Week 4 Lab - Sentiment Analysis II

Steven Cognac

2022-04-25

```
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(sentimentr)
library(reshape2)
library(patchwork)
```

Read in Twitter data

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

# updated columns
dat <- raw_tweets[, c(4, 6, 10:11)]

clean_tweets <- tibble(id = seq(1:length(dat$Title)),
                 text = dat$Title,
                 date = as.Date(dat$Date,'%m/%d/%y'),
                 sentiment = dat$Sentiment,
                 emotion = dat$Emotion)
```

Part 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.

Here we remove the "@" symbols, or account names, from the tweets.

```
# let's find the "@" from the tweets
tweets <- clean_tweets %>%
  mutate(contains_account = str_detect(text, "@[a-z,A-Z]*", negate = FALSE))

# let's remove those @ symbols and any word that immediately follows it
tweets$text <- gsub("@[a-z,A-Z]*", "", tweets$text)
```

```
tweets$text <- str_to_lower(tweets$text)

# let's double check if we removed all the "@" symbols
tweets <- tweets %>%
  mutate(contains_account = str_detect(text, "@[a-z,A-Z]*", negate = FALSE))
```

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

To do this we'll start by tokenizing our tweets. This includes extracting individual words by date from each tweet. While we're at it let's remove stop_words and remove useless string patterns.

```
#tokenize tweets to individual words
words <- tweets %>%
  select(id, date, text) %>%
  unnest_tokens(output = word, input = text, token = "words")

# removes all the stop_words from the 'words$text' column
words_tokenized <- words %>%
  anti_join(stop_words)
```

Looks like there's still a fair amount of fluff in our word bank. Let's remove additional phrases and fluff words manually.

```
# remove unwanted words
clean_tokens <- str_remove_all(words_tokenized$word, "t.co")
clean_tokens <- str_remove_all(clean_tokens, "here's")
clean_tokens <- str_remove_all(clean_tokens, "https")
clean_tokens <- str_remove_all(clean_tokens, "\\.")
clean_tokens <- str_remove_all(clean_tokens, "_ch")

#remove all numbers
clean_tokens <- str_remove_all(clean_tokens, "[:digit:]")

# removes "'s"
clean_tokens <- gsub("'s", '', clean_tokens)

# add cleaned words to stop_words
words_tokenized$clean <- clean_tokens

#remove the empty strings
tib <-subset(words_tokenized, clean!="")

#reassign
words_tokenized <- tib
```

Here we plot the top 10 words by day.

First we'll group our `tokenized_words()` by date, then `count()` the frequency of each word, and finally grab the top 10 words from each date.

We'll use facet_wrap() to group our top 10 words by day. But before we do that, we'll need to reorder our columns to decending order. Julie Silge has a great blog tutorial on that.

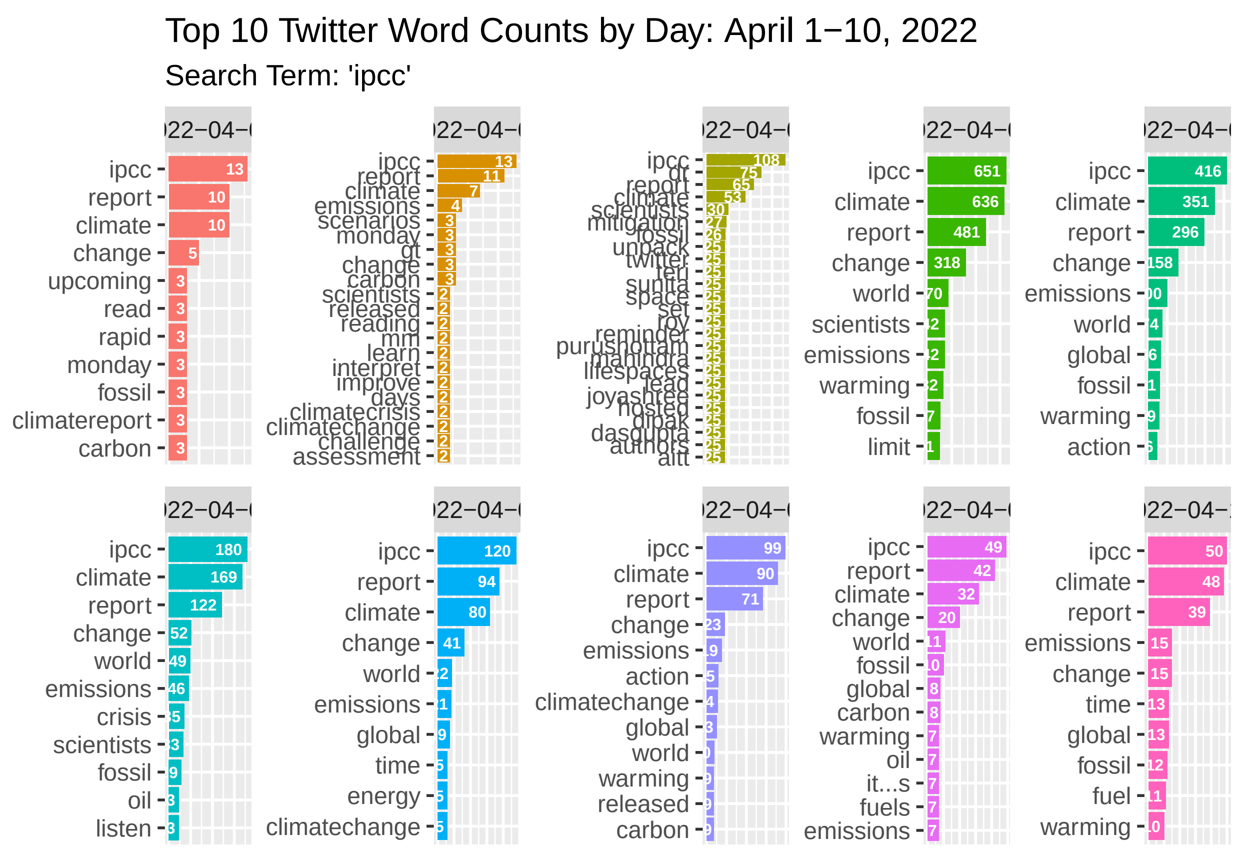
```r
# create frequency dataframe
tweet_freq <- words_tokenized %>%
  group_by(date) %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>%
  arrange(date, -n) %>%
  top_n(n=10, wt = n) %>% # here we grap the top 10
  ungroup()

# top 10 plot setup
tweet_freq_plot <- tweet_freq %>%
  mutate(name = reorder_within(word, n, date)) %>%

  ggplot(aes(n, name, fill = as.factor(date))) +
  geom_col() +
  geom_text(aes(label = n), hjust = 1.2, colour = "white", fontface = "bold", size = 2) +
  facet_wrap(~date, scales = "free", ncol = 5) +
  scale_y_reordered() +
  labs(title = "Top 10 Twitter Word Counts by Day: April 1-10, 2022",
       subtitle = "Search Term: 'ipcc'",
       y = NULL) +

  # remove x-axis text
  theme(legend.position="none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

tweet_freq_plot
```



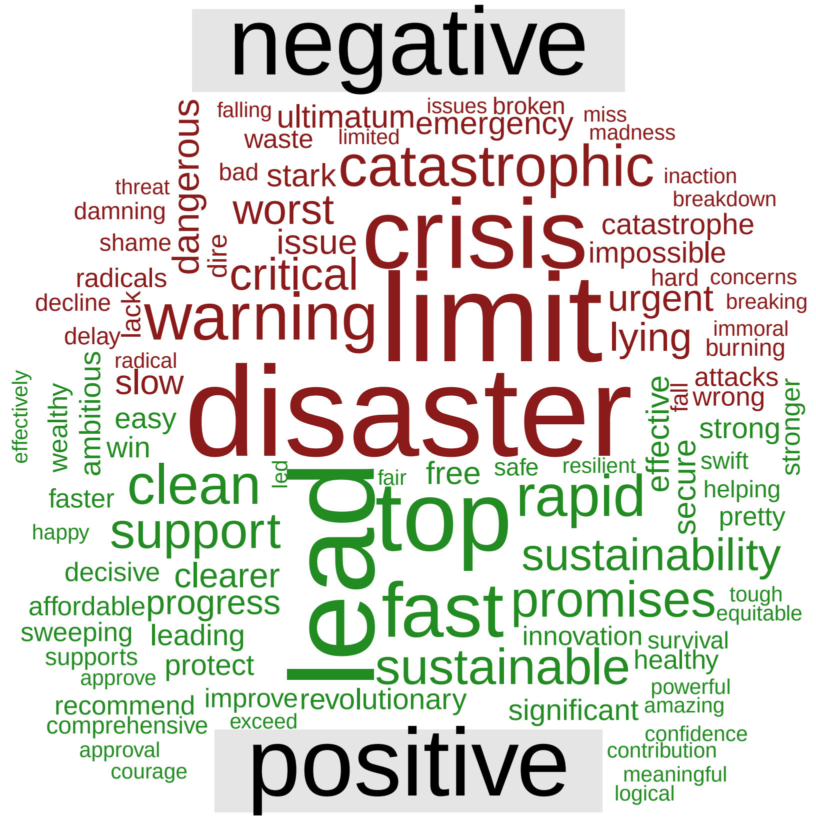
From this we can see words like **'ipcc', 'report', 'climate', and 'change'** seem to be our top contenders. We know that April 4, 2022 corresponds to an IPCC Working Group 3 press conference. From this we generally have low occurrences of each word 2+ days out from the IPCC event. On the day of the event, April 4, 2022 we see word frequency counts around 600+. After the event word mentions around "IPCC" gradually decline.

One thing to notice how many days have more than 10 words. Looks like this occurs when two or more words have the same number of counts. For example, on April 10, 2022, **emissions** and **change** both were tweeted 15 times that day.

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

Let's choose red for negative words and green for positive words.

```
words_tokenized %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
acast(word ~ sentiment, value.var = "n", fill = 0) %>%
comparison.cloud(colors = c("firebrick4", "forestgreen"),
                 max.words = 100)
```



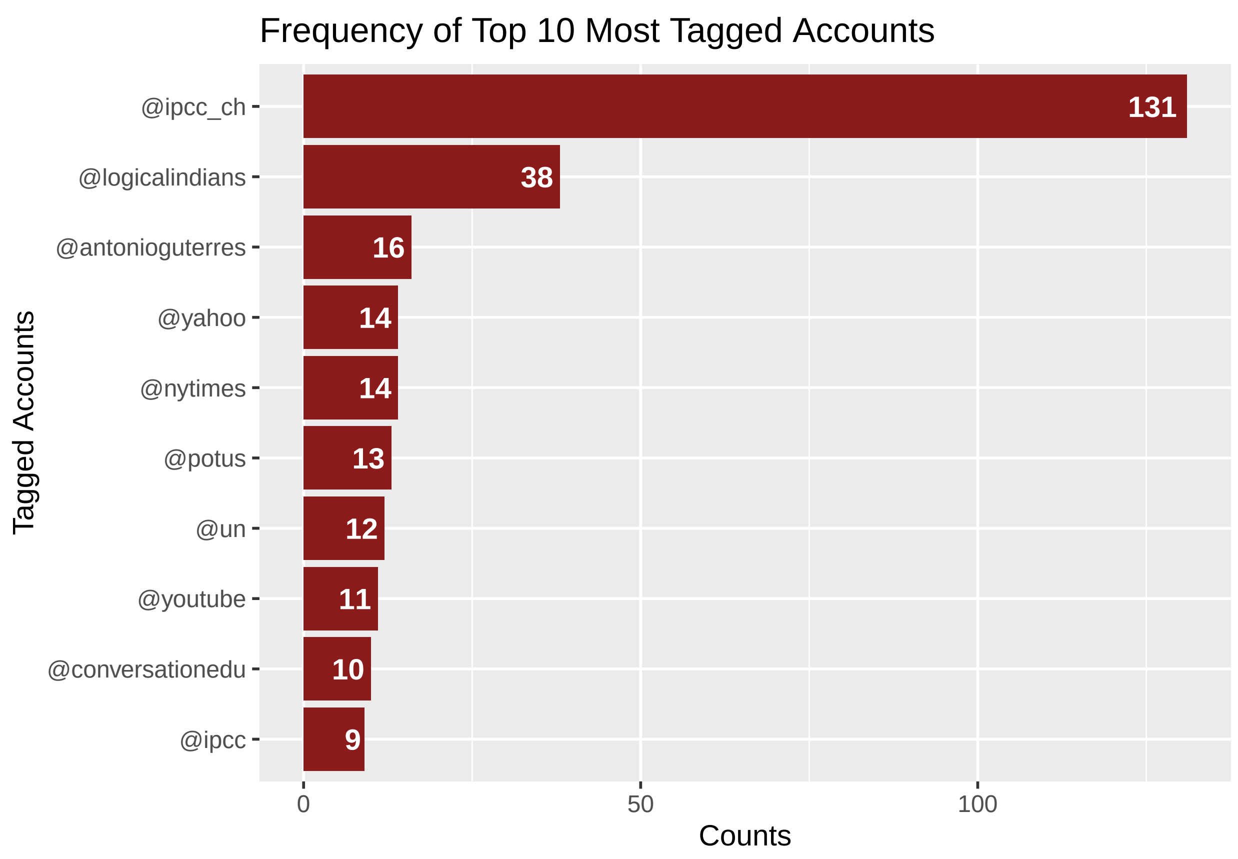
Part 4. Let's say we are interested in the most prominent entities in the Twitter discussion. Which are the top 10 most tagged accounts in the data set.

For this part we'll use `quantdata`. quanteda is 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.

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

account_tweets <- tokens(x = corpus, remove_punct = TRUE) %>%
  tokens_keep(pattern = "@*") %>%
  dfm() %>%
  textstat_frequency(n = 10)
```

```r
ggplot(data = account_tweets,
       aes(x = frequency,
           y = reorder(feature, frequency))) +
  geom_col(fill = "firebrick4") +
  geom_text(aes(label = frequency), hjust = 1.2, colour = "white", fontface = "bold") +
  labs(x = "Counts",
       y = "Tagged Accounts",
       title = "Frequency of Top 10 Most Tagged Accounts")
```



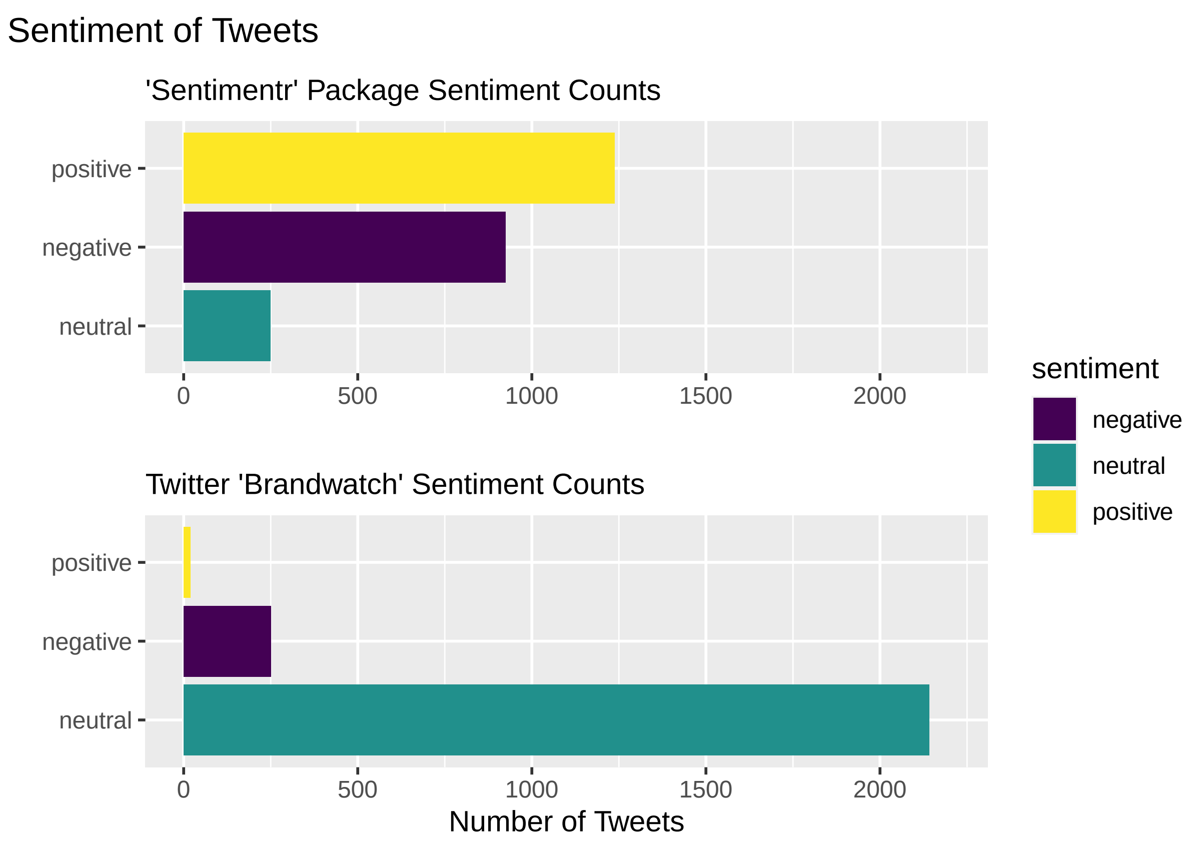
Part 5. The Twitter data download comes with a variable called "Sentiment" that must be calculated by Brandwatch. Use your own method to assign each tweet a polarity score (Positive, Negative, Neutral) and compare your classification to Brandwatch's (hint: you'll need to revisit the "raw_tweets" data frame).

```r
words_sentiment <- words_tokenized %>%
  anti_join(stop_words, by = "word") %>%
  left_join(get_sentiments('bing'), by = "word") %>%
  left_join(tribble(
    ~sentiment, ~sent_score,
    "positive", 1,
    "negative", -1), by = "sentiment")
```

```r
# Calculate sentiment and plot
sent_plot1 <- get_sentences(dat$Title) %>%
  sentiment() %>%
  group_by(element_id) %>%
  summarize(sentiment_score = mean(sentiment)) %>%
  mutate(sentiment = case_when(sentiment_score < 0 ~ "negative",
                               sentiment_score == 0 ~ "neutral",
                               sentiment_score > 0 ~ "positive")) %>%
  group_by(sentiment) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = count,
             y = reorder(sentiment, count),
             fill = sentiment)) +
  geom_col() +
  scale_fill_viridis_d() +
  xlim(NA, 2200) +
  labs(x = "",
       y = "",
       subtitle = "'Sentimentr' Package Sentiment Counts")


# Plot Brandwatch sentiment scores
sent_plot2 <- clean_tweets %>%
  filter(sentiment %in% c("positive", "negative", "neutral")) %>%
  group_by(sentiment) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = count,
             y = reorder(sentiment, count),
             fill = sentiment)) +
  geom_col() +
  scale_fill_viridis_d() +
  xlim(NA, 2200) +
  scale_y_discrete(limits = rev) +
  labs(x = "Number of Tweets",
       y = "",
       subtitle = "Twitter 'Brandwatch' Sentiment Counts")

(sent_plot1 / sent_plot2) +
  plot_annotation(title = "Sentiment of Tweets") +
  plot_layout(guides = "collect")
```


Sentiment of Tweets
'Sentimentr' Package Sentiment Counts
positive
negative
neutral
0
500
1000
1500
2000
Twitter 'Brandwatch' Sentiment Counts
positive
negative
neutral
0
500
1000
1500
2000
Number of Tweets
sentiment
negative
neutral
positive