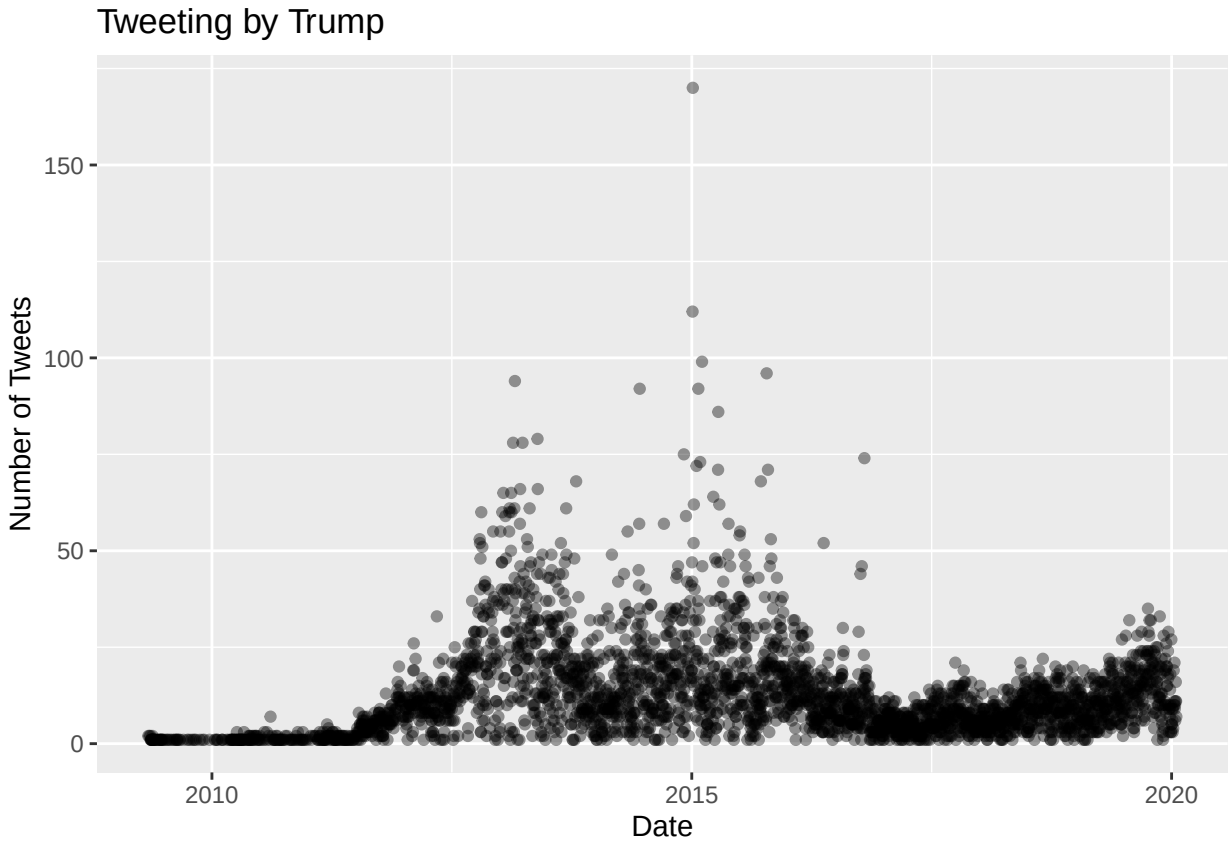
Note that we are going to compare pre-post presidency but you have the tools, and also the code based on what we do today, to do much finer-grained analyses. You can compare how the behavior changes each year. Or each month? Or in response to his impeachments. Or how his activity in the 2016 campaign compares to his activity in his 2020 campaign. Or dive into the 2016 campaign and see how he acted and reacted through the Republican nomination during his rise to the Republican nomination. There are many, many fascinating (and largely unanswered) questions that you should be empowered to be able to do based on what we cover! We will dive deeper a few times, but largely to illustrate the amazing stuff that you can do.

So let's get to it!

```
tweets <- readRDS(file="Trumptweets.Rds")
```

So let's start by describing our data graphically. How frequently was President Trump tweeting throughout the time series we possess?

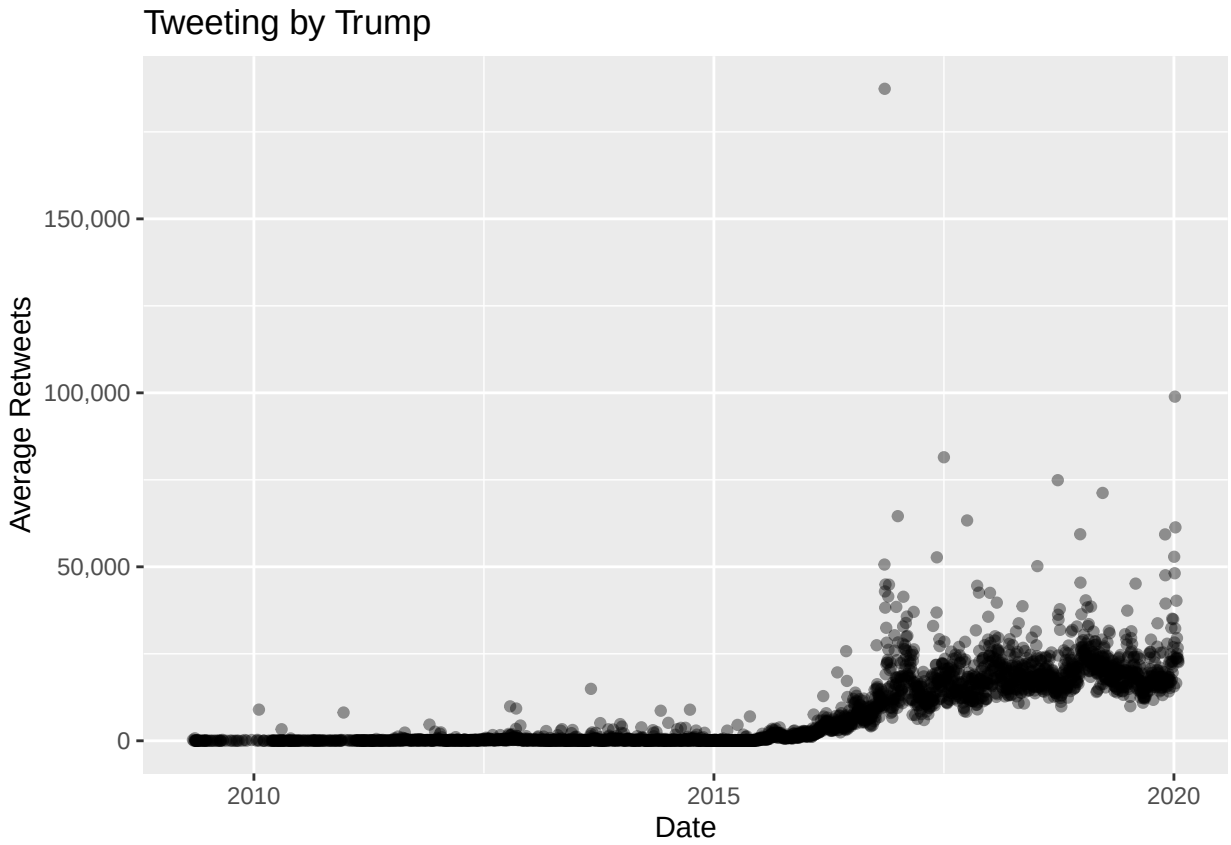
```
tweets %>%
  group_by(Tweeting.date) %>%
  count() %>%
  ggplot() +
  geom_point(aes(x=Tweeting.date,y=n),alpha=.4) +
  labs(x="Date",y="Number of Tweets",title="Tweeting by Trump")
```



Here each point is the number of tweets in a day. Some days there was clearly a lot of activity. Moreover we can see that Trump was more active before becoming President and his frequency of tweeing increased over his presidency.

We can also consider how did the number of retweets, on average, change across days on which a tweet occurred? (Here we are using the `scales` library to set the `scale_y_continuous` to label numbers with commas (`label=comma`).)

```
tweets %>%
  group_by(Tweeting.date) %>%
  summarize(AvgRetweet = mean(retweets)) %>%
  ggplot() +
  geom_point(aes(x=Tweeting.date,y=AvgRetweet),alpha=.4) +
  labs(x="Date",y="Average Retweets",title="Tweeting by Trump") +
  scale_y_continuous(label=comma)
```



Clearly there is a lot of variation here. Which tweets were re-tweeted the most?

```
tweets %>%
  select(content,retweets) %>%
  top_n(retweets,n=10) %>%
  arrange(-retweets)
```

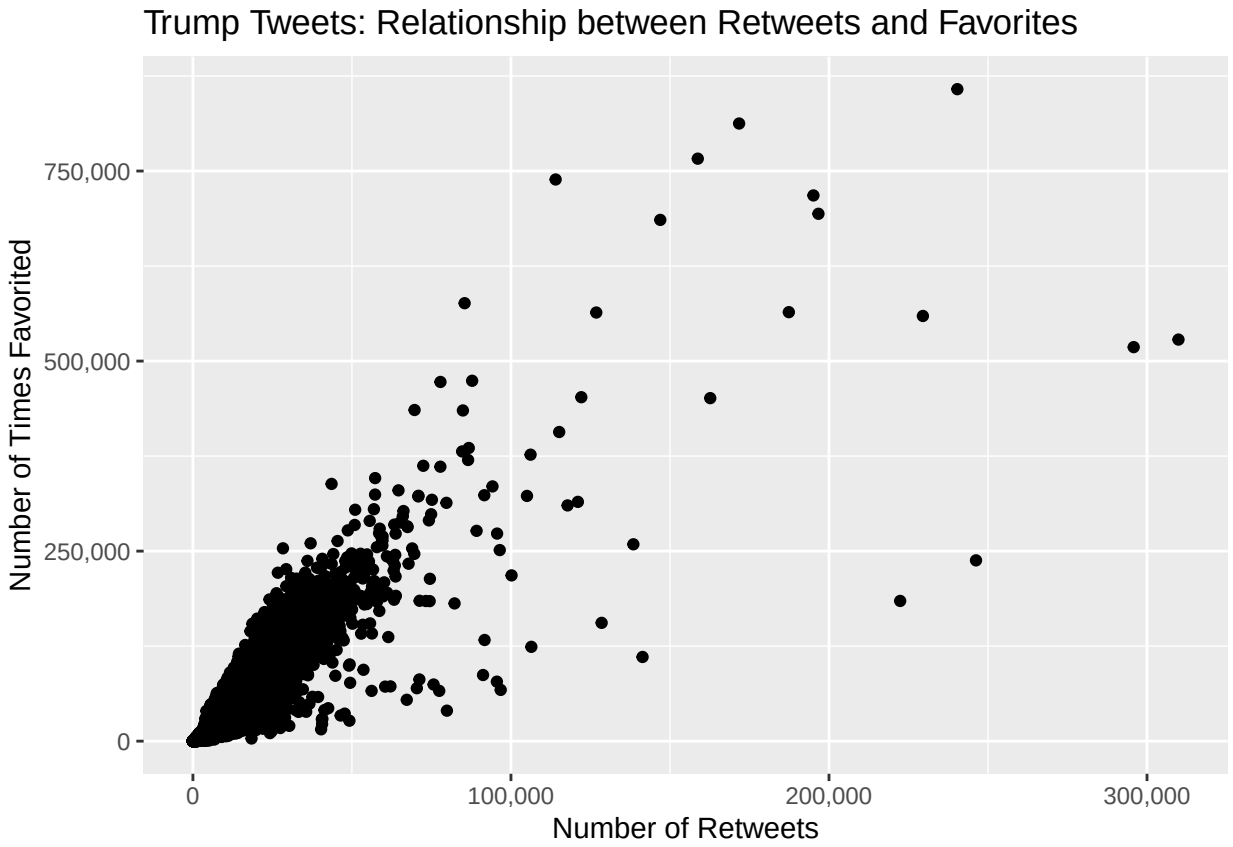
```
## # A tibble: 10 x 2
##    content                                                          retweets
##    <chr>                                                               <dbl>
##  1 "# FraudNewsCNN # FNNpic.twitter.com/WYUnHjjUjg"                   309892
##  2 "TODAY WE MAKE AMERICA GREAT AGAIN!"                               295817
##  3 "Are you allowed to impeach a president for gross incompetence?"   246232
##  4 "A$AP Rocky released from prison and on his way home to the United ~   240363
##  5 "Why would Kim Jong-un insult me by calling me \"old,\" when I woul~   229531
##  6 "Be prepared, there is a small chance that our horrendous leadershi~   222385
##  7 "pic.twitter.com/11nzKwOCtU"                                       196687
##  8 "Just spoke to @ KanyeWest about his friend A$AP Rocky's incarcerat~   195109
##  9 "Such a beautiful and important evening! The forgotten man and woma~   187379
## 10 "pic.twitter.com/VXeKiVzpTf"                                       171742
```

Now can you do the same to identify the tweets that were selected as a favorite? How does this list compare to the tweets that were most likely to be retweeted? Can you write this code?

```
# INSERT CODE HERE
```

In general, how should we measure influence/impact? Favorites or retweets? Does the choice matter? Let's look at the relationship to see how consequential the determination is.

```
tweets %>%
  ggplot(aes(x=retweets, y=favorites)) +
  geom_point() +
  scale_x_continuous(label=comma) +
  scale_y_continuous(label=comma) +
  labs(x= "Number of Retweets", y = "Number of Times Favorited",title="Trump Tweets: Relationship betwe
```



In general they seem pretty related, but there are a handful of tweets that are retweeted far more than they are favorited. (On your own, can you figure out which ones these are?)

We can also use `plotly` to create an HTML object with each point denoting how many retweets each tweet receieved and the date of the tweet. We can use this to label the tweets to get a better sense of what tweets were most re-tweeted? (This will be a very large plot given the number of tweets and the length of the content being pasted into the object! To keep things manageable let's focus on the top 75th-percentile of tweets.)

```
library(plotly)
gg <- tweets %>%
  filter(retweets > quantile(retweets,.75)) %>%
  ggplot(aes(x=Tweeting.date,y=retweets,text=paste(content))) +
  geom_point(,alpha=.4) +
  labs(x="Date",y="Retweets",title="Tweeting by Trump") +
  scale_y_continuous(label=comma)
```

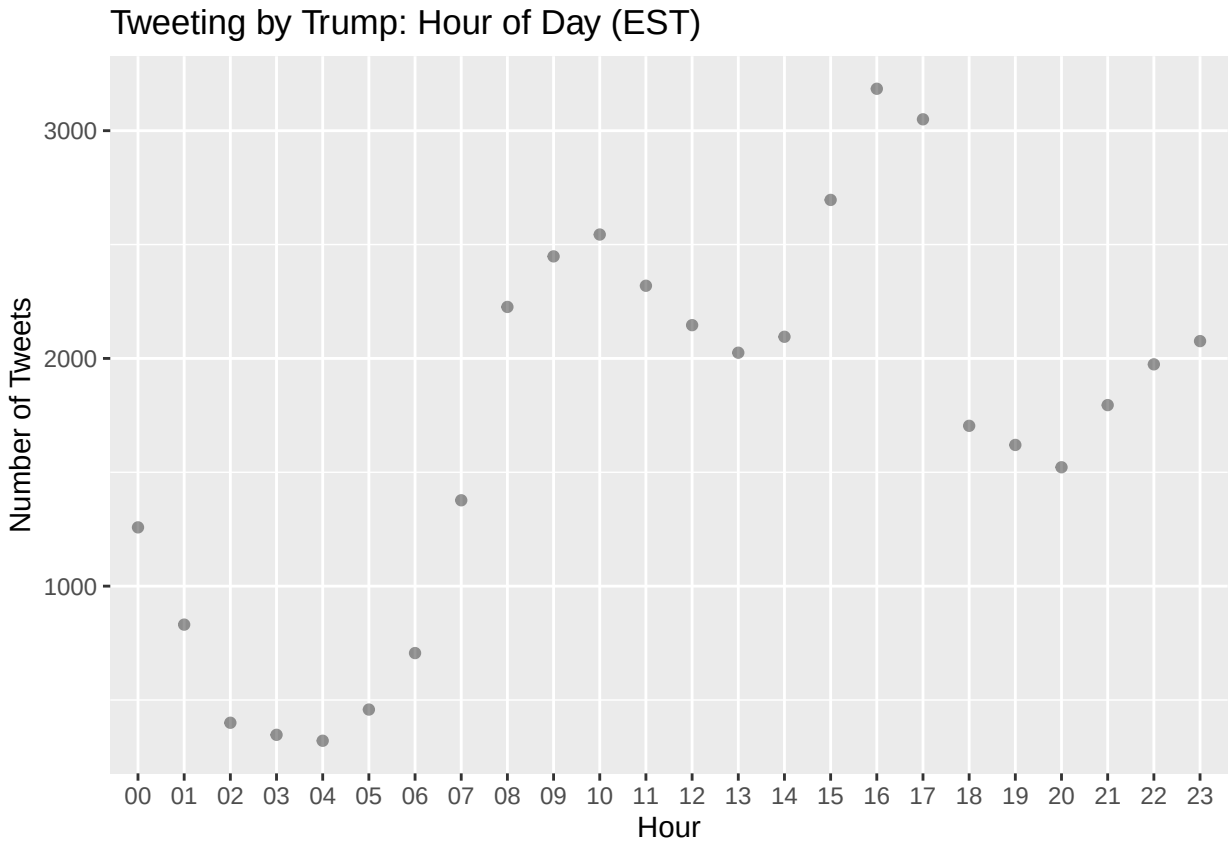
```
ggplotly(gg,tooltip = "text")
```

On your own, can you do the same for the most favorited tweets?

```
# INSERT CODE HERE
```

In addition to looking at the change over time we can also look at the hour at which Trump was tweeting using the `hour` variable we created. To start let's do the total number of tweets at each hour across the entire corpus.

```
tweets %>%
  group_by(Tweeting.hour) %>%
  count() %>%
  ggplot() +
  geom_point(aes(x=Tweeting.hour,y=n),alpha=.4) +
  labs(x="Hour",y="Number of Tweets",title="Tweeting by Trump: Hour of Day (EST)")
```

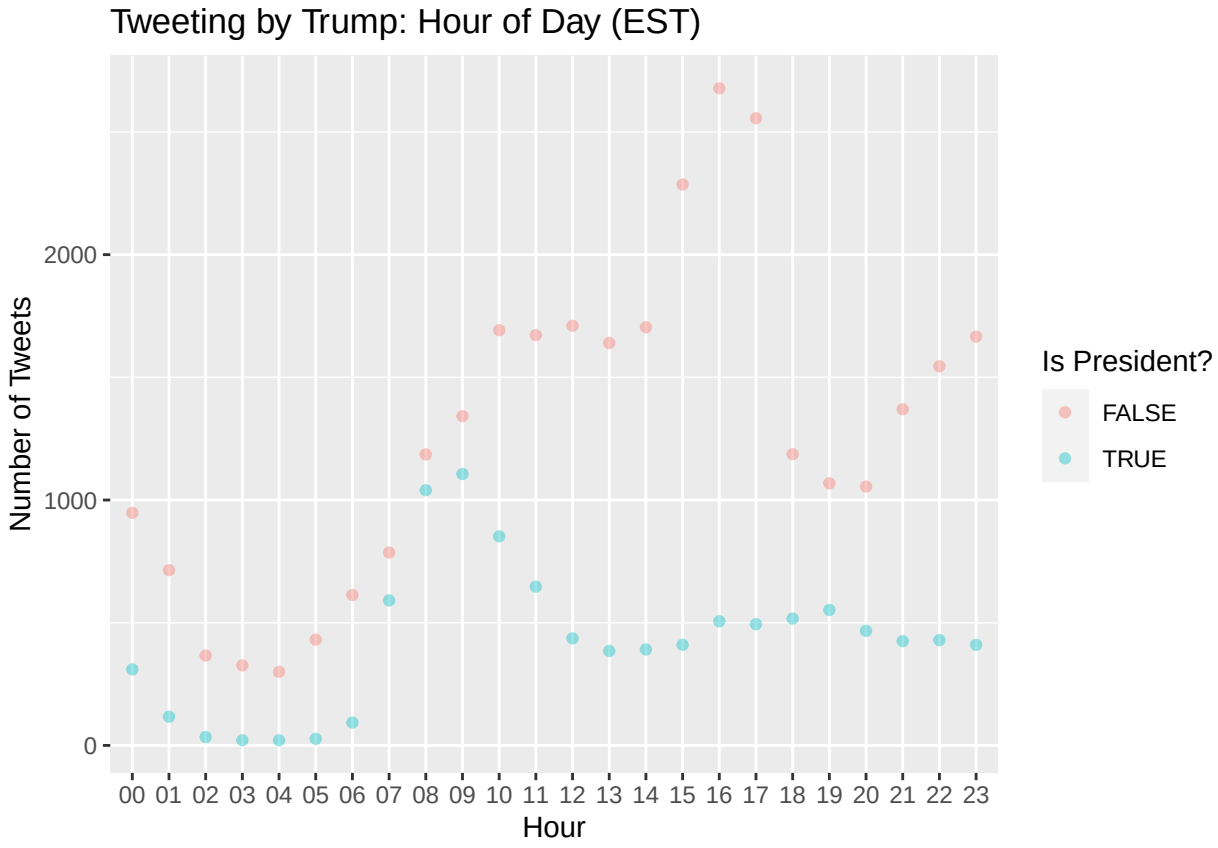


Did Trump's use to twitter change after he was elected President? Certainly we would think that the content might change – as well as how likely it was to be favorited and retweeted – but how about the frequency and timing of the tweets?

Let's create an indicator variable `PostPresident` using the `date` variable to define whether the date is before (`FALSE`) or after (`TRUE`) his election as president (we could also use the inauguration date, but some claimed that his behavior would change once he was officially projected.)

```
tweets %>%
  mutate(PostPresident = date > "2016-11-03") %>%
  group_by(PostPresident,Tweeting.hour) %>%
  count() %>%
```

```
  ggplot() +
  geom_point(aes(x=Tweeting.hour,y=n,color=PostPresident),alpha=.4) +
  labs(x="Hour",y="Number of Tweets",title="Tweeting by Trump: Hour of Day (EST)",color="Is President?")
```

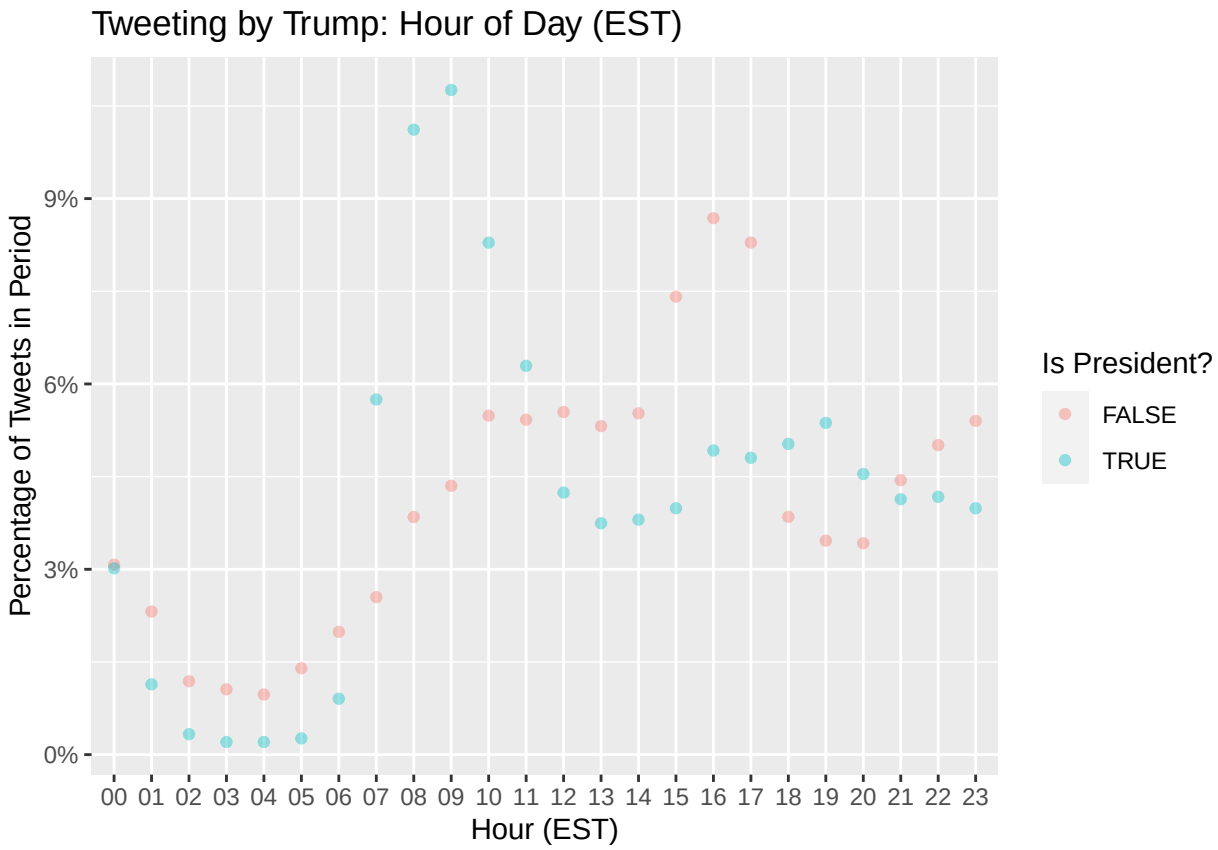


What do you observe?

But is it right to use the overall frequency? Do we prefer the proportion of tweets that were set at each hour pre/post presidency?

Let's use `mutate` to compute the proportion of tweets that occur at each hour in each period and then plot those using `color` to denote the pre/post time period.

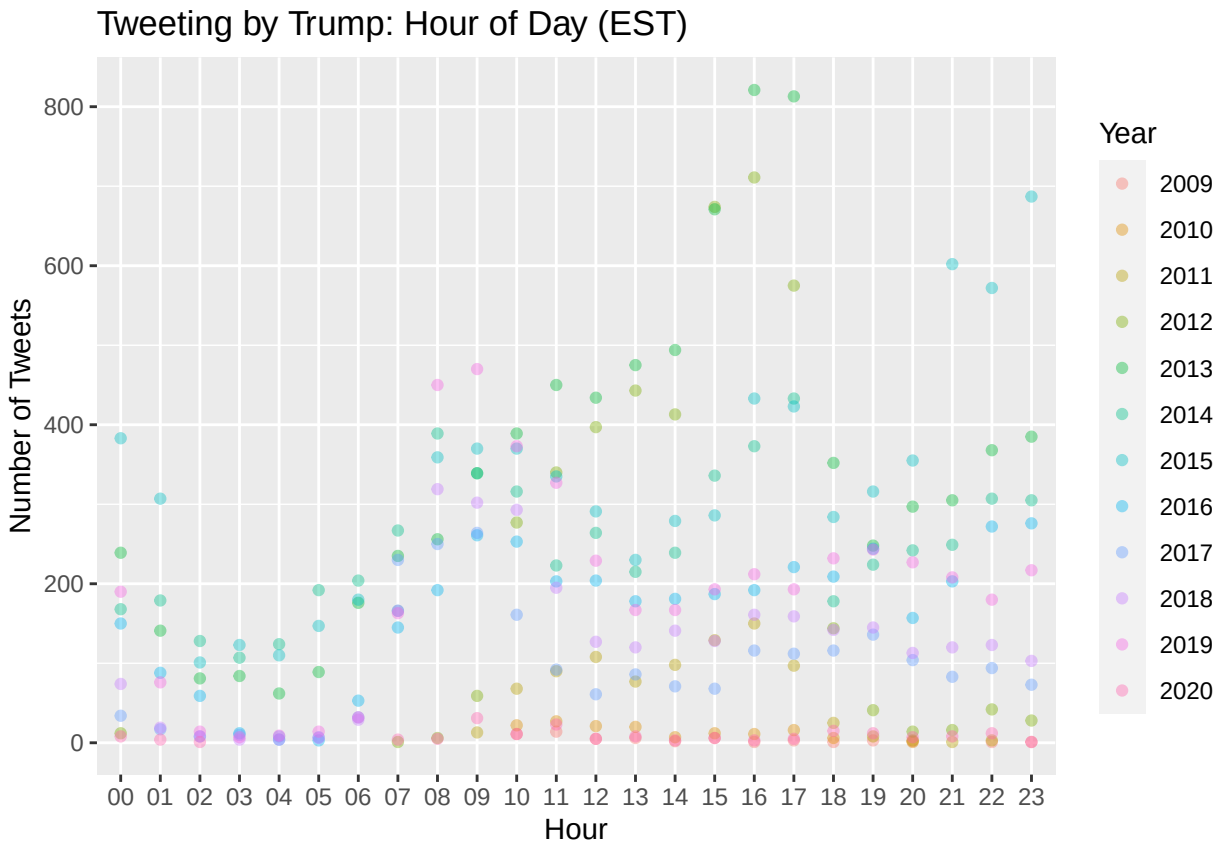
```
tweets %>%
  mutate(PostPresident = date > "2016-11-03") %>%
  group_by(PostPresident,Tweeting.hour) %>%
  count() %>%
  ungroup(Tweeting.hour) %>%
  mutate(Prop = n/sum(n)) %>%
  ggplot() +
  geom_point(aes(x=Tweeting.hour,y=Prop,color=PostPresident),alpha=.4) +
  labs(x="Hour (EST)",y="Percentage of Tweets in Period",title="Tweeting by Trump: Hour of Day (EST)",c
  scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



Hmm. A very different characterization! Always think about what the right calculation is. R will do what you tell it – usually ;) – but you need to think about what you are asking it to do!

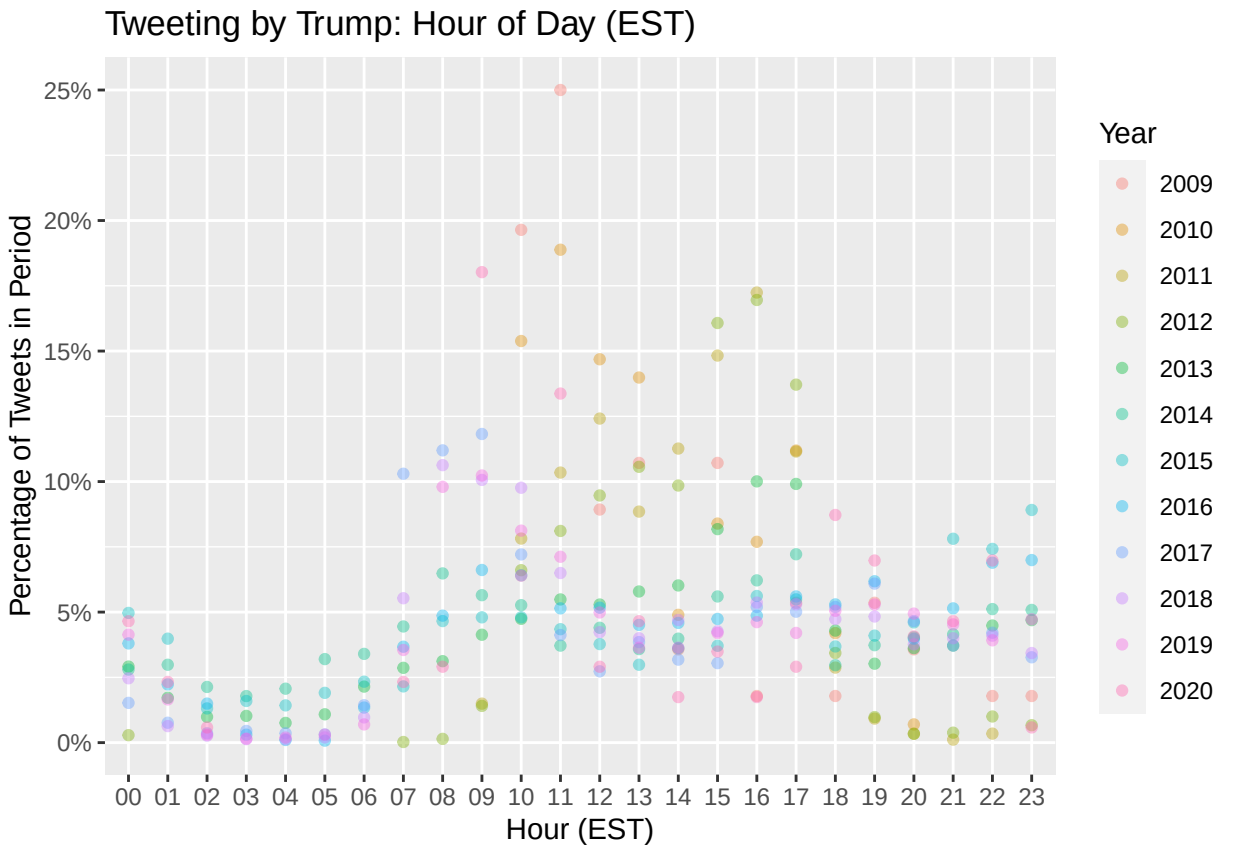
We could also go deeper and look at variation by year. Here is a graph of the overall frequency. (Can you do the same for proportions?)

```
tweets %>%
  group_by(Tweeting.year,Tweeting.hour) %>%
  count() %>%
  ggplot() +
  geom_point(aes(x=Tweeting.hour,y=n,color=Tweeting.year),alpha=.4) +
  labs(x="Hour",y="Number of Tweets",title="Tweeting by Trump: Hour of Day (EST)",color="Year")
```



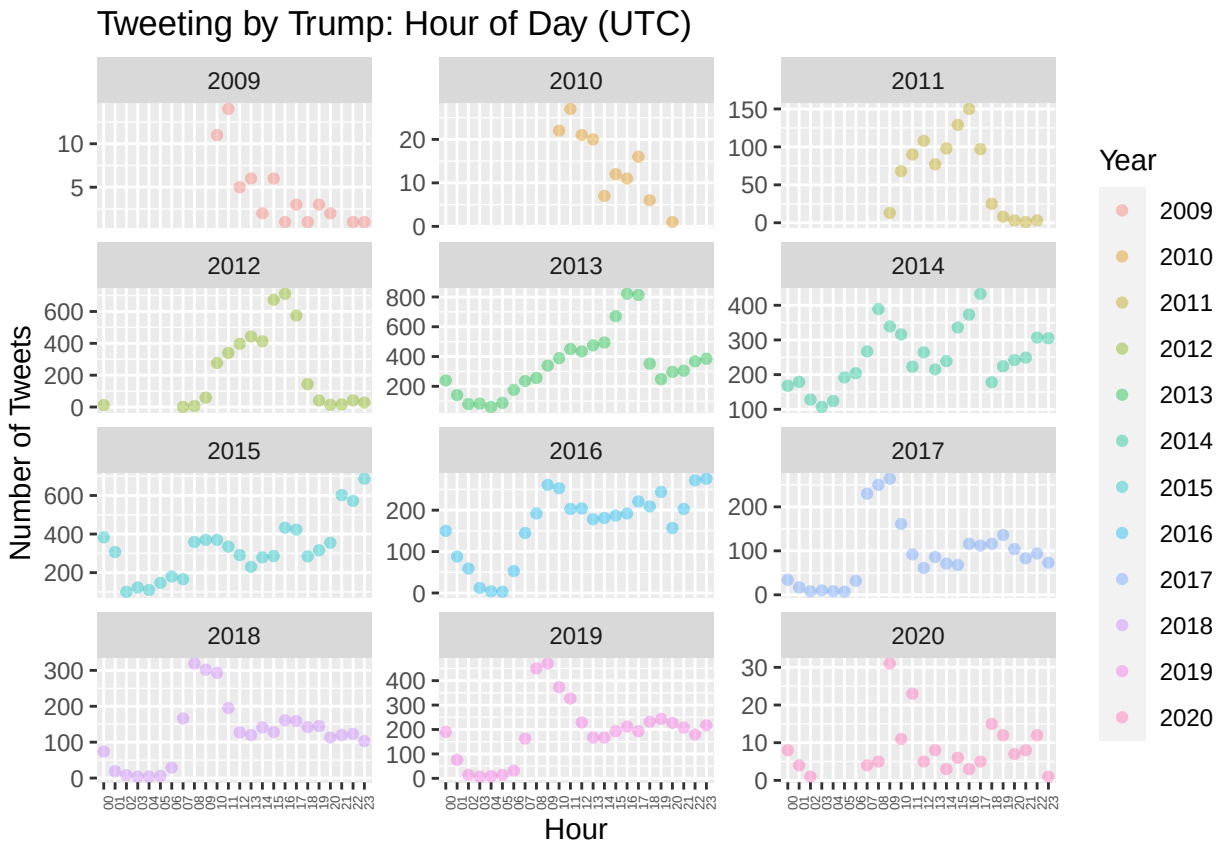
Can you graph the proportion of tweets per hour per year? How does that change your characterization. In you opinion, which is the more appropriate measure? The number of tweets or the proportion of tweets? Why or why not?

```
tweets %>%
  group_by(Tweeting.year,Tweeting.hour) %>%
  count() %>%
  ungroup(Tweeting.hour) %>%
  mutate(Prop = n/sum(n)) %>%
  ggplot() +
  geom_point(aes(x=Tweeting.hour,y=Prop,color=Tweeting.year),alpha=.4) +
  labs(x="Hour (EST)",y="Percentage of Tweets in Period",title="Tweeting by Trump: Hour of Day (EST)",c
  scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



Here we put everything on the same graph, but maybe it makes sense to create separate graphs - one for each value that we are using to define the `color`. To do this we just need to use a `facet_wrap` to create a buynch of graphs that will "wrap around" the screen starting from the lowest value of the facet defined after the `~` and arranged in a grid with `nrow=4` (Try different values here!). We have defined `scales = "free_y"` to let the graphs vary in the scales of the y-axis (because the frequencies vary). We could also choose `"fixed"` to give every graph the same x and y limits, or `"free_x"` to use the same y-axis scale and allow the x-axis to vary. `scale="free"` allows both the x and y axes to vary. Experiment with what happens if you change the scale. Why did I do what we did?

```
tweets %>%
  group_by(Tweeting.year,Tweeting.hour) %>%
  count() %>%
  ggplot() +
  facet_wrap(~ Tweeting.year, scales = "free_y", nrow = 4) +
  geom_point(aes(x=Tweeting.hour,y=n,color=Tweeting.year),alpha=.4) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size=5)) +
  labs(x="Hour",y="Number of Tweets",title="Tweeting by Trump: Hour of Day (UTC)",color="Year")
```



First try to do the same thing with the tweeting behavior Pre/Post Presidency. Can you use the `facet_wrap` to create that visualization? Which visualization do you prefer?

```
# INSERT CODE HERE
```

Now see if you can pull this together by plotting the time of tweeting by year but graphing the proportions this time. How should we define the `scales` in this case?

```
# INSERT CODE HERE
```

[OPTIONAL] Data Wrangling Text

You can skip this section and just load in the data in the next section, but this is if you want to know what I did to get the data from above ready for the analyses we do below. It invovles working with the string variable `content` and deleting elements of that string to leave us only word "tokens."

OK so that is what we can do at the level of the tweet. To analyze the content of the tweet we need to transform the string of each tweet into "tokens" (words) that we can then analyze. The first part is often the hardest – data-wrangling is typically 85% of the issue in data science. Rather than give you the cleaned data, here is a sense of what you need to do to make it work. Do not worry about understanding the content at this point.

The following is included for the interested student to get a sense of what is required to convert the `content` into tokens. Recall that our "data" looks like this:

```
tweets$content[1]
```

```
## [1] "Be sure to tune in and watch Donald Trump on Late Night with David Letterman as he presents the
```

And we need to transform that into something we can analyse! This takes some work...

```
library(qdapRegex)
library(tm)
library(tidytext)
library(SnowballC)
```

First we are going to strip out all of the url and twitter-formatted url from the tweet text using some pre-defined functions.

```
tweets <- tweets %>%
  mutate(content = rm_twitter_url(content),
         content = rm_url(content),
         document = id)
```

Now we are going to write a function that takes as an argument a string (`x`) and then uses multiple functions to remove strings satisfying certain conditions.

First we are going to process the string `content` to remove combinations of letters/numbers that are not words. To do so we are going to define a function called `clean_tweets` and then apply it to the `content` variable in `tweets` tibble.

```
clean_tweets <- function(x) {
  x %>%
    # Remove mentions e.g. "@my_account"
    str_remove_all("@[[:alnum:]_]{4,}") %>%
    # Remove mentions e.g. "@ my_account"
    str_remove_all("@ [[:alnum:]_]{4,}") %>%
    # Remove hashtags
    str_remove_all("#[[:alnum:]_]+") %>%
    # Remove hashtags
    str_remove_all("# [[:alnum:]_]+") %>%
    # Remove twitter references
    str_remove_all("twitter[[:alnum:]_]+") %>%
    # Remove twitter pics references
    str_remove_all("pictwitter[[:alnum:]_]+") %>%
    # Replace "&" character reference with "and"
    str_replace_all("&amp;", "and") %>%
    # Remove punctuation, using a standard character class
    str_remove_all("[[:punct:]]") %>%
    # Remove "RT: " from beginning of retweets
    str_remove_all("^RT:? ") %>%
    # Replace any newline characters with a space
    str_replace_all("\\\n", " ") %>%
    # Make everything lowercase
    str_to_lower() %>%
    # Remove any trailing whitespace around the text
    str_trim("both")
}

# Now apply this function to the `content` of `tweets`
tweets$content <- clean_tweets(tweets$content)
```

Now that we have pre-processed the `content` string lets do some more more wrangling to extract word "tokens" from this string and then also remove the tokens that are not meaningful words. Let's also define the object `reg` containing all the various characters that an be used.

```
reg <- "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"

tweet_words <- tweets %>%
  filter(!str_detect(content, '^"')) %>%
  unnest_tokens(word, content, token = "regex", pattern = reg)  %>%
  filter(!word %in% stop_words$word,str_detect(word, "[a-z]")) %>%
  mutate(word = str_replace_all(word, "\\d+", "")) %>%    # drop numbers
  mutate(word = str_replace_all(word, "twitter[A-Za-z\\d]+|&amp;", "")) %>%
  mutate(word = str_replace_all(word, "pictwitter[A-Za-z\\d]+|&amp;", "")) %>%
  mutate(word = str_replace_all(word, "pic[A-Za-z\\d]+|&amp;", "")) %>%
  mutate(word = str_replace_all(word, "pic", "")) %>%
  mutate(word = str_replace_all(word, "againpic[A-Za-z\\d]+|&amp;", "")) %>%
 # mutate(word = wordStem(word)) %>%
  mutate(document = id) %>%
  select(-id) %>%
  filter(word != "")   # drop any empty strings
```

Now use the `anti_join` to remove all `stop_words` to focus on the words with "content."

```
data("stop_words", package = "tidytext")
tweet_words <- anti_join(tweet_words, stop_words, by = "word")
```

And save this for future use.

```
save(tweet_words,file="data/Trump_tweet_words.Rda")
```

Back to Reality

We can also just read in the already-wrangled data `tweet_words` and proceed from here.

```
tweet_words <- readRDS(file="Trump_tweet_words.Rds")
```

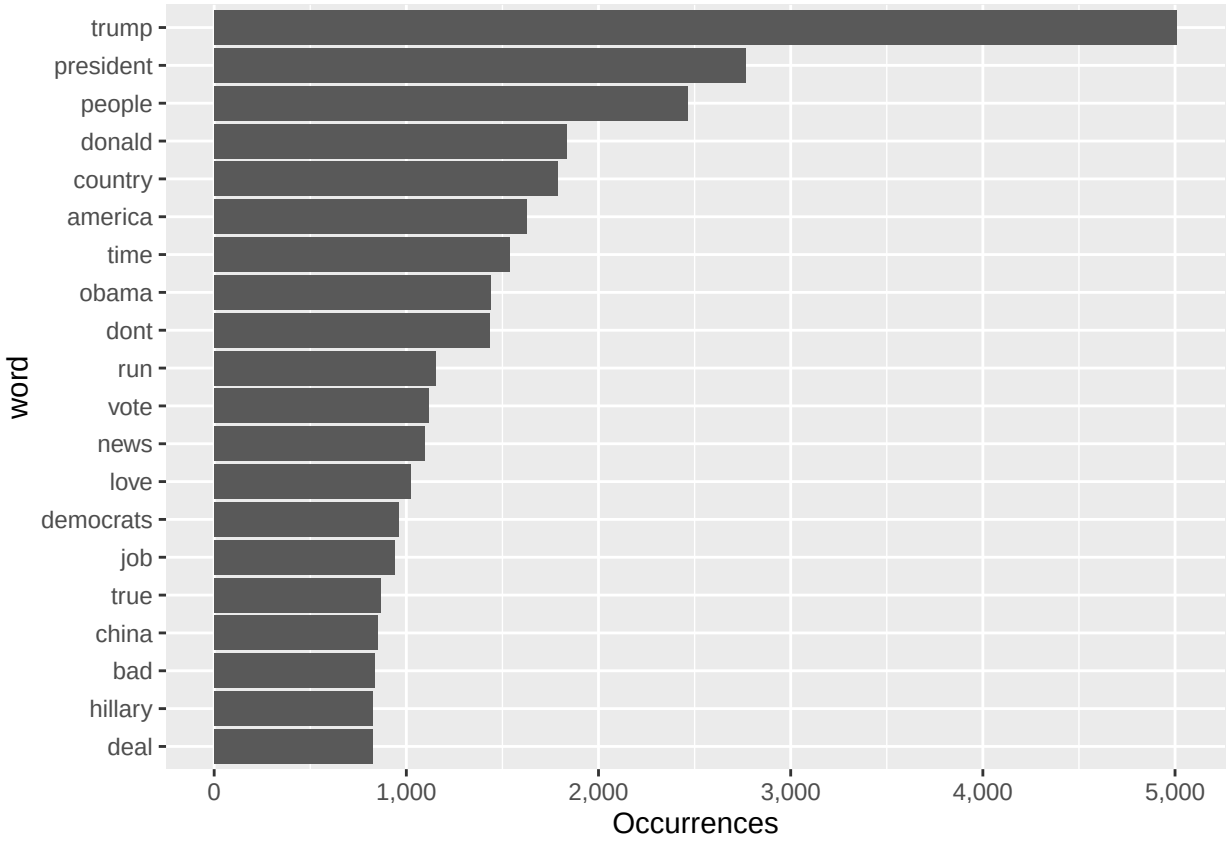
So what are the most commonly tweeted word stems?

```
tweet_words %>%
  count(word) %>%
  arrange(-n)
```

```
## # A tibble: 23,453 x 2
##    word          n
##    <chr>     <int>
##  1 trump      5010
##  2 president  2766
##  3 people     2465
##  4 donald     1833
##  5 country    1788
##  6 america    1627
##  7 time       1540
##  8 obama      1440
##  9 dont       1434
## 10 run        1152
## # ... with 23,443 more rows
```

Let's plot the 20 most frequently used words in descending order using a barplot.

```
tweet_words %>%
  count(word, sort = TRUE) %>%
  head(20) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat = "identity") +
  ylab("Occurrences") +
  scale_y_continuous(label=comma) +
  coord_flip()
```



Interesting.But we want to know how twitter use changed, if at all, over time and in response to being elected president. So let's start by defining a variation that is `PostPresident` defined to be tweets after being projected as the winner of the 2016 Presidential election.

NOTE: You could also look at other variation (e.g., years, pre/post certain events, etc.). There are lots of opportunities to expand/refine this! Try some!

```
tweet_words <- tweet_words %>%
  mutate(PostPresident = Tweeting.date > "2016-11-03")
```

To compare lets compare the top 10 word stems that were tweeted pre-presidency.

```
tweet_words %>%
    filter(PostPresident == FALSE) %>%
    select(word) %>%
    count(word) %>%
    top_n(10, wt= n) %>%
    arrange(-n)
```

```
## # A tibble: 10 x 2
##    word          n
##    <chr>     <int>
##  1 trump      4303
##  2 president  1790
##  3 donald     1703
##  4 people     1286
##  5 obama      1206
##  6 america    1102
##  7 run        1055
##  8 dont       1012
##  9 time        976
## 10 country     954
```

And the top 10 stems tweeted post-presidency.

```
tweet_words %>%
  filter(PostPresident == TRUE) %>%
  select(word) %>%
  count(word) %>%
  top_n(10, wt= n) %>%
  arrange(-n)
```

```
## # A tibble: 10 x 2
##    word          n
##    <chr>     <int>
##  1 people     1179
##  2 president   976
##  3 democrats   867
##  4 country     834
##  5 news        806
##  6 fake        724
##  7 trump       707
##  8 border      614
##  9 time        564
## 10 america     525
```

Putting together in a nicer table using `group_by`.

```
tweet_words %>%
  group_by(PostPresident) %>%
  select(word) %>%
  count(word) %>%
  top_n(10, wt= n) %>%
  arrange(-n)   %>%
  summarise(top_words = str_c(word, collapse = ", "))
```

```
## Adding missing grouping variables: `PostPresident`
```

```
## # A tibble: 2 x 2
##   PostPresident top_words
##   <lgl>         <chr>
## 1 FALSE         trump, president, donald, people, obama, america, run, dont, ti~
## 2 TRUE          people, president, democrats, country, news, fake, trump, borde~
```

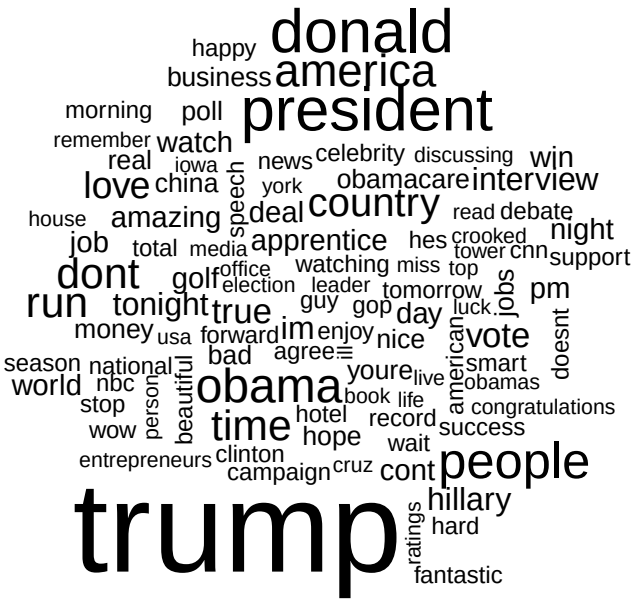
And now graphing them using a `wordcloud`. (Why are we setting a seed?)

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
set.seed(42)

tweet_words %>%
  filter(PostPresident == FALSE) %>%
  select(word) %>%
  count(word) %>%
  { wordcloud(.$word, .$n, max.words = 100) }
```




```
tweet_words %>%
  filter(PostPresident == TRUE) %>%
  select(word) %>%
  count(word) %>%
  { wordcloud(.$word, .$n, max.words = 100) }
```

```
## Warning in wordcloud(.$word, .$n, max.words = 100): people could not be fit on
## page. It will not be plotted.
```

```
## Warning in wordcloud(.$word, .$n, max.words = 100): democrats could not be fit
## on page. It will not be plotted.
```

```
## Warning in wordcloud(.$word, .$n, max.words = 100): president could not be fit
## on page. It will not be plotted.
```

```
## Warning in wordcloud(.$word, .$n, max.words = 100): border could not be fit on
## page. It will not be plotted.
```

But what about variation over time? Lots of events happened every year and looking at all tweets before 2016 compared to all of the tweets after Election Day 2016 may lose important nuance and variation. So let's look at the 10 most frequently tweeted words each year.

```
tweet_words %>%
  group_by(Tweeting.year) %>%
  select(word) %>%
  count(word) %>%
  top_n(10, wt= n) %>%
  arrange(-n)   %>%
  summarise(top_words = str_c(word, collapse = ", ")) %>%
  knitr::kable()
```

```
## Adding missing grouping variables: `Tweeting.year`
```

Tweeting.year	top_words
2009	donald, trump, champion, trumps, book, watch, contest, david, dont, enter, happy, read, signed
2010	pm, apprentice, trump, nbc, miss, tonight, fantastic, night, tune, episode, hotel
2011	cont, interview, china, pm, trump, america, watch, book, debate, discussing, tonight
2012	cont, obama, trump, interview, china, discussing, people, dont, time, president
2013	trump, people, donald, obama, president, true, dont, time, love, country
2014	trump, president, donald, run, obama, country, love, true, vote, dont
2015	trump, donald, president, america, run, people, country, love, time, dont
2016	hillary, trump, people, clinton, america, crooked, cruz, join, vote, pm
2017	people, news, fake, america, tax, country, president, jobs, american, media
2018	people, country, president, democrats, border, trump, news, fake, trade, time
2019	president, people, democrats, country, news, fake, trump, border, time, united

Tweeting.year	top_words
2020	iran, impeachment, democrats, american, house, party, hoax, people, time, president

An now, how about by hour? What is on President Trump's mind, on average, at every hour of the day?

```
# INSERT CODE HERE
```

Pushing ahead, we could also do hour by pre/post presidency (or year) to see how the content changed. Or perhaps how activity varies across parts of the day by creating periods of time based on the hours (e.g., "late-night", "early morning", "normal work-day"). Here is where you as a data-scientist can propose (and defend!) categorizations that are useful for understanding the variation in the data.

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

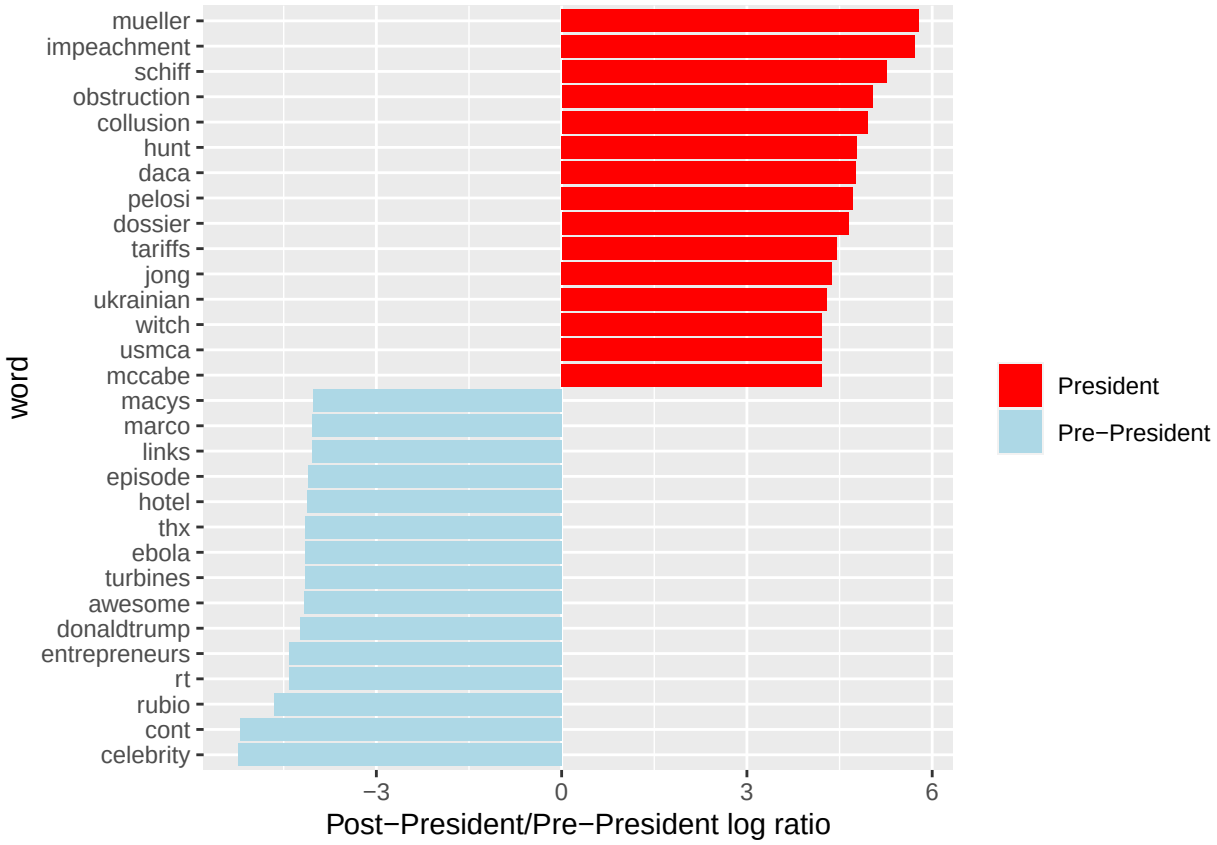
So let's compute the log odds ratio for each word pre and post presidency.

```
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

So 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 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 associates words with 10 sentiments: **positive**, **negative**, **anger**, **anticipation**, **disgust**, **fear**, **joy**, **sadness**, **surprise**, and **trust**.

As an aside, the way the sentiment was created was by showing a bunch of people a series of words and then asking them what emotion they associated with each word. The responses of regular people were then aggregated to determine whether each word was associated with each emotion (by effectively looking at the extent to which individuals associated the same emotion with each word). This is what we mean by "crowd-sourcing" – using people to collect data for us. Note that we could create our own DSCI 1000 sentiment index. All we would need to do is to have each of you indicate the emotion(s) you associate with a series of words. For example, when you see `ggplot` do you feel **positive**, **negative**, **anger**, **anticipation**, **disgust**, **fear**, **joy**, **sadness**, **surprise**, or **trust**? (Multiple emotions are allowed.)

So let's load the NRC sentiment library in and take a look at what it looks like by calling `nrc` to the console.

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

nrc
```

```
## # A tibble: 13,875 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 13,865 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

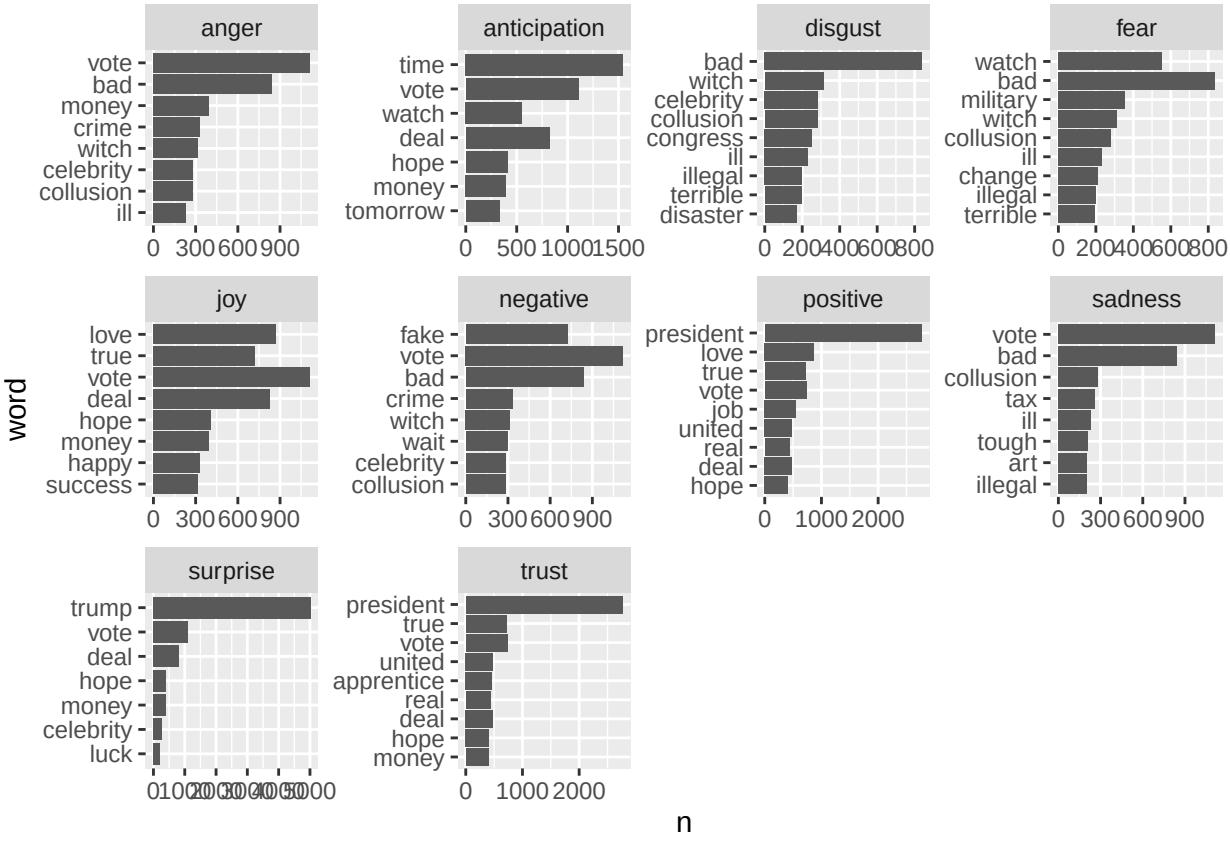
word tokens this would be problematic!

```
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).

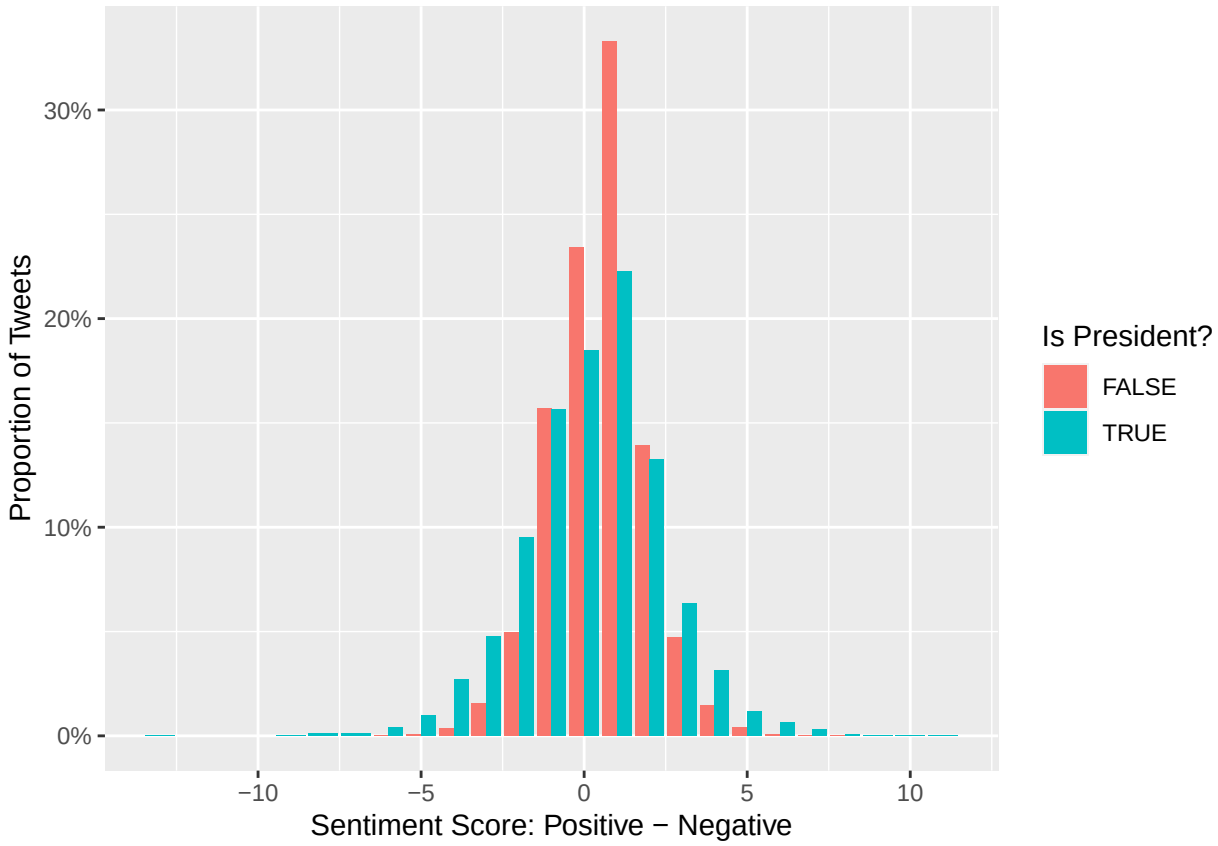
If we wanted to see how many total words were used in the all of the tweets we observed pre-post presidency we can summarize.

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

```
## # A tibble: 2 x 13
##   PostPresident anger anticipation disgust  fear   joy negative positive sadness
##   <lgl>         <int>        <int>   <int> <int> <int>    <int>    <int>   <int>
## 1 FALSE          8319        13624    5503  8194 12635    15242    27956    8151
## 2 TRUE           7115         6652    4892  6845  5366    12651    14783    5552
## # ... with 4 more variables: surprise <int>, trust <int>, sentiment <int>,
## #   ntweet <dbl>
```

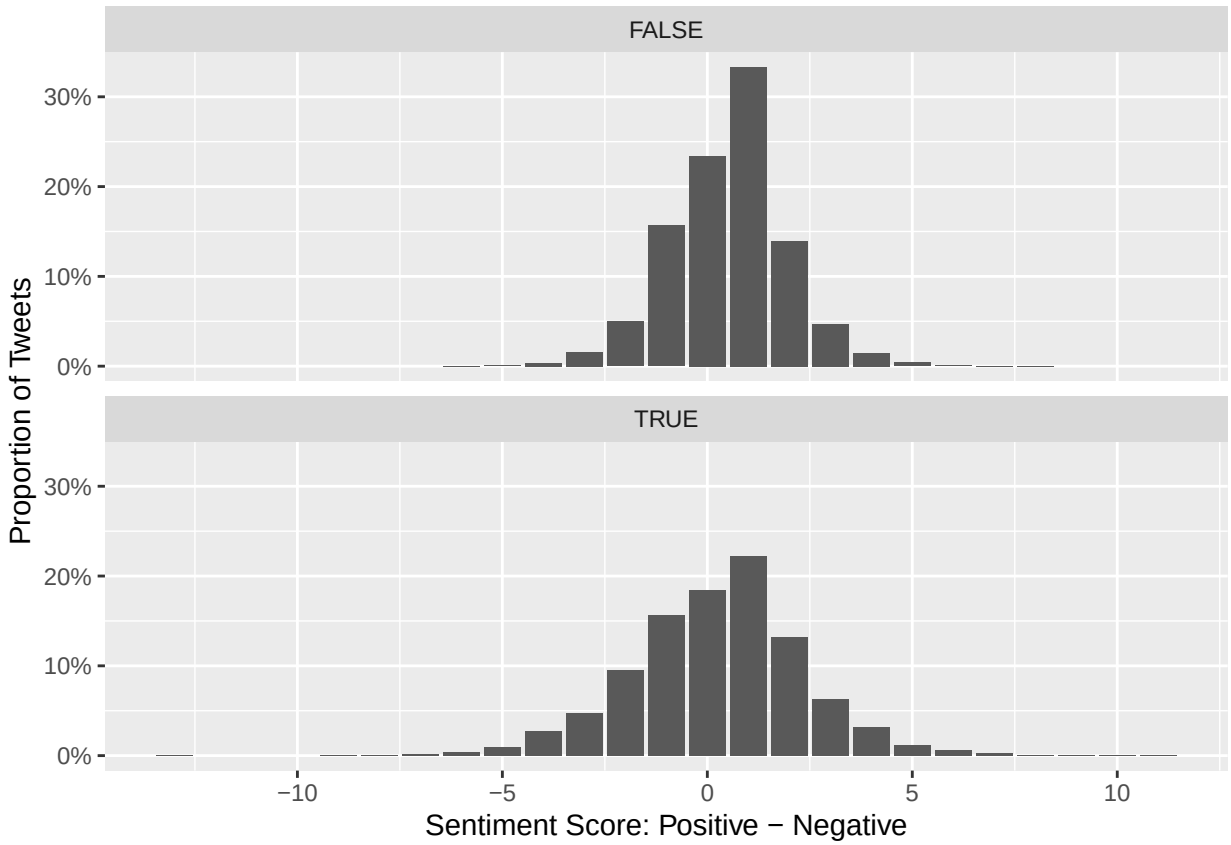
We can plot the distribution of sentiment scores pre/post presidency as follows. Not that the `position` controls where the bars for each `fill` are placed. (Try running it without `position` to see the default.) The `aes` of `geom_bar` is defined so as to plot the proportion rather than the frequency. 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?

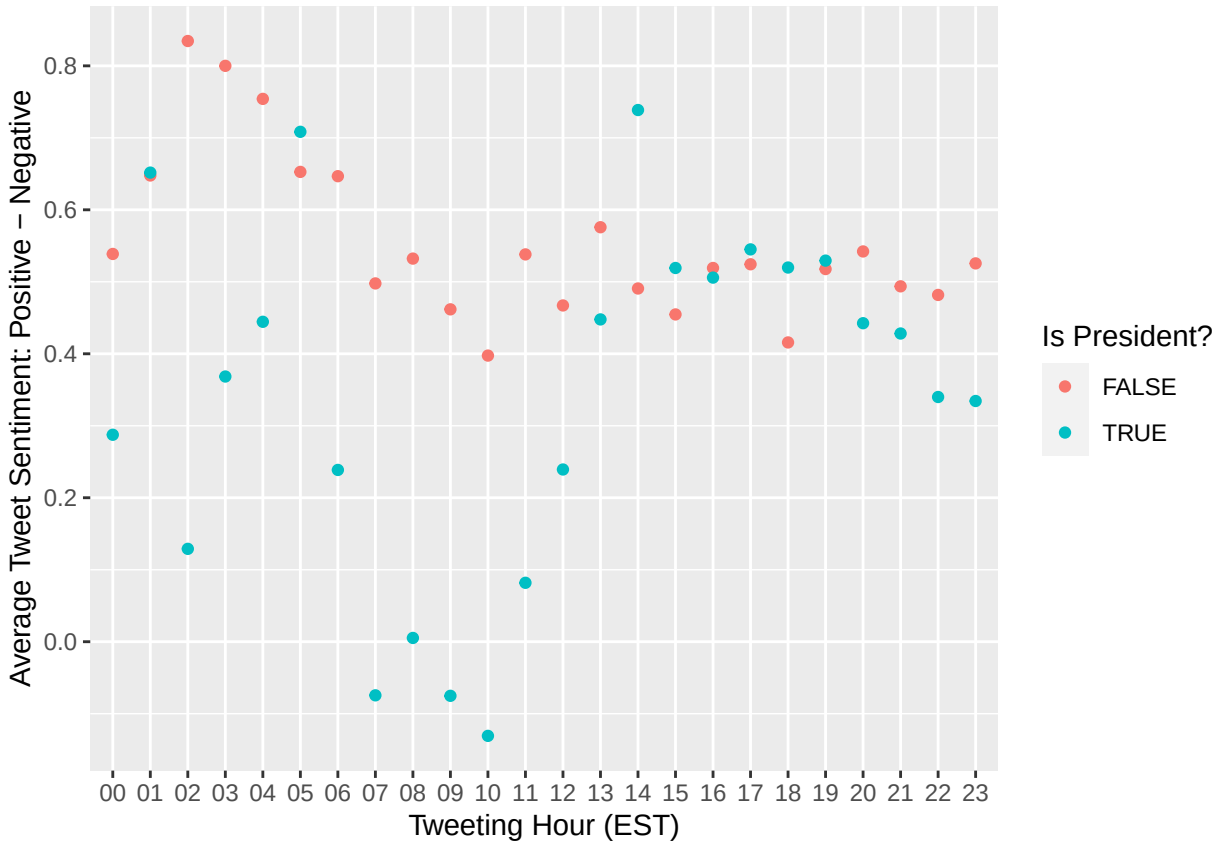
```
tweet_sentiment_summary %>%
  ggplot(aes(x = sentiment)) +
  facet_wrap(~ PostPresident, scales = "fixed", nrow = 2) +
  geom_bar(aes(y = ..prop..)) +
  scale_y_continuous(labels = percent) +
  labs(y= "Proportion of Tweets", x = "Sentiment Score: Positive - Negative")
```



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`.

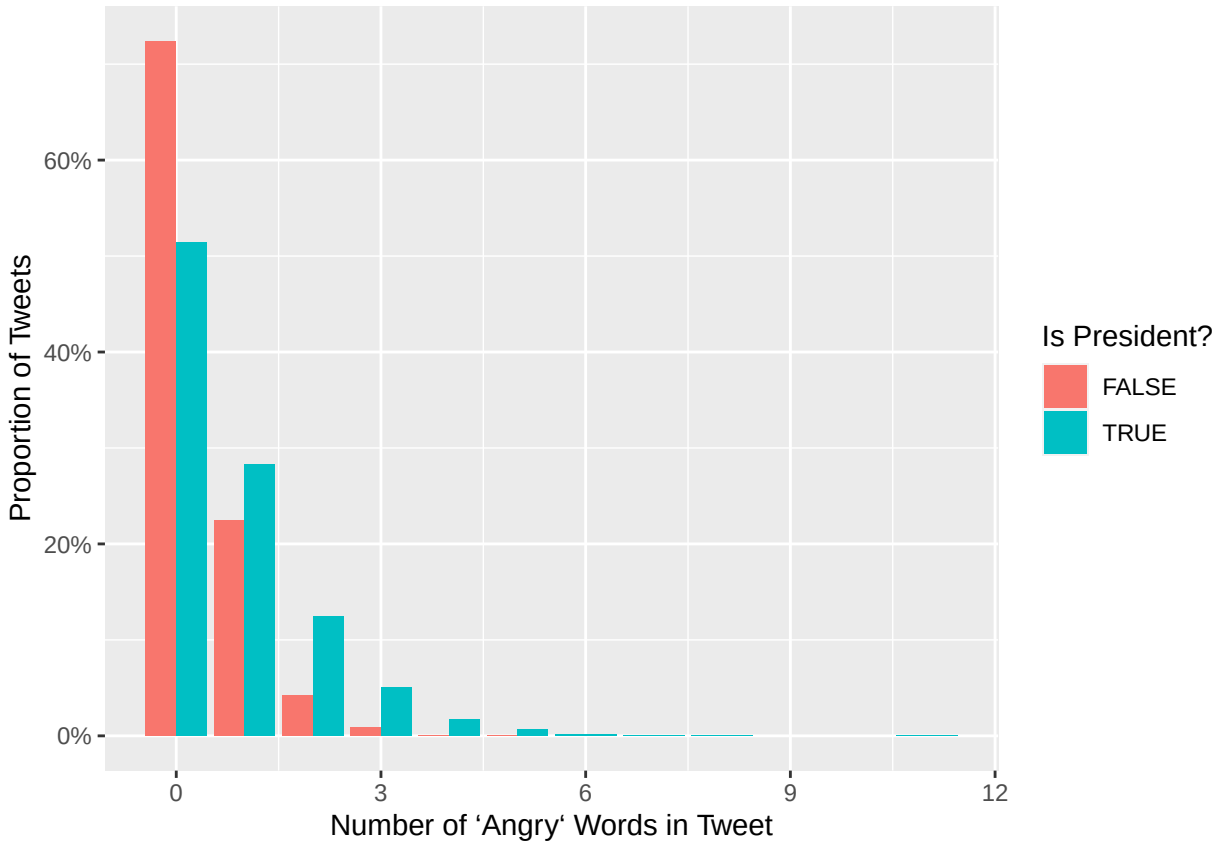
```
tweet_sentiment %>%
  group_by(PostPresident,Tweeting.hour,sentiment) %>%
  count(document,sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  summarize(AvgSentiment = mean(sentiment)) %>%
  ggplot(aes(y = AvgSentiment,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

```
## `summarise()` has grouped output by 'PostPresident'. You can override using the
## `.groups` argument.
```



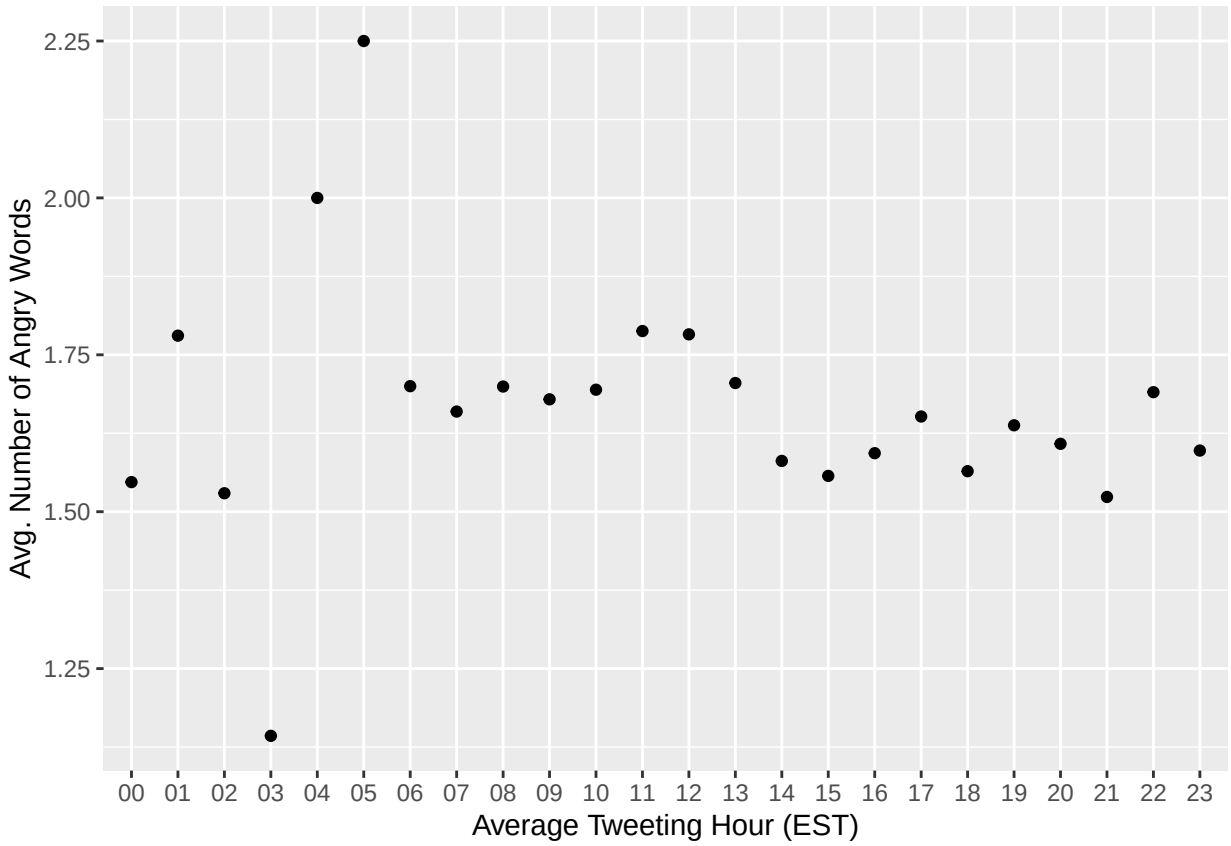
We can also dive in deeper and focus on particular sentiments. What is the distribution of the number of "angry" words being used in each tweet? Was Trump getting "angrier" over time?

```
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?")
```



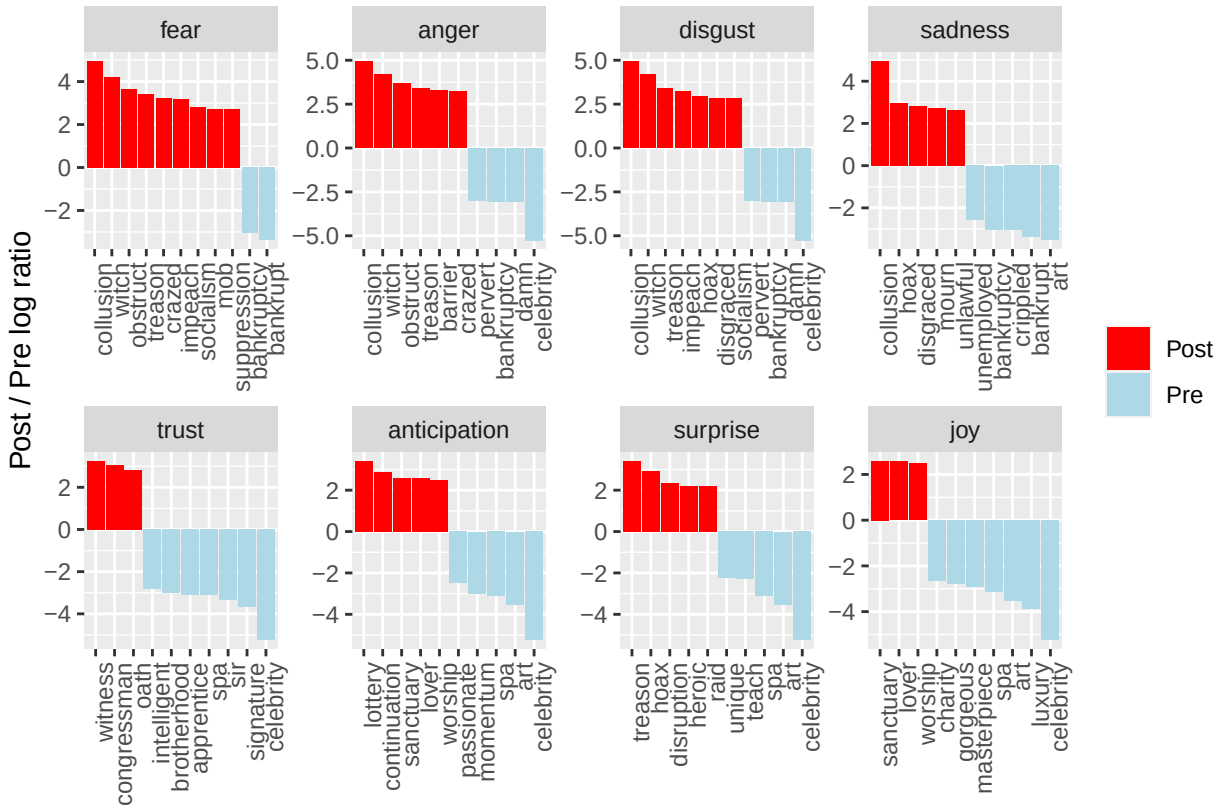
Or angrier based on the time of day post-presidency?

```
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))
```

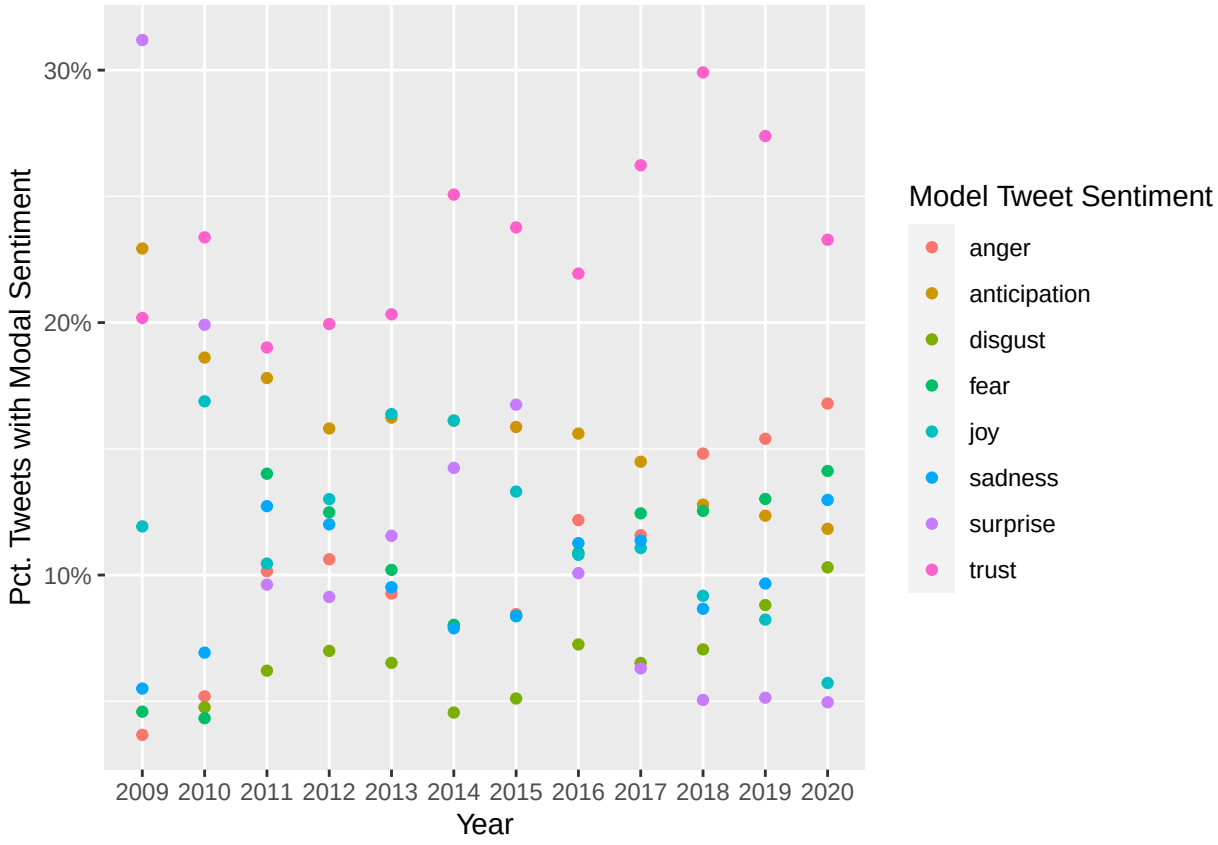
```
## # A tibble: 8 x 3
##   sentiment        n PctSentiment
##   <chr>        <int>        <dbl>
## 1 anger         6569        0.106
## 2 anticipation  9442        0.153
```

```
## 3 disgust        3940       0.0637
## 4 fear           6518       0.105
## 5 joy            7992       0.129
## 6 sadness        5945       0.0961
## 7 surprise       6786       0.110
## 8 trust         14643       0.237
```

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))
```



Describing Twitter content using `kmeans`

Instead of classifying content using a pre-defined dictionary of sentiment we can also return to using `kmeans` to analyze content? Given the large number of documents this can be a challenge (why?). As a result we are going to filter to analyze just tweets in 2016 and using only the tweets that are retweeted more than average. (Of course, it would be interesting to compare the content of tweets that are and are not retweeted (or favorited).)

```
tokens <- tweet_words %>%
  filter(Tweeting.date >= "2016-1-1" & Tweeting.date <= "2016-12-31") %>%
  filter(retweets > quantile(retweets,.5)) %>%
  select(document, word)
```

Now define the document-term-matrix, and cast it into a matrix for analyses using manipulations we have done before. Again, you can examine each object to see how the code is changing the content of our tibbles.

```
dtm <- tokens %>%
  count(document, word)

rm(tokens)

dtm.tfidf <- bind_tf_idf(dtm, word, document, n)
rm(dtm)

castdtm <- cast_dtm(dtm.tfidf, document, word, tf)
rm(dtm.tfidf)
```

How many tweets are we going to analyze?

```
nrow(castdtm)
```

```
## [1] 1730
```

Now let's analyze and uncover the 5 clusters in these tweets.

```
set.seed(42)
km_out.2016 <- kmeans(castdtm,
                  centers = 5,
                  nstart = 25)
```

And now summarize the top 4 words associated with each of the 5 clusters. (Note that because tweets are so sparse, increasing the number of words results in a large number of words that are used with a similar frequency.)

```
table(km_out.2016$cluster)
```

```
##
##    1    2    3    4    5
##   32 1250  110  324   14
```

```
words_kmean <- tibble(word = colnames(castdtm))
words_kmean <- bind_cols(words_kmean, as_tibble(t(km_out.2016$centers)))  # Binding the transpose of th

gather(words_kmean, cluster, value, -word) %>%
  group_by(cluster) %>%
  top_n(4, value) %>%
  arrange(-value) %>%
  summarise(Top_Words = str_c(word, collapse = ", ")) %>%
  mutate(NumTweets = table(km_out.2016$cluster))
```

```
## # A tibble: 5 x 3
##   cluster Top_Words                                        NumTweets
##   <chr>   <chr>                                            <table>
## 1 1       america, safe, polls, exciting, ohio, terrible    32
## 2 2       people, america, vote, florida                  1250
## 3 3       trump, donald, tower, clinton                    110
## 4 4       hillary, crooked, clinton, bernie                324
## 5 5       poll, florida, america, iowa, nevada, progress    14
```

How does that compare to what was happening in 2019?

```
# INSERT CODE
```

Now let's analyze and uncover the 5 clusters.

```
# INSERT CODE
```

And now summarize the top 4 words associated with each of the 5 clusters.

```
# INSERT CODE
```

How does that compare? How does the content and emphasis of we find vary over time? And how do those isights compare to the insights we get from analyzing sentiments?