Lab 4: Sentiment Analysis II

Desik Somasundaram

Assignment

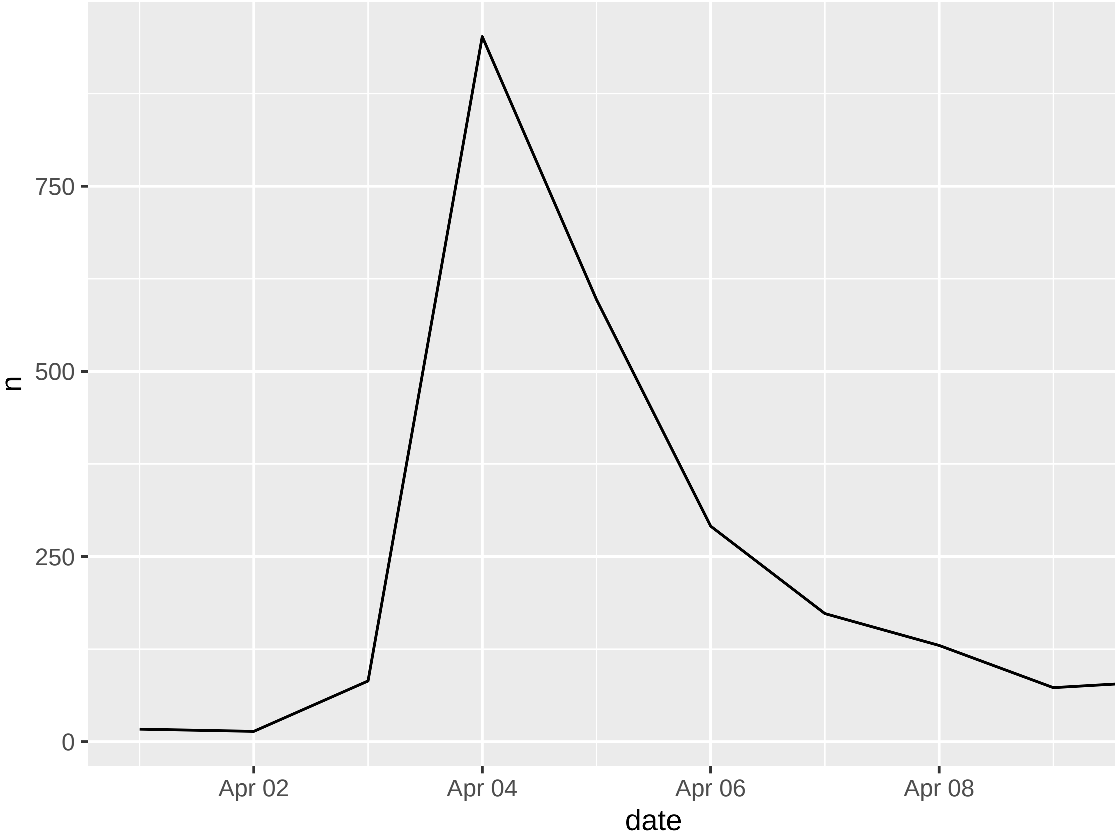
You will use the tweet data from class today for each part of the following assignment.

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).

```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)
library(patchwork)
library(janitor)
```

```r
raw_tweets <- read.csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/dat/IPCC_t

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'))
```

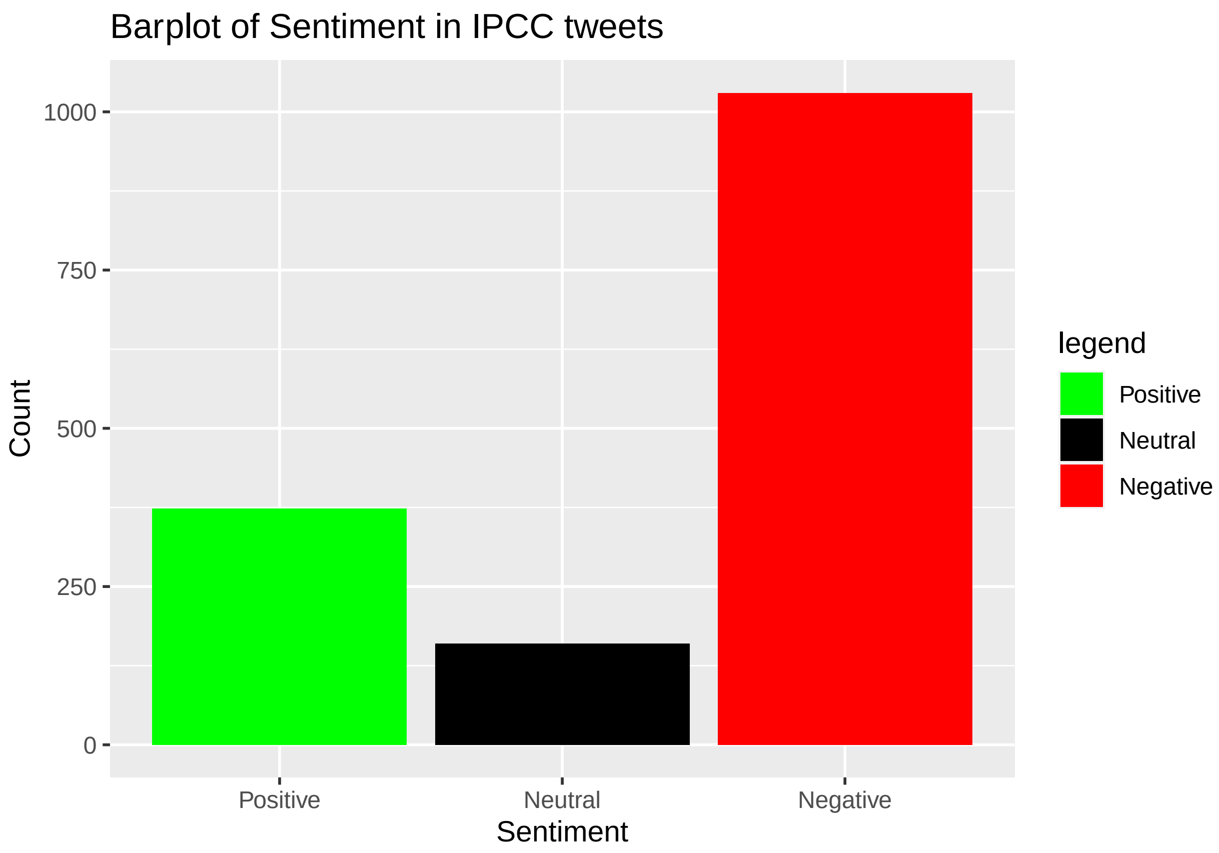
```
#head(tweets$text, n = 10)

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



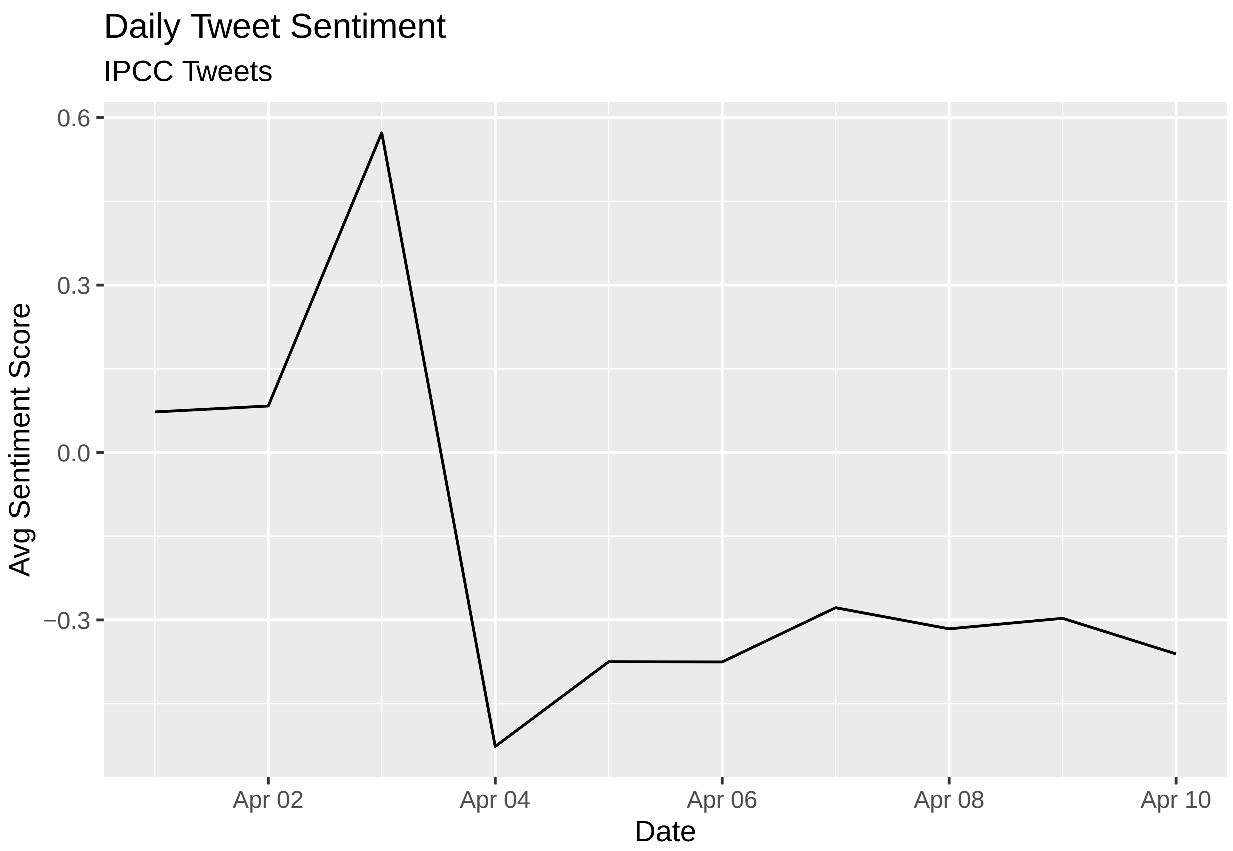
IPCC Report Twitter

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

#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")
```



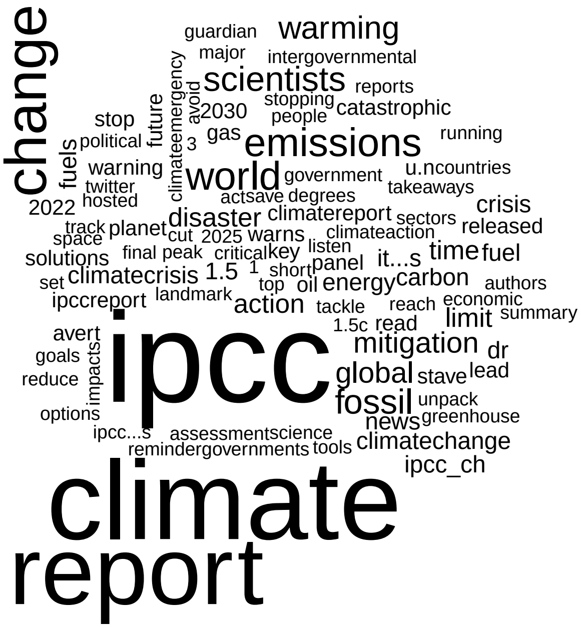
```
# tally sentiment score per day
daily_sent <- tweets_sent %>%
  group_by(date) %>%
  summarize(sent_score = mean(sent_score, na.rm = T))

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")
```



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

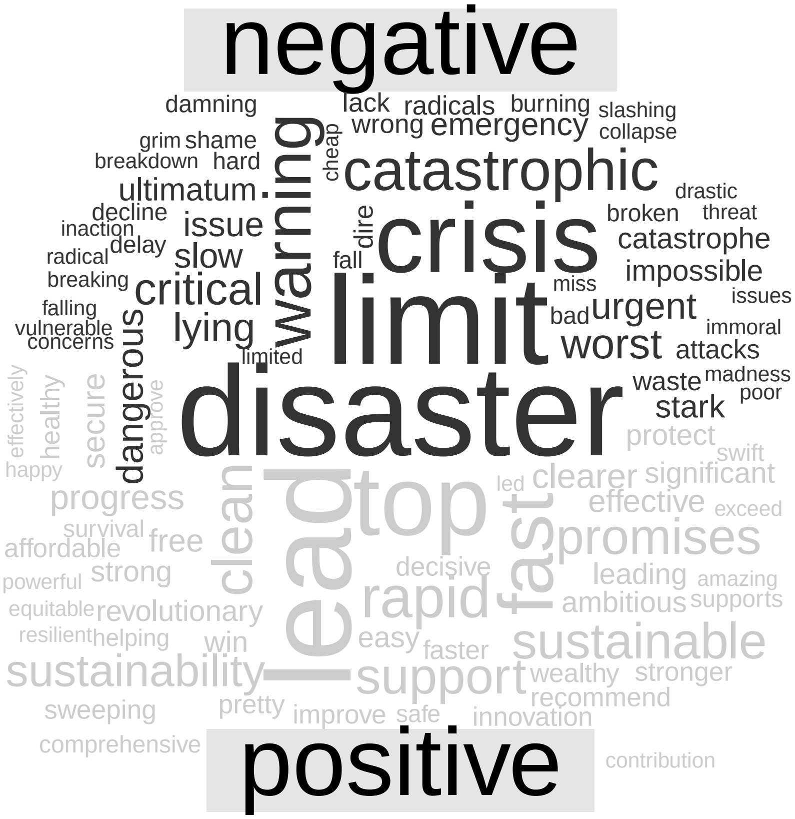
```
words %>%
   anti_join(stop_words) %>%
   count(word) %>%
   with(wordcloud(word, n, max.words = 100))
```

```
## Joining, by = "word"

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'it's'
## in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'it's'
## in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'it's'
## in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it's' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it's' in 'mbcsToSbcs': dot substituted for
## <80>
## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it's' in 'mbcsToSbcs': dot substituted for
## <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'ipcc's' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'ipcc's' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'ipcc's' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'ipcc's' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'ipcc's' in 'mbcsToSbcs': dot substituted for
## <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'ipcc's' in 'mbcsToSbcs': dot substituted for
## <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019
```



```r
words %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
acast(word ~ sentiment, value.var = "n", fill = 0) %>%
comparison.cloud(colors = c("gray20", "gray80"),
                 max.words = 100)
```

```
## Joining, by = c("word", "sentiment")
```



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)
```

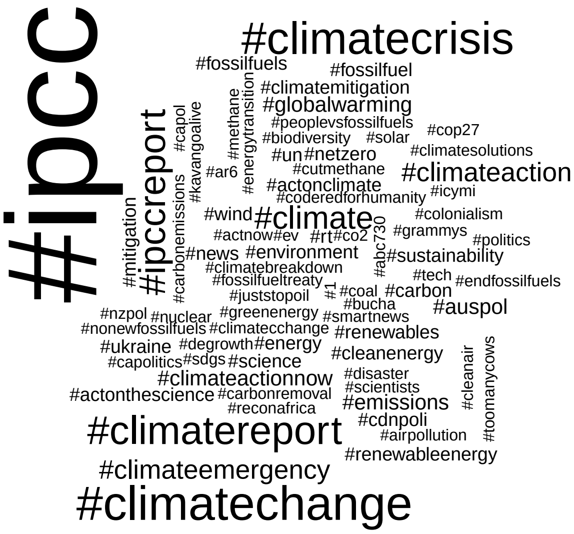
We can use the kwic function (keywords-in-context) to briefly examine the context in which certain words or patterns appear.

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

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

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

dfm_hash<- dfm(hash_tweets)

tstat_freq <- textstat_frequency(dfm_hash, n = 100)
head(tstat_freq, 10)
```

```
##                feature frequency rank docfreq group
## 1                #ipcc       464    1     460   all
## 2       #climatechange       137    2     135   all
## 3       #climatecrisis       118    3     117   all
## 4       #climatereport        97    4      97   all
## 5          #ipccreport        87    5      87   all
## 6             #climate        68    6      67   all
## 7    #climateemergency        45    7      45   all
## 8       #climateaction        44    8      44   all
## 9       #globalwarming        24    9      24   all
## 10 #climateactionnow          23   10      23   all
```

```
#tidytext gives us tools to convert to tidy from non-tidy formats
hash_tib<- tidy(dfm_hash)

hash_tib %>%
   count(term) %>%
   with(wordcloud(term, n, max.words = 100))
```



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)

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

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

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.

```
theString <- unlist(strsplit(tweets$text, " "))
regex <- "(^|[^@\\w])@(\\w{1,15})\\b"
tweets$text <-gsub(regex, "", tweets$text)
```

```
tokenized_tweets <- tweets %>%
                    unnest_tokens(word, text)
```

```
data(stop_words)
```

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

```
words_by_date <- tokenized_tweets %>%
    anti_join(stop_words) %>%
    group_by(date) %>%
    count(date, word)
```

```
## Joining, by = "word"
```

```
top_words_by_date <- words_by_date %>% group_by(date) %>% top_n(n = 10, wt = n)
top_words_by_date[order(top_words_by_date$n, decreasing = TRUE),]
```

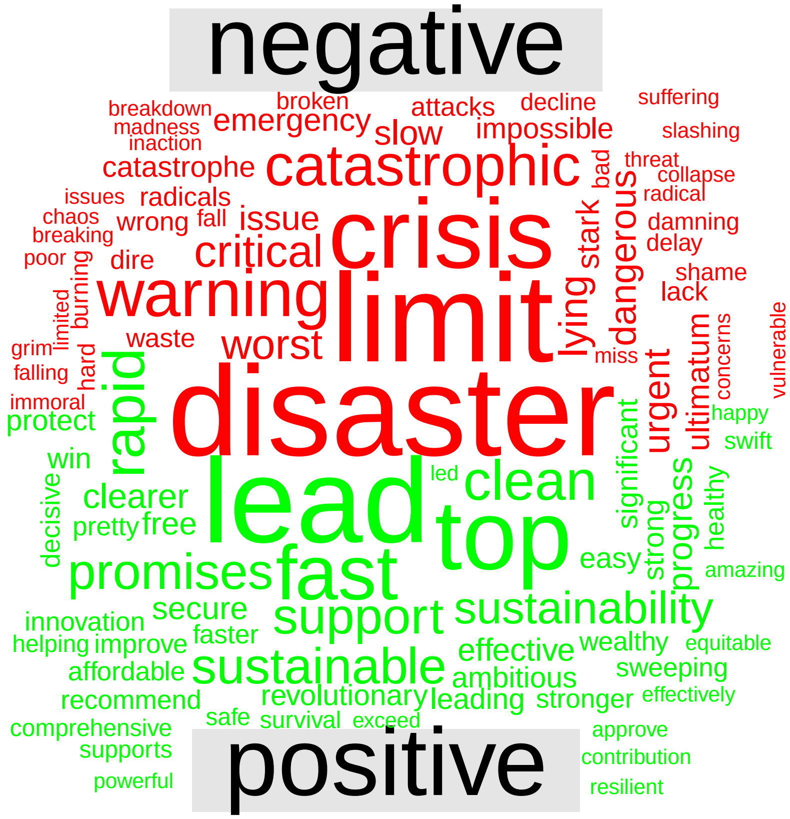
```
## # A tibble: 122 x 3
## # Groups:   date [10]
##    date       word        n
##    <date>     <chr>   <int>
##  1 2022-04-04 ipcc      651
##  2 2022-04-04 climate   636
##  3 2022-04-04 report    481
##  4 2022-04-05 ipcc      416
##  5 2022-04-05 climate   351
##  6 2022-04-04 change    318
##  7 2022-04-05 report    296
##  8 2022-04-06 ipcc      180
##  9 2022-04-04 world     170
## 10 2022-04-06 climate   169
## # ... with 112 more rows
```

It is evident that throughout all the days, "ipcc", "climate", "report" and "change" are the top words. On some days, words like "dr." and people's last names made it to the top 10 indicating the release of important publications.

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

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

```
## Joining, by = c("word", "sentiment")
```



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.

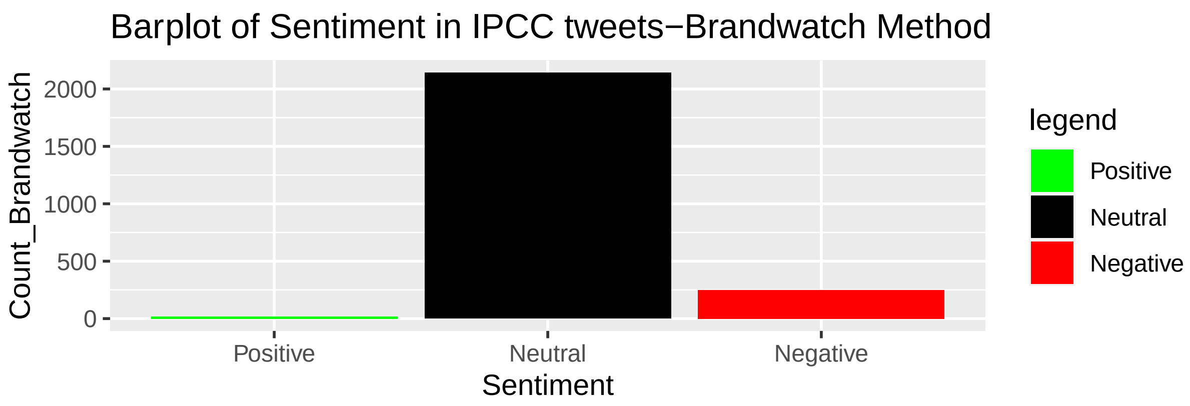
```
at_tweets <- tokens(corpus, remove_punct = TRUE) %>%
               tokens_keep(pattern = "@*")

dfm_at<- dfm(at_tweets)

tstat_freq <- textstat_frequency(dfm_at, n = 10)

tstat_freq
```

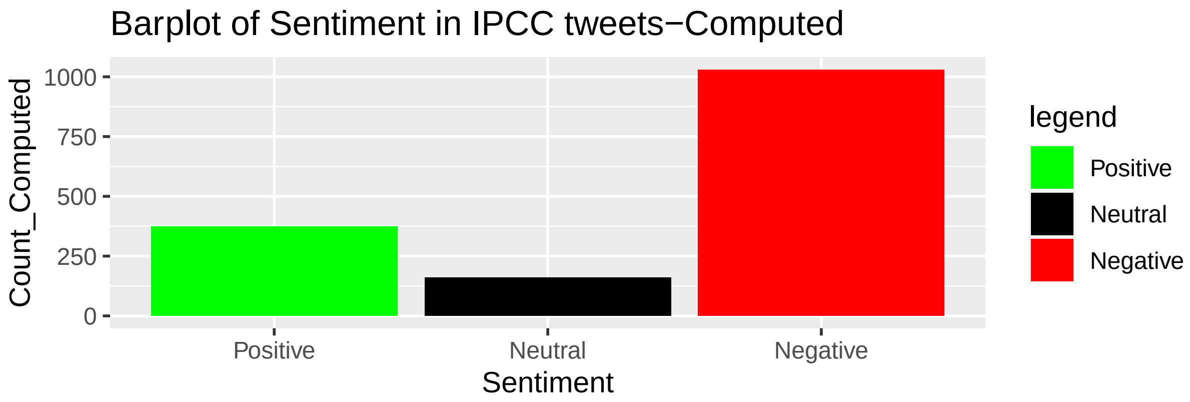
```
##               feature frequency rank docfreq group
## 1            @ipcc_ch       131    1     131   all
## 2      @logicalindians        38    2      38   all
## 3   @antonioguterres        16    3      16   all
## 4            @nytimes        14    4      14   all
## 5              @yahoo        14    4      14   all
## 6              @potus        13    6      13   all
## 7                 @un        12    7      12   all
## 8            @youtube        11    8      11   all
## 9   @conversationedu        10    9      10   all
## 10              @ipcc         9   10       9   all
```

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).

```
#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))

b_neutral <- length(which(raw_tweets$Sentiment == "neutral"))
b_positive <- length(which(raw_tweets$Sentiment =="positive"))
b_negative <- length(which(raw_tweets$Sentiment == "negative"))


Sentiment <- c("Positive","Neutral","Negative")
Count_Computed <- c(positive,neutral,negative)
Count_Brandwatch <- c(b_positive,b_neutral,b_negative)
output <- data.frame(Sentiment,Count_Computed,Count_Brandwatch)
output$Sentiment<-factor(output$Sentiment,levels=Sentiment)
brandwatch <- ggplot(output, aes(x=Sentiment,y=Count_Brandwatch))+
              geom_bar(stat = "identity", aes(fill = Sentiment))+
              scale_fill_manual("legend", values = c("Positive" = "green", "Neutral" = "black", "Negati
              ggtitle("Barplot of Sentiment in IPCC tweets-Brandwatch Method")

computed <- ggplot(output, aes(x=Sentiment,y=Count_Computed))+
              geom_bar(stat = "identity", aes(fill = Sentiment))+
              scale_fill_manual("legend", values = c("Positive" = "green", "Neutral" = "black", "Negati
              ggtitle("Barplot of Sentiment in IPCC tweets-Computed")

brandwatch/computed
```





It is interesting to see that the Brandwatch computed scores are mostly neutral while the computed ones scores are mostly negative. The lack of much positive tweets in the Brandwatch methodology is a bit concerning and warranted further research.