Topic 4: Sentiment Analysis II

Charles Hendrickson

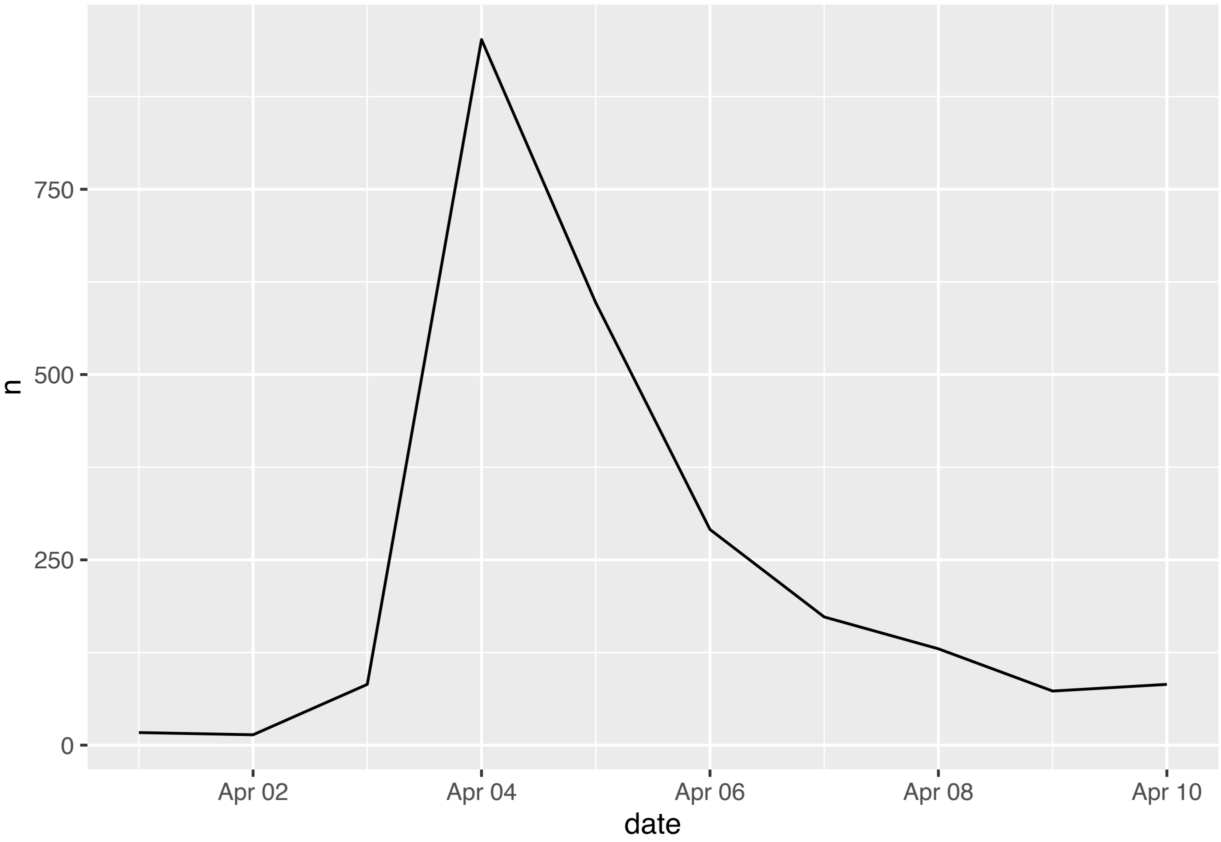
4/26/2022

This .Rmd available here: https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/topic_4.Rmd

```r
library(quanteda)
#devtools::install_github("quanteda/quanteda.sentiment") #not available currently through CRAN
library(quanteda.sentiment)
library(quanteda.textstats)
library(tidyverse)
library(tidytext)
library(lubridate)
library(wordcloud) #visualization of common words in the data set
library(reshape2)
library(sentimentr)
```

IPCC Report Twitter Last week we used the tidytext approach to sentiment analysis for Nexis Uni .pdf data on coverage of the recent IPCC report. This week we will look at the conversation on Twitter about the same report. We'll start with the familiar tidy approach, and then introduce the quanteda package later.

```r
raw_tweets <- read.csv("data/IPCC_tweets_April1-10_sample.csv")

dat<- raw_tweets[,c(4,6)] # Extract Date and Title fields

tweets <- tibble(text = dat$Title,
                  id = seq(1:length(dat$Title)),
                 date = as.Date(dat$Date,'%m/%d/%y'))


#head(tweets$text, n = 10)

#simple plot of tweets per day
tweets %>%
  count(date) %>%
  ggplot(aes(x = date, y = n))+
  geom_line()
```

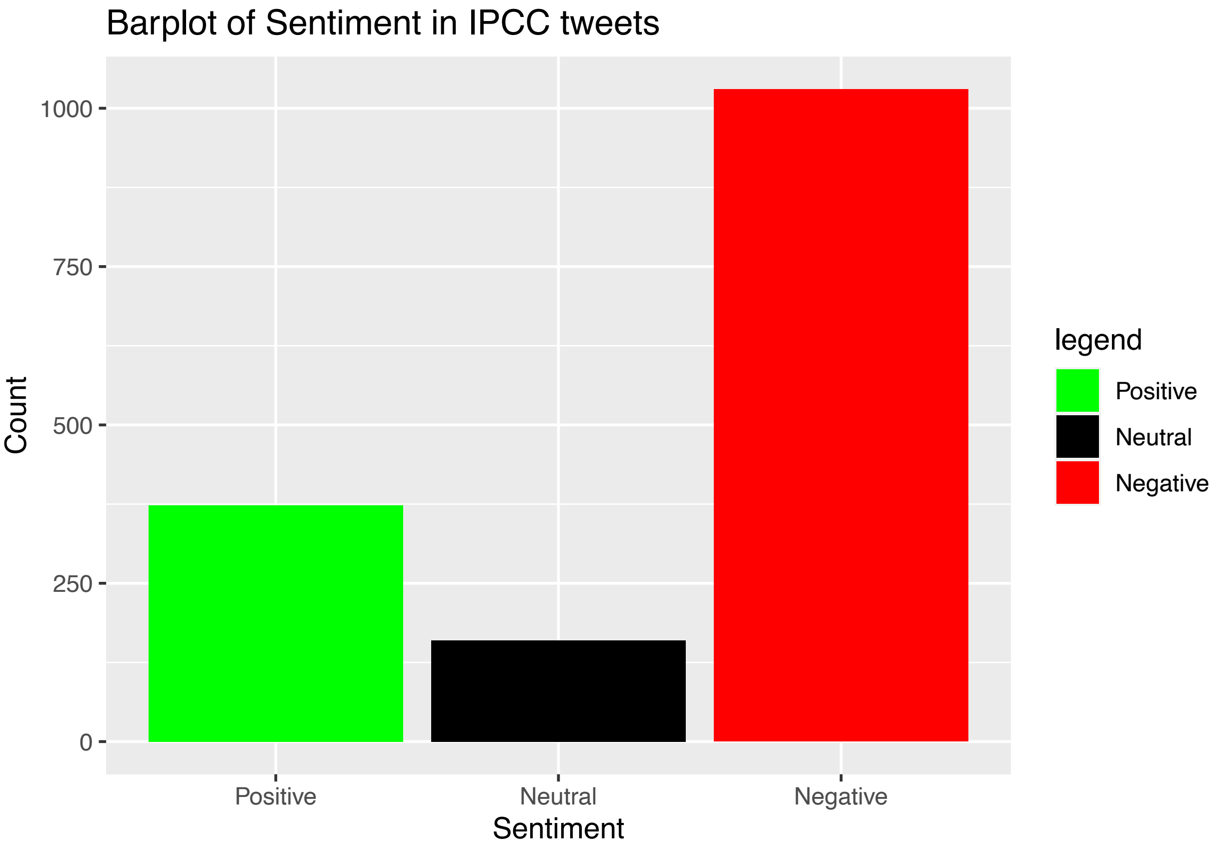


Q1

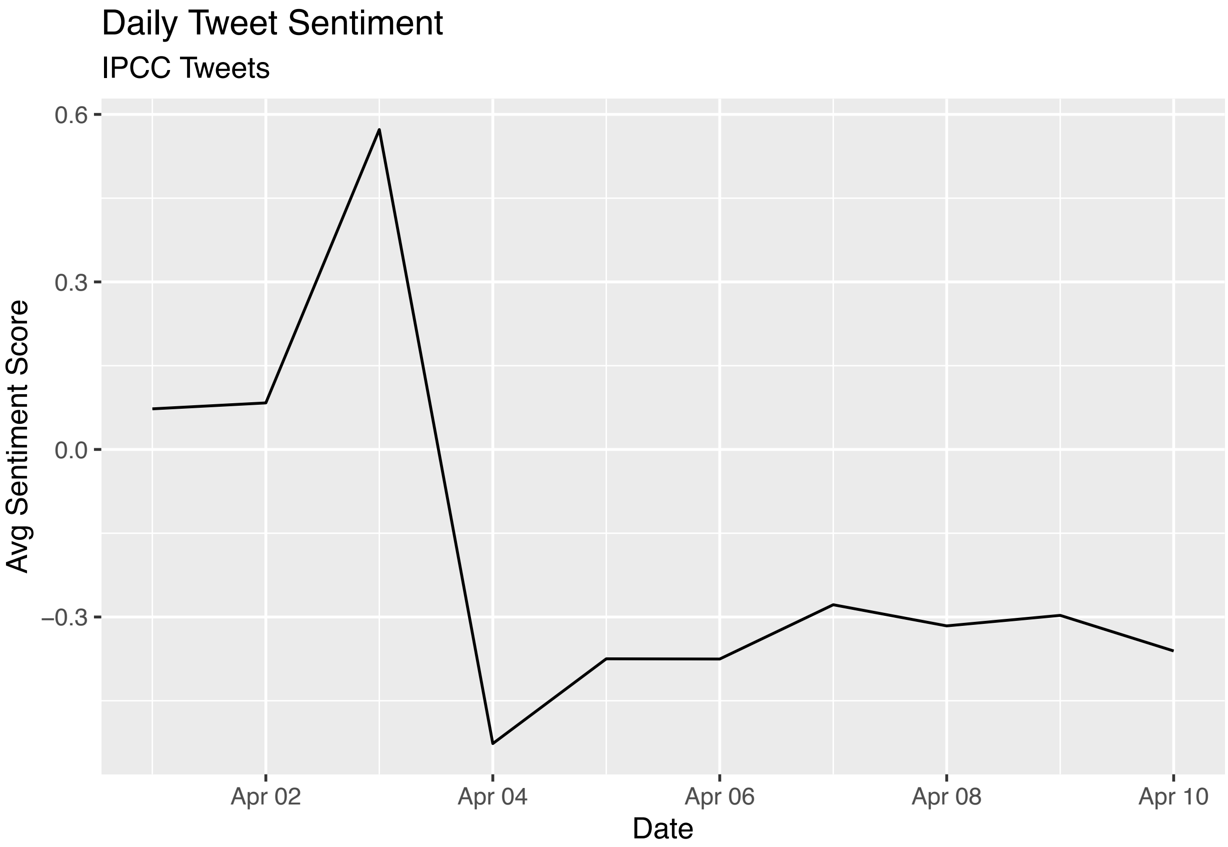
Remove twitter account mentions

```
#let's clean up the URLs from the tweets
tweets$text <- gsub("http[^[:space:]]*", "",tweets$text)
tweets$text <- str_to_lower(tweets$text)

#Remove twitter account mentions from the text field of the tweets tibble.
tweets$text <- gsub("@[^[:space:]]*", "",tweets$text)

#head(tweets$text, n = 10)

#load sentiment lexicons
bing_sent <- get_sentiments('bing')
nrc_sent <- get_sentiments('nrc')

#tokenize tweets to individual words
words <- tweets %>%
  select(id, date, text) %>%
  unnest_tokens(output = word, input = text, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  left_join(bing_sent, by = "word") %>%
  left_join(
    tribble(
      ~sentiment, ~sent_score,
      "positive", 1,
      "negative", -1),
    by = "sentiment")
```

```
#take average sentiment score by tweet
tweets_sent <- tweets %>%
  left_join(
    words %>%
      group_by(id) %>%
      summarize(
        sent_score = mean(sent_score, na.rm = T)),
    by = "id")

neutral <- length(which(tweets_sent$sent_score == 0))
positive <- length(which(tweets_sent$sent_score > 0))
negative <- length(which(tweets_sent$sent_score < 0))

Sentiment <- c("Positive","Neutral","Negative")
Count <- c(positive,neutral,negative)
output <- data.frame(Sentiment,Count)
output$Sentiment<-factor(output$Sentiment,levels=Sentiment)
ggplot(output, aes(x=Sentiment,y=Count))+
  geom_bar(stat = "identity", aes(fill = Sentiment))+
  scale_fill_manual("legend", values = c("Positive" = "green", "Neutral" = "black", "Negative" = "red")
  ggtitle("Barplot of Sentiment in IPCC tweets")
```



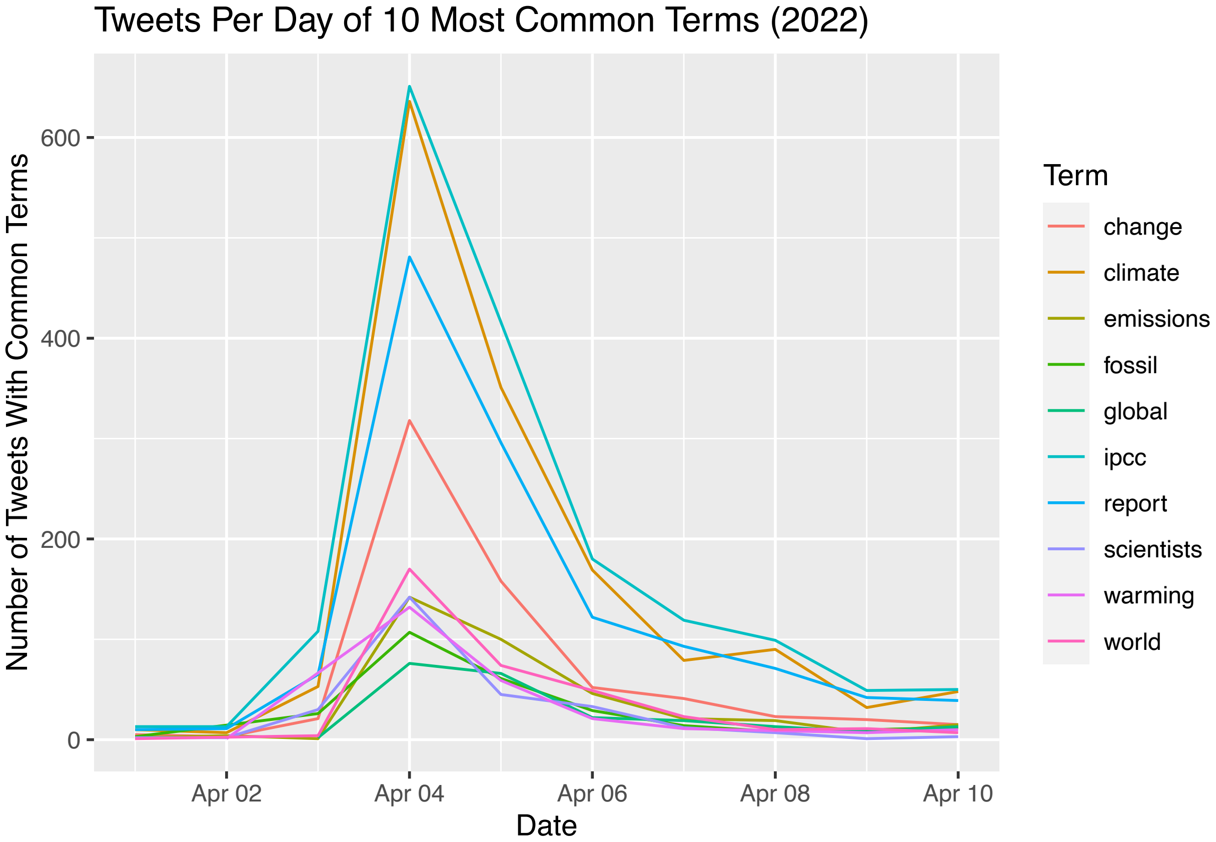
```
# tally sentiment score per day
daily_sent <- tweets_sent %>%
  group_by(date) %>%
  summarize(sent_score = mean(sent_score, na.rm = T))

daily_sent %>%
  ggplot( aes(x = date, y = sent_score)) +
  geom_line() +
    labs(x = "Date",
    y = "Avg Sentiment Score",
    title = "Daily Tweet Sentiment",
    subtitle = "IPCC Tweets")
```



Q2

Compare the ten most common terms in the tweets per day. Do you notice anything interesting?

```
#Word count
text_wordcounts <- words  %>% count(word, sort = TRUE)

#Get only the ten most common terms
text_wordcounts <- text_wordcounts[1:10, ]

#select terms in words df that match the 10 most common terms
common_terms <- words %>%
  subset(word %in% text_wordcounts$word)

# Group by date
common_terms_date <- common_terms %>%
  group_by(date) %>%
  count(word)

#Plot of ten most common terms in the tweets per day
ggplot(data = common_terms_date, aes(x = date, y = n)) +
  geom_line(aes(color = word)) +
  labs(title = "Tweets Per Day of 10 Most Common Terms (2022)",
       x = "Date",
       y = "Number of Tweets With Common Terms") +
  labs(color = "Term")
```



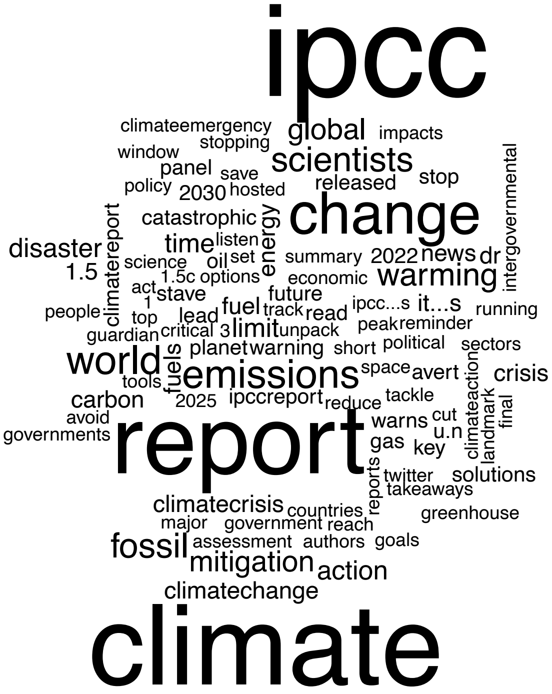
I noticed that the number of tweets per day with the ten most common terms peaked on April 4th, 2022. This is likely because the Summary for Policymakers of the IPCC Working Group III report, Climate Change 2022: Mitigation of climate change was approved on April 4 2022, by 195 member governments of the IPCC. The most common terms were 'IPCC', 'climate', 'report', and 'world' which was not surprising because the IPCC tweet data is going to contain a lot of references to reports on climate change around the world. However, I expected 'global' to be used more often then 'world' because it is often included in the term 'global warming' which is a key term for the IPCC.

Q3

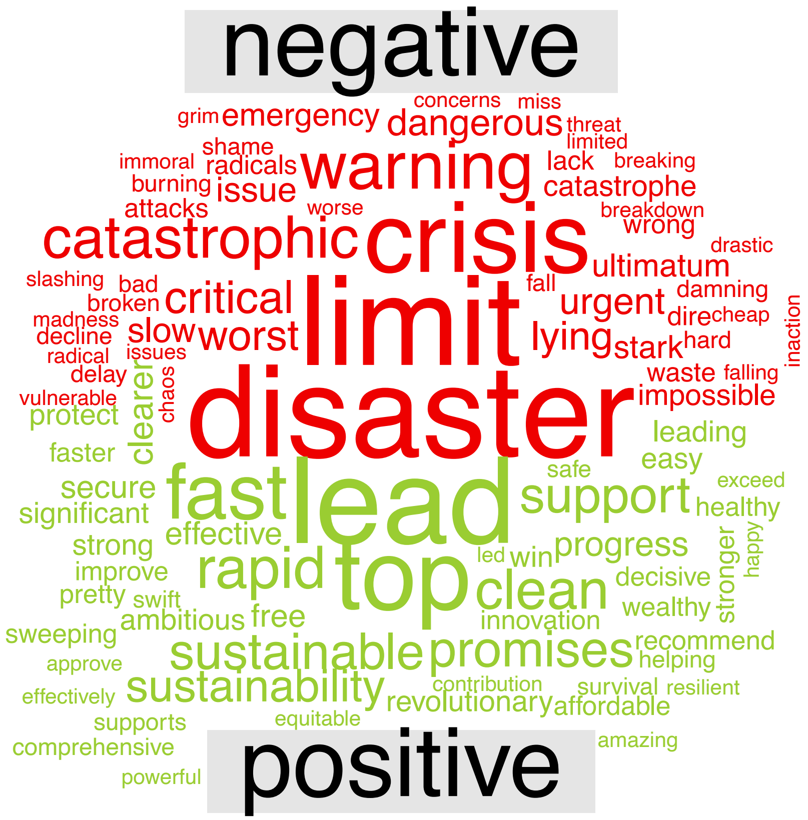
Adjust the wordcloud in the "wordcloud" chunk by coloring the positive and negative words so they are identifiable.

Now let's try a new type of text visualization: the wordcloud.

```
words %>%
   anti_join(stop_words) %>%
   count(word) %>%
   with(wordcloud(word, n, max.words = 100))
```



```
words %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
acast(word ~ sentiment, value.var = "n", fill = 0) %>%
comparison.cloud(colors = c("red2", "olivedrab3"),
                 max.words = 100)
```



The quanteda package quanteda is a package (actually a family of packages) full of tools for conducting text analysis. quanteda.sentiment (not yet on CRAN, download from github) is the quanteda modular package for conducting sentiment analysis.

quanteda has its own built in functions for cleaning text data. Let's take a look at some. First we have to clean the messy tweet data:

```
corpus <- corpus(dat$Title) #enter quanteda
#summary(corpus)
```

```
tokens <- tokens(corpus) #tokenize the text so each doc (page, in this case) is a list of tokens (words

#examine the uncleaned version
#tokens

#clean it up
tokens <- tokens(tokens, remove_punct = TRUE,
                         remove_numbers = TRUE)

tokens <- tokens_select(tokens, stopwords('english'),selection='remove') #stopwords lexicon built in to

#tokens <- tokens_wordstem(tokens) #stem words down to their base form for comparisons across tense and

tokens <- tokens_tolower(tokens)
```

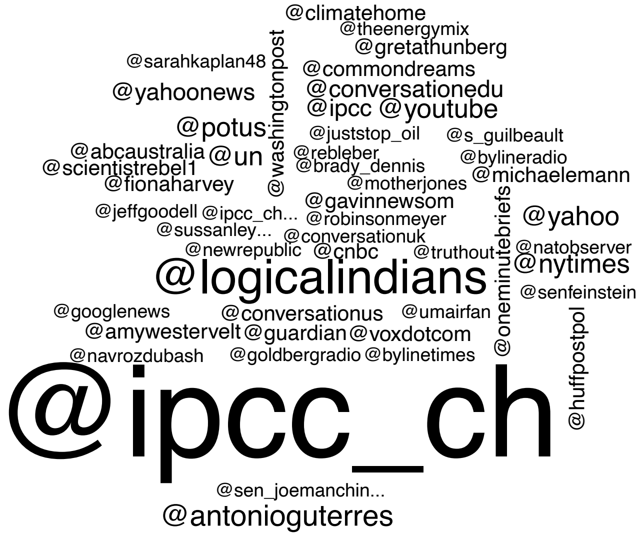
We can use the kwic function (keywords-in-context) to briefly examine the context in which certain words or patterns appear.

```
#head(kwic(tokens, pattern = "climate", window = 3))

#head(kwic(tokens, pattern = phrase("climate change"), window = 3))
```

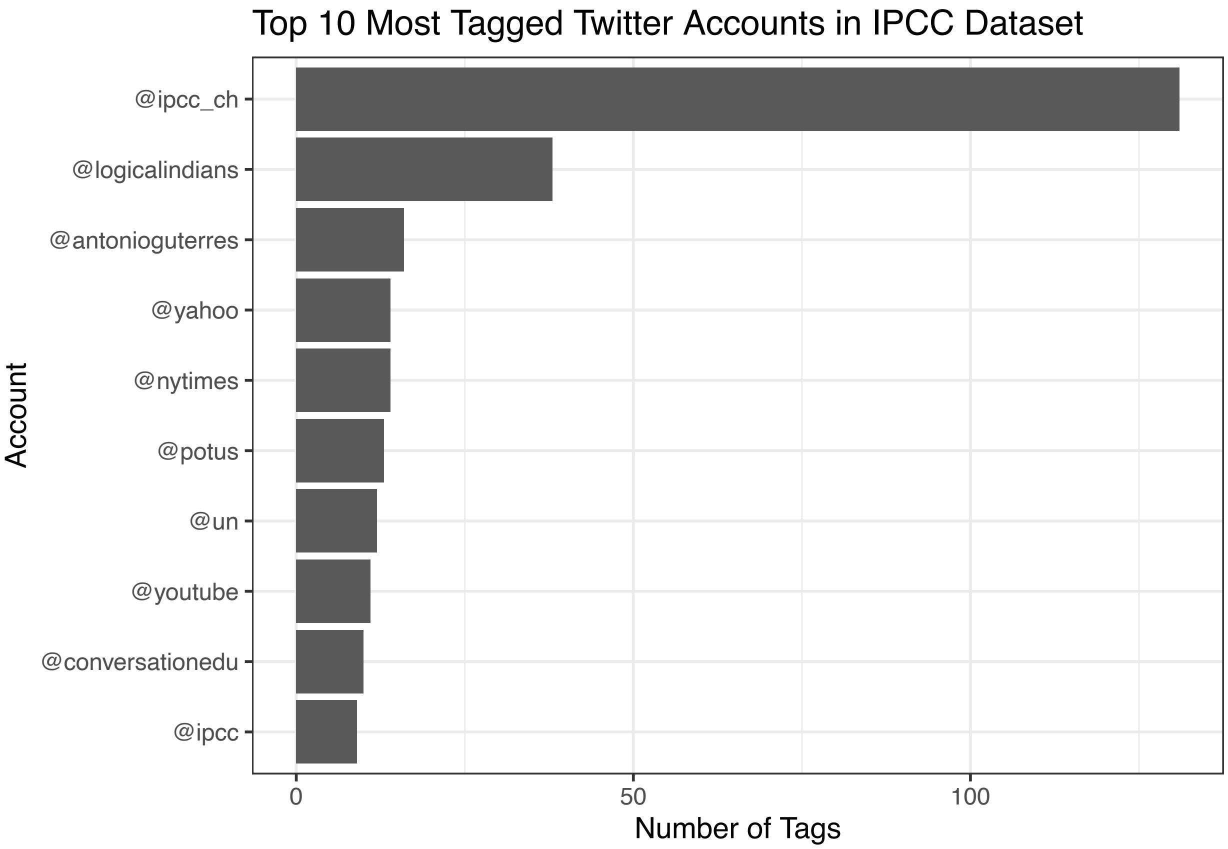
Q4

Let's say we are interested in the most prominent entities in the Twitter discussion. Which are the top 10 most tagged accounts in the data set? Hint: the "explore_hashtags" chunk is a good starting point.

```
tagged_tweets <- tokens(corpus, remove_punct = TRUE) %>%
               tokens_keep(pattern = "@*")

dfm_tagged<- dfm(tagged_tweets)

tstat_freq <- textstat_frequency(dfm_tagged, n = 100)

tstat_freq <- tstat_freq[1:10, ]

#tidytext gives us tools to convert to tidy from non-tidy formats
tagged_tib<- tidy(dfm_tagged)

tagged_tib %>%
   count(term) %>%
   with(wordcloud(term, n, max.words = 50))
```



```
# Plot Top 10 Most Tagged Twitter Accounts in IPCC Dataset
ggplot(data = tstat_freq, aes(y = reorder(feature, -rank), x = frequency)) +
  geom_bar(stat = "identity") +
  labs(title = "Top 10 Most Tagged Twitter Accounts in IPCC Dataset",
       x = "Number of Tags",
       y = "Account") +
  theme_bw()
```



Create the sparse matrix representation known as the document-feature matrix. quanteda's textstat_polarity function has multiple ways to combine polarity to a single score. The sent_logit value to fun argument is the log of (pos/neg) counts.

```
dfm <- dfm(tokens)

topfeatures(dfm, 12)
```

```
##    climate       ipcc     report     change        now      #ipcc      world 
##       1396       1243       1225        651        505        464        346 
##  emissions      never        new     latest scientists 
##        333        291        279        279        274 
```

```
dfm.sentiment <- dfm_lookup(dfm, dictionary = data_dictionary_LSD2015)

#head(textstat_polarity(tokens, data_dictionary_LSD2015, fun = sent_logit))
```

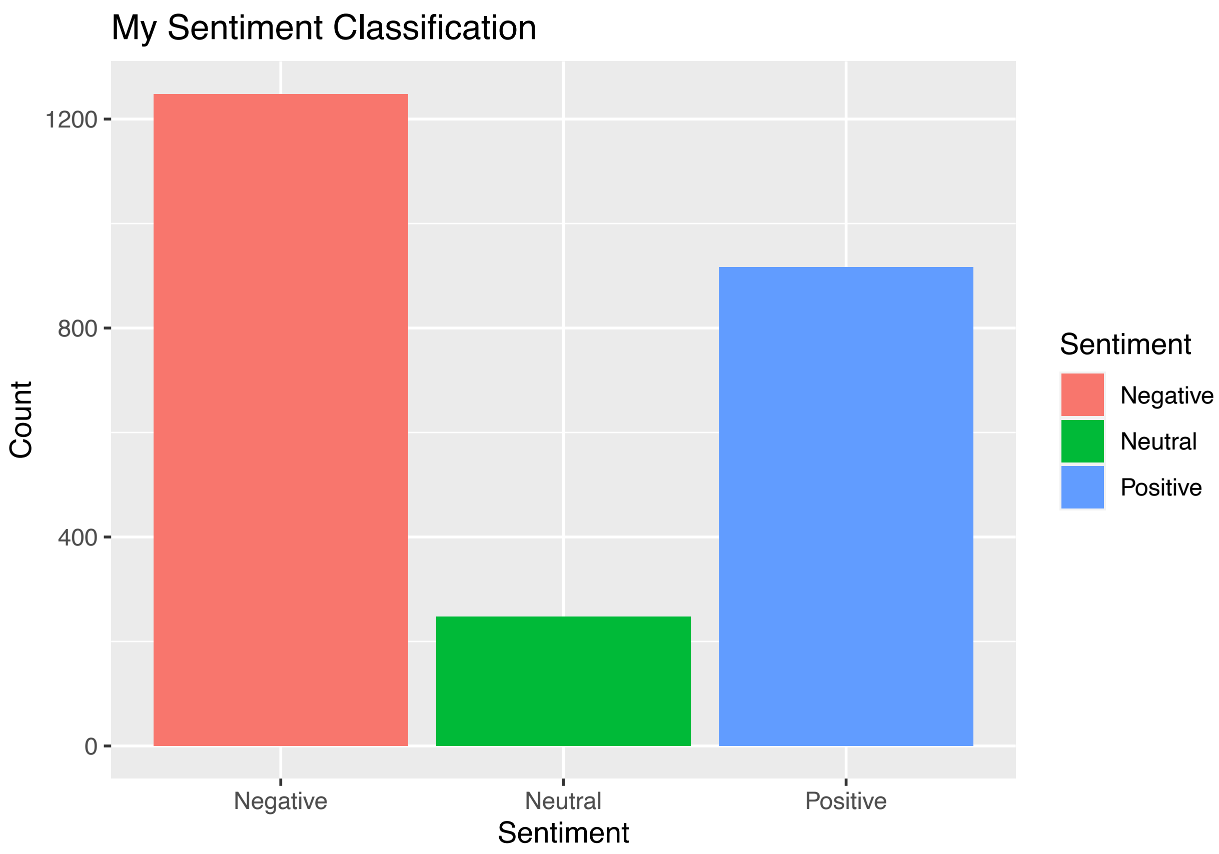
Q5

The Twitter data download comes with a variable called "Sentiment" that must be calculated by Brandwatch. Use your own method to assign each tweet a polarity score (Positive, Negative, Neutral) and compare your classification to Brandwatch's (hint: you'll need to revisit the "raw_tweets" data frame).

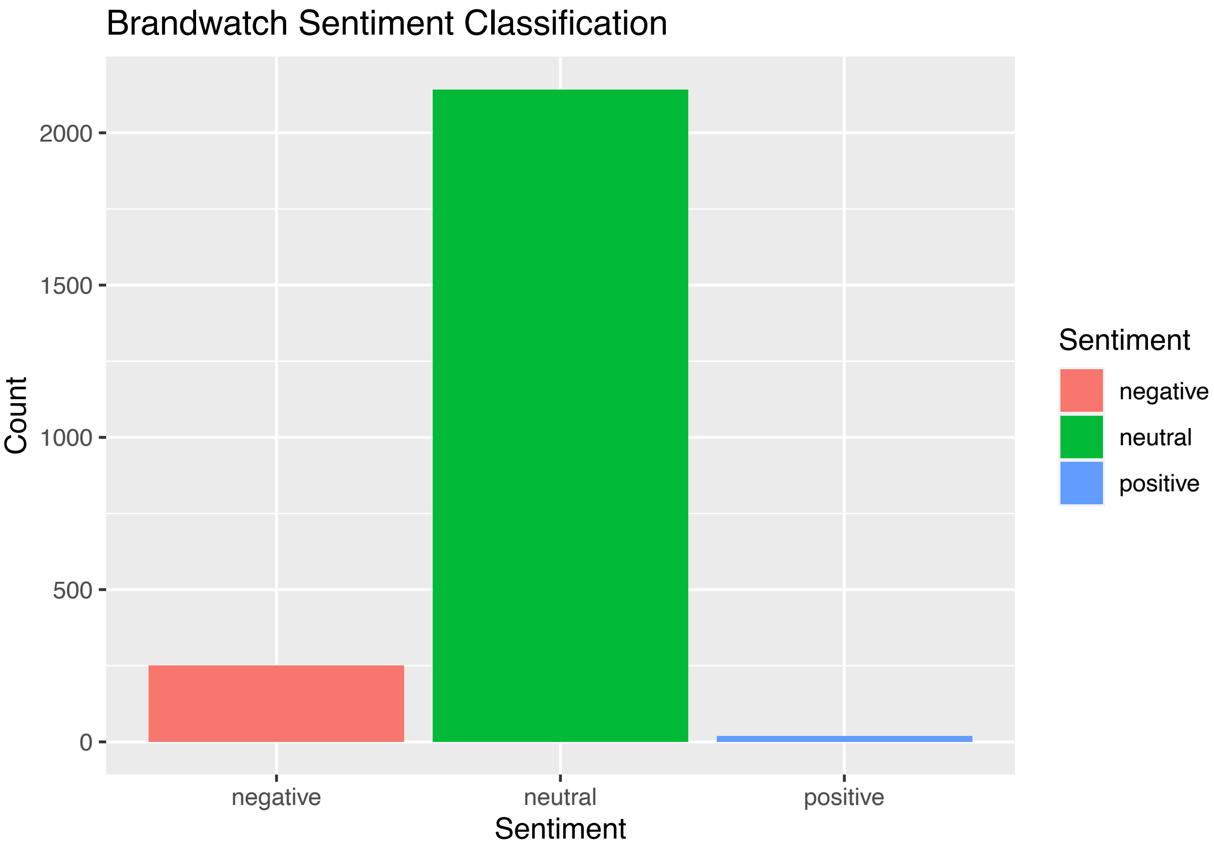
```
#Assigning polarity score

dat2<- raw_tweets[,c(4,6)] # Extract Date and Title fields

#Create text, id, and date columns
```

```r
tweets2 <- tibble(text = dat2$Title,
                  id = seq(1:length(dat2$Title)),
                 date = as.Date(dat2$Date,'%m/%d/%y'))


#clean up the URLs from the tweets
tweets2$text <- gsub("http[^[:space:]]*", "",tweets2$text)
tweets2$text <- str_to_lower(tweets2$text)

#Remove twitter account mentions from the text field of the tweets tibble.
tweets2$text <- gsub("@[^[:space:]]*", "",tweets2$text)

#Run the raw `character` vector through the `get_sentences` function
#Sentiment function allows for us to approximate the sentiment (polarity) of text by sentence.
mytext <- get_sentences(tweets2$text) %>%
  sentiment() %>%
  rename(sentiment_score = sentiment)

#Assign sentiment based on polarity
sent_assigned <- mytext %>%
  group_by(element_id) %>%
  summarise(sentiment_score = mean(sentiment_score))

sent_assigned$sentiment <- ifelse(sent_assigned$sentiment_score <0, "Negative", ifelse(sent_assigned$sen
```

```r
#Find the number of tweets with each sentiment type for my sentiment calculation
my_raw_sentiment <- sent_assigned %>%
  group_by(sentiment) %>%
  summarise(sentiment_count = n())


#Plot the count per sentiment
ggplot(data = my_raw_sentiment, aes(x = Sentiment, y = sentiment_count)) +
         geom_bar(stat = "identity", aes(fill = Sentiment)) +
  labs(title = "My Sentiment Classification",
         x= "Sentiment",
         y= "Count")
```



```
#Find the number of tweets with each sentiment type for Brandwatch
raw_sentiment <- raw_tweets %>%
  group_by(Sentiment) %>%
  summarise(sentiment_count = n())

#Plot the count per sentiment
ggplot(data = raw_sentiment, aes(x = Sentiment, y = sentiment_count)) +
         geom_bar(stat = "identity", aes(fill = Sentiment)) +
  labs(title = "Brandwatch Sentiment Classification",
         x= "Sentiment",
         y= "Count")
```



My sentiment classification method resulted in many more 'negative' and 'positive' sentiments and much fewer 'neutral' sentiments being assigned to the tweets data compared to Brandwatch's methods. These results could be due to my methods, in which I first used the 'get_sentences()' and 'sentiment()' function to assign a polarity score (<0, 0, >0) to each sentence. Next, I took the mean of the polarity scores for each sentence to get an average polarity score for each tweet. I think used a ifelse() statement to assign the tweet a negative, positive, or neutral sentiment based on if its its mean polarity score is less than zero, greater than zero, or equal to zero respectively. However, this method makes it much more difficult for a sentence to be neutral because all the sentences within a tweet would need to have a polarity score of zero in order for that tweet to have a 'neutral' sentiment. Thus, tweets that are made up of a majority of neutral sentences will still be classified as positive or negative based on a minority of the sentiment score makeup."

Assignment

You will use the tweet data from class today for each part of the following assignment.

1. Think about how to further clean a twitter data set. Let's assume that the mentions of twitter accounts is not useful to us. Remove them from the text field of the tweets tibble.
2. Compare the ten most common terms in the tweets per day. Do you notice anything interesting?
3. Adjust the wordcloud in the "wordcloud" chunk by coloring the positive and negative words so they are identifiable.
4. Let's say we are interested in the most prominent entities in the Twitter discussion. Which are the top 10 most tagged accounts in the data set. Hint: the "explore_hashtags" chunk is a good starting point.
5. The Twitter data download comes with a variable called "Sentiment" that must be calculated by Brandwatch. Use your own method to assign each tweet a polarity score (Positive, Negative, Neutral)

and compare your classification to Brandwatch's (hint: you'll need to revisit the "raw_tweets" data frame).