EDS 231: Sentiment Analysis I

Juliet Cohen

4/19/2022

```
library(tidyr) #text analysis in R
library(lubridate) #working with date data
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(pdftools) #read in pdfs
```

```
## Using poppler version 22.02.0
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v dplyr   1.0.7
## v tibble  3.1.6     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
library(tidytext)
library(here)
```

```
## here() starts at /Users/juliet/Documents/MEDS/Text_Analysis/Text_Analysis
```

```
library(LexisNexisTools) #Nexis Uni data wrangling
```

```
## LexisNexisTools Version 0.3.5
```

```
library(sentimentr)
library(readr)
library(textdata)
```

0: Using the "IPCC" Nexis Uni data set from the class presentation and the pseudo code we discussed, recreate Figure 1A from Froelich et al. (Date x # of 1) positive, 2) negative, 3) neutral headlines)

```
# read in nexis data, needs to be specific steps or else errors
nexis <- "nexis_dat.docx"
nexis_data <- lnt_read(nexis)
```

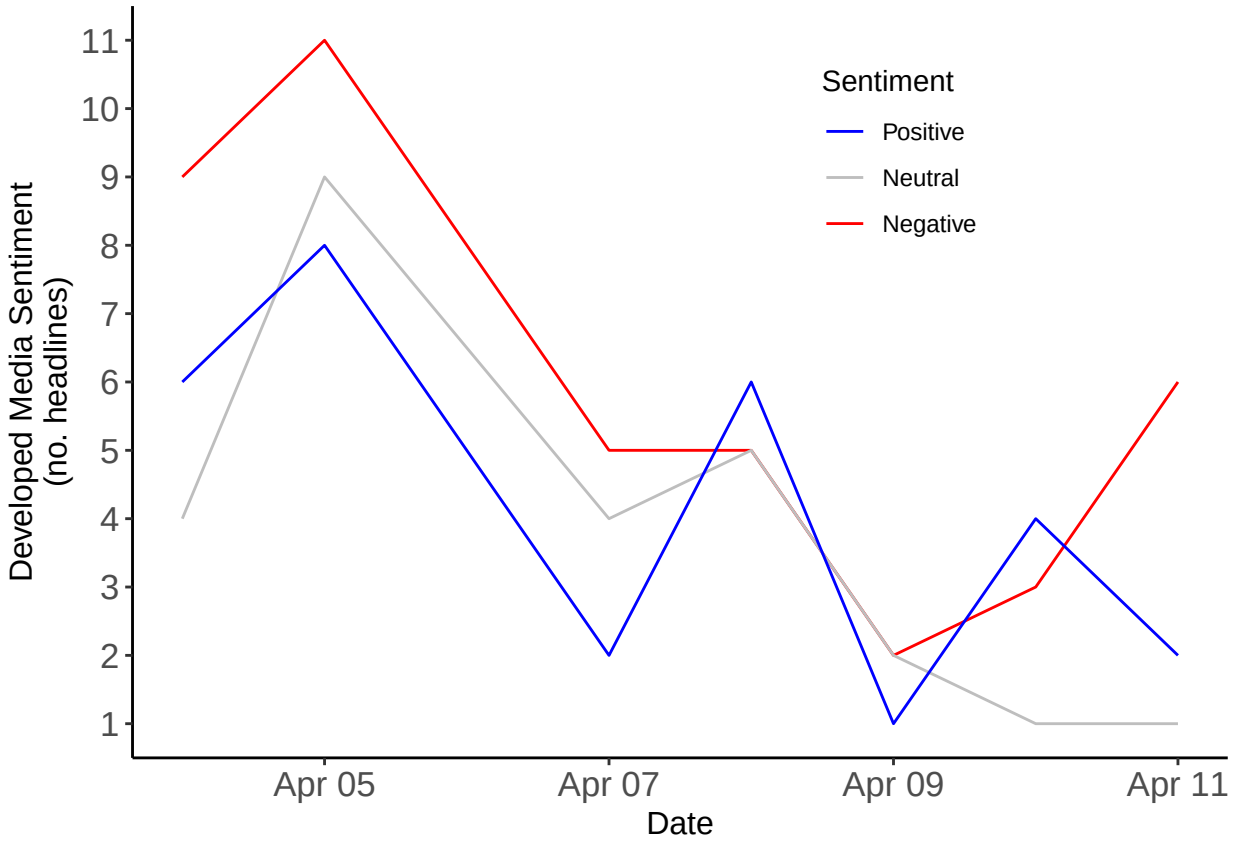
```
## Creating LNToutput from 1 file...
##  ...files loaded [0.15 secs]
##  ...articles split [0.18 secs]
##  ...lengths extracted [0.19 secs]
##  ...headlines extracted [0.19 secs]
##  ...newspapers extracted [0.19 secs]
##  ...dates extracted [0.20 secs]
##  ...authors extracted [0.21 secs]
##  ...sections extracted [0.21 secs]
##  ...editions extracted [0.21 secs]
##  ...dates converted [0.21 secs]
##  ...metadata extracted [0.22 secs]
##  ...article texts extracted [0.22 secs]
##  ...superfluous whitespace removed [0.25 secs]
## Elapsed time: 0.25 secs
```

```
nexis_metadata <- nexis_data@meta

# convert nexis_metadata into a dataframe
nexis_df <- data_frame(element_id = seq(1:length(nexis_metadata$Headline)),
                               nexis_date = nexis_metadata$Date,
                               nexis_headlines = nexis_metadata$Headline)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
# retreieve bing sentiments
bing_sentiments <- get_sentiments('bing')

nexis_sentiment <- nexis_df$nexis_headlines %>%
  get_sentences() %>%
  sentiment() %>%
  inner_join(nexis_df, by = "element_id") %>%
  mutate(sent_category = case_when(
      sentiment < 0 ~ "Negative",
      sentiment > 0 ~ "Positive",
      T ~ "Neutral")) %>%
  count(sent_category, nexis_date)
```

```r
# define limits for y axis
min_count <- min(nexis_sentiment$n)
max_count <- max(nexis_sentiment$n)

ggplot(data = nexis_sentiment, aes(x = nexis_date, y = n, color = sent_category)) +
  geom_line() +
  labs(y = "Developed Media Sentiment\n(no. headlines)",
       x = "Date") +
  labs(color = "Sentiment") +
  scale_y_continuous(breaks = seq(min_count, max_count, by = 1)) +
  scale_color_manual(values = c("Positive" = "blue", "Neutral" = "grey", "Negative" = "red")) +
  theme_classic() +
  theme(legend.position = c(0.7, 0.8),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 13))
```



Graph inspiration from Froelich et al. 2017

1: Access the Nexis Uni database through the UCSB library: https://www.library.ucsb.edu/research/db/211

Done

2: Choose a key search term or terms to define a set of articles.

Search term = "leopard+shark"

3: Use your search term along with appropriate filters to obtain and download a batch of at least 100 full text search results (.docx).

Done

4: Read in the 'leopard shark' text data downloaded from Nexis Uni database through the UCSB library

```
leopard_shark_filepath <- list.files(pattern = "leopard_shark_articles.docx", path = getwd(),
                       full.names = TRUE, recursive = TRUE, ignore.case = TRUE)

data <- lnt_read(leopard_shark_filepath)
```

```
## Creating LNToutput from 1 file...
##  ...files loaded [0.16 secs]
##  ...articles split [0.20 secs]
##  ...lengths extracted [0.20 secs]
##  ...headlines extracted [0.20 secs]
##  ...newspapers extracted [0.20 secs]
##  ...dates extracted [0.22 secs]
##  ...authors extracted [0.22 secs]
##  ...sections extracted [0.22 secs]
##  ...editions extracted [0.22 secs]
## Warning in lnt_asDate(date.v, ...): More than one language was detected. The
## most likely one was chosen (English 98%)
##  ...dates converted [0.23 secs]
##  ...metadata extracted [0.23 secs]
##  ...article texts extracted [0.23 secs]
##  ...superfluous whitespace removed [0.27 secs]
## Elapsed time: 0.27 secs
```

5: Use the full text for analysis

```
meta_df <- data@meta
articles_df <- data@articles
paragraphs_df <- data@paragraphs

data2 <- data_frame(element_id = seq(1:length(meta_df$Headline)), Date = meta_df$Date, Headline = meta_

paragraphs_data <- data_frame(element_id = paragraphs_df$Art_ID, Text  = paragraphs_df$Paragraph)

# join the headlines with the paragraphs to access the full text
data3 <- inner_join(data2, paragraphs_data, by = "element_id")
# this df contains columns for the headline of the articles and the paragraphs, with each paragraph as
```

Clean the data and remove polaized words from the sentiment words

```
# make a new column that states TRUE if the row has the string in it
data3_clean <- data3 %>%
  mutate(http_present = grepl(pattern = "http", x = Text))
# nrows = 5250

# check how many rows were TRUE and how many were FALSE
T_F_summary <- data3_clean %>%
  group_by(http_present) %>%
  summarize(sum = n()) # 5190 rows should be maintained because they do NOT contain the undesired strin

# remove all rows with that string
data3_clean <- data3_clean %>%
  filter(http_present == FALSE)

# check that it worked
unique(data3_clean$http_present)
```

```
## [1] FALSE
```

```
# grab the bing sentiment lexicon from tidytext
nrc_sentiment <- get_sentiments('nrc')
head(nrc_sentiment, n = 20)
```

```
## # A tibble: 20 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
## 11 abandonment negative
## 12 abandonment sadness
## 13 abandonment surprise
## 14 abba        positive
## 15 abbot       trust
## 16 abduction   fear
## 17 abduction   negative
## 18 abduction   sadness
## 19 abduction   surprise
## 20 aberrant    negative
```

```
data3_clean_words <- data3_clean  %>%
  select(!http_present) %>%
  unnest_tokens(output = word, input = Text, token = 'words')

data3_clean_sentiment_words <- data3_clean_words %>%
  anti_join(stop_words, by = 'word') %>% # remove stop words
```

```
  inner_join(nrc_sentiment, by = 'word') %>% # keeps only sentiment words
  filter(!sentiment %in% c("negative", "positive")) # remove the polarized words
```

6: Data Exploration: graph annual raw count of each sentiment word

```
graph_data <- data3_clean_sentiment_words %>%
  mutate(year = substring(Date, 1, 4)) %>%
  select(!Date) %>%  # remove Date col because we only care about year, which is a column we retain
  group_by(year, sentiment) %>% # unique years in this df are 2017, 2019, 2020, 2021, 2022, and NA
  summarise(count = n()) %>%
  drop_na() # drop na rows, we need the year for all observations to make graph
```

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

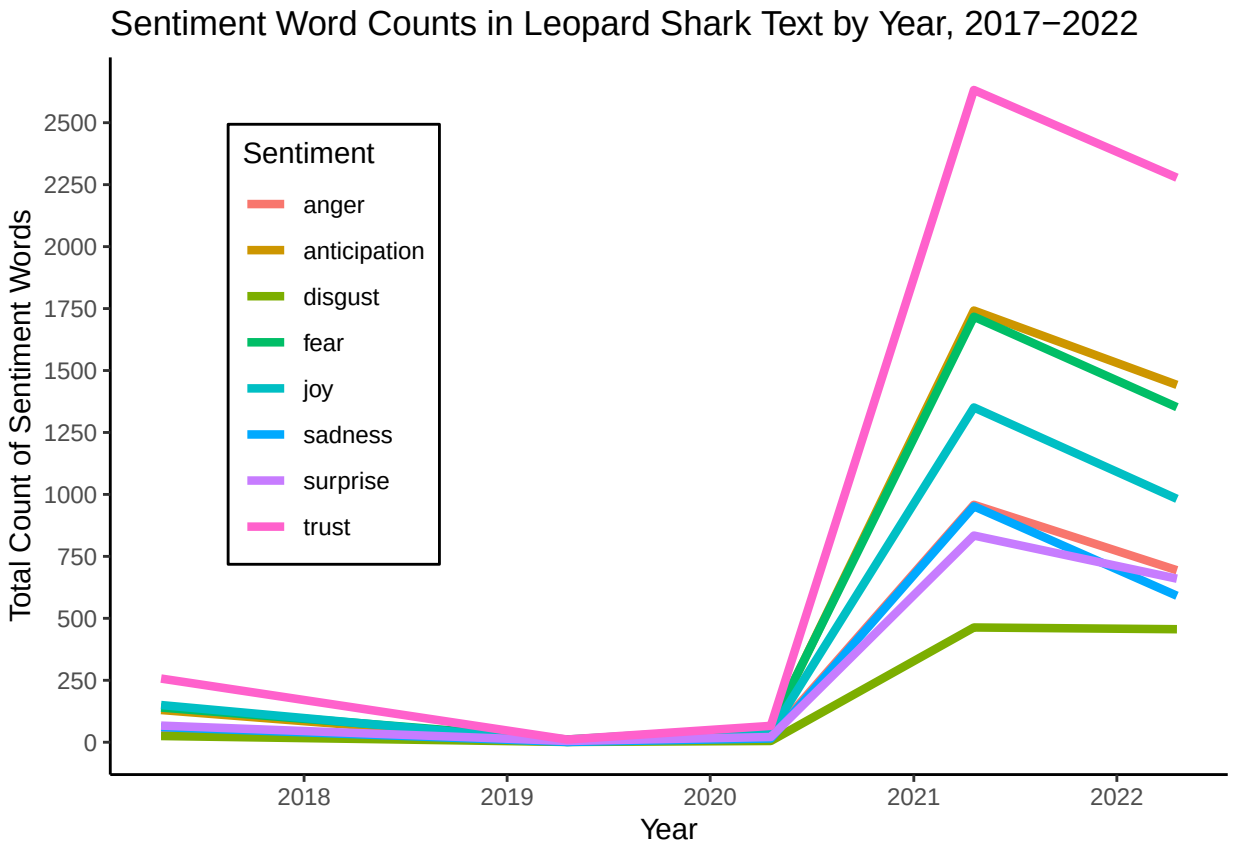
```
max_count <- max(graph_data$count)

class(graph_data$year) # character
```

```
## [1] "character"
```

```
graph_data$year <- as.Date(graph_data$year, format = "%Y")
class(graph_data$year) # Date
```

```
## [1] "Date"
```

```
ggplot(data = graph_data, aes(x = year, y = count, color = sentiment)) +
  geom_line(lwd = 1.5) +
  theme_classic() +
  scale_y_continuous(breaks = seq(0, (max_count +10), by = 250)) +
  theme(legend.position = c(0.2, 0.6),
        legend.background = element_rect(size=0.5,
                                         linetype="solid",
                                         color = "black")) +
  labs(y = "Total Count of Sentiment Words",
       x = "Year",
       title = "Sentiment Word Counts in Leopard Shark Text by Year, 2017-2022") +
  guides(color = guide_legend()) +
  labs(color = "Sentiment")
```



7: Plot the amount of emotion words (the 8 from nrc) as a percentage of all the emotion words used each day (aggregate text from articles published on the same day). How does the distribution of emotion words change over time? Can you think of any reason this would be the case?

```
graph_data_prep <- data3_clean_sentiment_words %>%
  drop_na() %>%
  group_by(Date, sentiment) %>%
  summarise(daily_count = n())
```

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

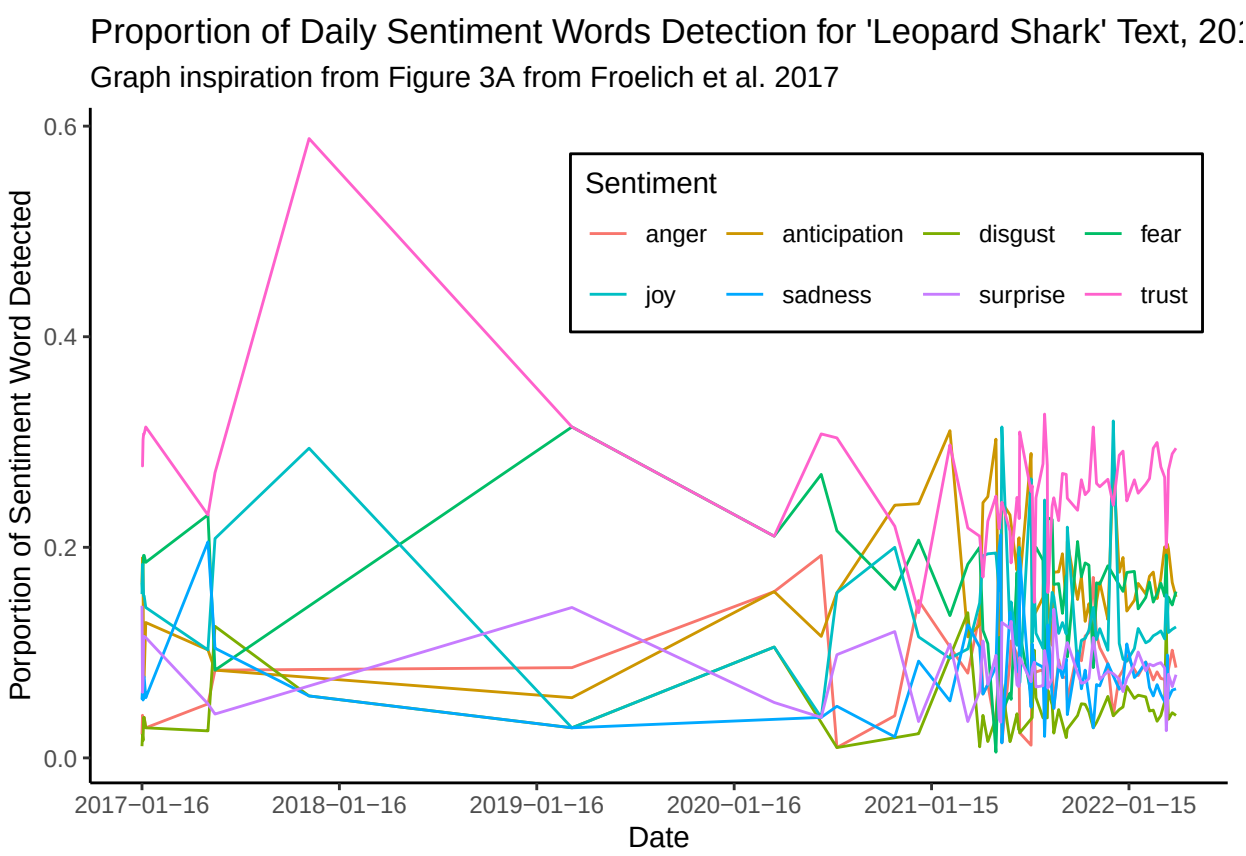
```
daily_total_words <- graph_data_prep %>%
  group_by(Date) %>%
  summarise(total_daily_words = sum(daily_count))

graph_data <- left_join(graph_data_prep, daily_total_words, by = "Date")

graph_data_proportions <- graph_data %>%
  group_by(sentiment, Date) %>%
  summarise(proportion = (daily_count / total_daily_words))
```

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

```
# ensure Date col is of class Date, a requirement for line graphs
class(graph_data_proportions$Date) # Date
```

```
## [1] "Date"
```

```
earliest_date <- min(graph_data_proportions$Date)
latest_date <- max(graph_data_proportions$Date)

ggplot(data = graph_data_proportions, aes(x = Date, y = proportion, color = sentiment)) +
  geom_line() +
  theme_classic() +
  theme(legend.position = c(0.7, 0.8),
        legend.background = element_rect(size=0.5,
                                         linetype="solid",
                                         color = "black")) +
  guides(color = guide_legend(nrow = 2, byrow = TRUE)) +
  scale_x_continuous(breaks = seq(earliest_date, latest_date, by = 365)) +
  labs(y = "Porportion of Sentiment Word Detected",
       x = "Date",
       title = "Proportion of Daily Sentiment Words Detection for 'Leopard Shark' Text, 2017-2022",
       subtitle = "Graph inspiration from Figure 3A from Froelich et al. 2017",
       color = "Sentiment")
```



The distribution of emotion words starts off with fewer observations in the earlier years (2017-2019), and during these years **trust** is the most common sentiment while **disgust** and then **sadness** are the least common sentiments. Starting in 2020 and continuing into 2022, the sentiment observations for "leopard shark" become much more frequent, and **trust** is still the most common sentiment, even though the proportion of **trust** detected in the text decreases compared to the earlier years. However, **disgust** is consistently

the least common sentiment during these later years. I believe an explanation for these trends can be found in the increase in environmental awareness and policies protecting marine habitat in recent years. Additionally, I speculate that more people were talking about leopard sharks starting during the pandemic because perhaps this species was more active in coastal waters with fewer tourists and less water pollution during the lockdowns. Perhaps an explanation for the exceptionally low proportion of **disgust** in 2020-2022 can be found in that fact that people became more aware of environmental issues, species extinction, and the need to talk about the issues on a public platform in a positive, hopeful way. Lastly, **sadness** seems to fluctuate a lot in 2020-2022 while it did not fluctuate much in the earlier years. This would make sense during the pandemic because many people experienced emotional instability in general.