Assignment 2

Colleen McCamy

April 15, 2023

Sentiment analysis is a tool for assessing the mood of a piece of text. For example, we can use sentiment analysis to understand public perceptions of topics in environmental policy like energy, climate, and conservation.

Reading in Nexis Article Data (1-4)

```
text_file_path <- "/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/labs/data"

my_files <- list.files(pattern = ".docx", path = text_file_path,
                       full.names = TRUE, recursive = TRUE, ignore.case = TRUE)

# setting directory to save
setwd("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sentiment-assignmen
saveRDS(my_files, "decarb_text_data.RDS") # will need to do this four our portion

# reading in the data as an RDS
dat <- readRDS("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sentiment-

# changing to a LNT output
dat <- lnt_read(dat)

# saving individual dataframes from the LNT output
meta <- dat@meta
articles_df <- dat@articles
paragraphs_df <- dat@paragraphs

write_csv(articles_df, "data/articles_df.csv")
write_csv(meta, "/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sentimen
```

Cleaning Artifacts about the Data (5)

```
# reading in the articles & meta dataframe (to skip the first chunk when knitting)
articles_df <- read_csv("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-s
meta <- read_csv("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sentimen

# define the word string to split at for the query information at the end
split_word <- "Classification Language:"
```

```
# remove the classification language sub-text at the end of each article
articles_df$split_text <- unlist(lapply(strsplit(articles_df$Article, split_word), '[', 1))

# removing the query info from the paranthesis
articles_df <- articles_df |>
  mutate(text_noq = gsub("^\\[[0-9]+\\] |\\([^\\(]*\\)", "", split_text))
```

Exploring the Data (6)

```
# putting the format into tidy text for analysis
text_words <- articles_df |>
  tidytext::unnest_tokens(output = word,
                input = text_noq,
                token = 'words') |>
  select(-("split_text"))


# loading in stop words
stop <- stop_words

text_words_clean <- text_words |>
  anti_join(stop_words, by = "word")

# loading in the bing sentiment words
bing_sent <- get_sentiments('bing')

# adding the sentiment words and score to the dataframe
sent_words <- text_words_clean |>
  inner_join(bing_sent, by = "word") |>
  mutate(sent_num = case_when(sentiment == 'negative' ~ -1,
                              sentiment == 'positive' ~ 1))
```

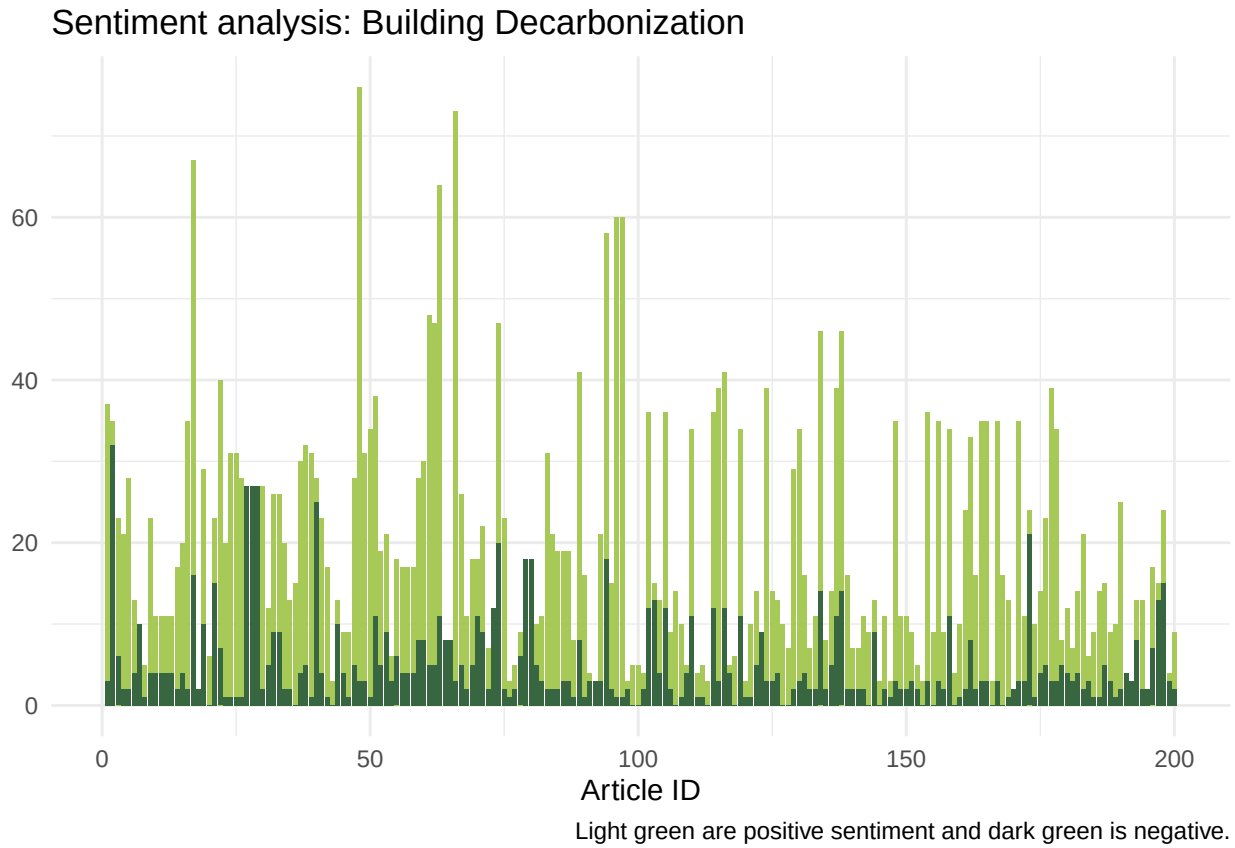
Calculating Mean Sentiment

```
# take the average and look at all of the articles
sent_article <- sent_words |>
  group_by(ID) |>
  count(ID, sentiment) |> # counts the words
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) |>  # pivoting wider
  mutate(polarity = positive - negative)

# calculating the mean sentiment or total polarity
mean_sent <- mean(sent_article$polarity)

print(paste0("The mean sentiment of the articles for the term 'building decarbonization' is ", mean_sent
```

```
## [1] "The mean sentiment of the articles for the term 'building decarbonization' is 14.6."
```

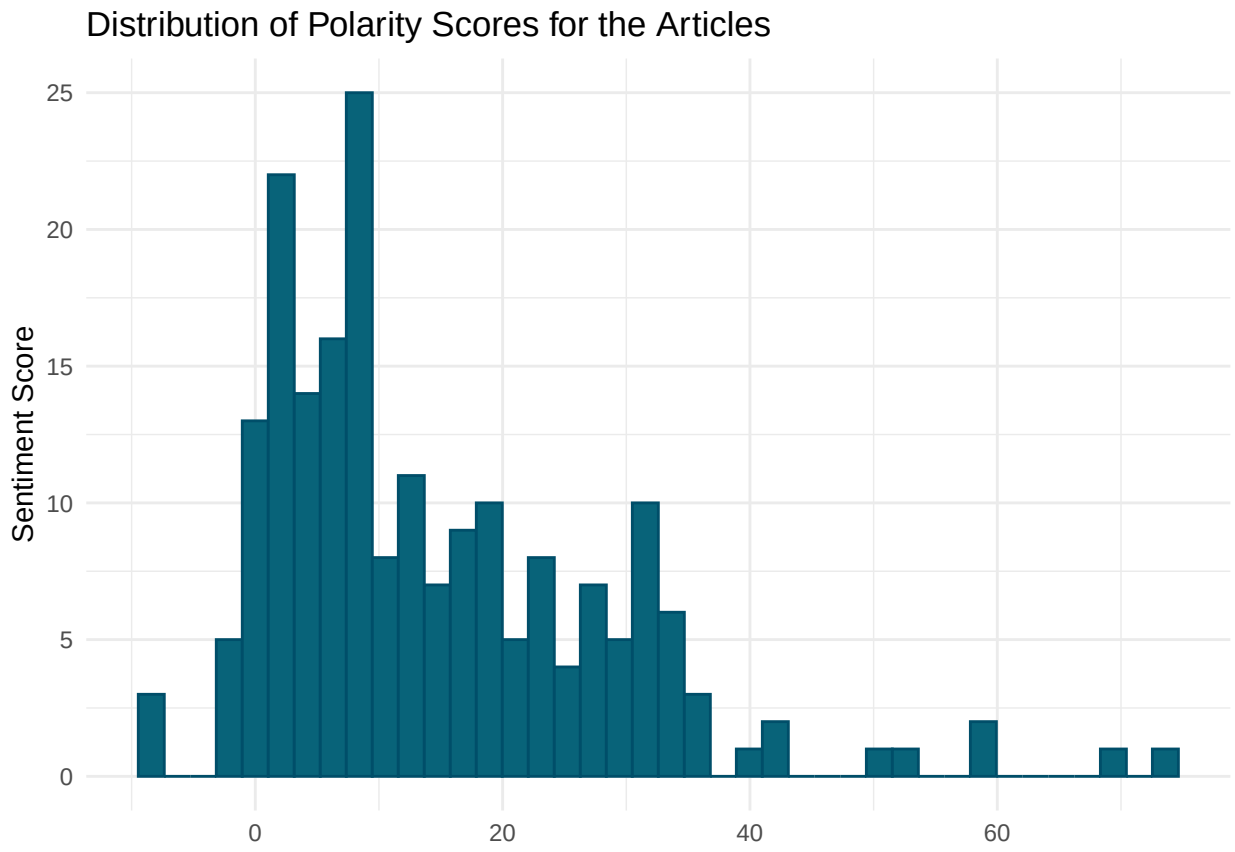
Sentiment by Article Plot & Plotting Polarity

```
ggplot(sent_article, aes(x = ID)) +
  theme_minimal() +
  geom_col(aes(y = positive),
           stat = 'indentity',
           fill = "#a7c957") +
  geom_col(aes(y = negative),
           stat = 'indentity',
           fill = "#386641") +
  theme(axis.title.y = element_blank()) +
  labs(title = "Sentiment analysis: Building Decarbonization",
       y = "Sentiment score",
       x = "Article ID",
       caption = "Light green are positive sentiment and dark green is negative.") +
  scale_fill_manual(values = c("#a7c957", "#386641"),
                    labels = c("Positive", "Negative"))
```



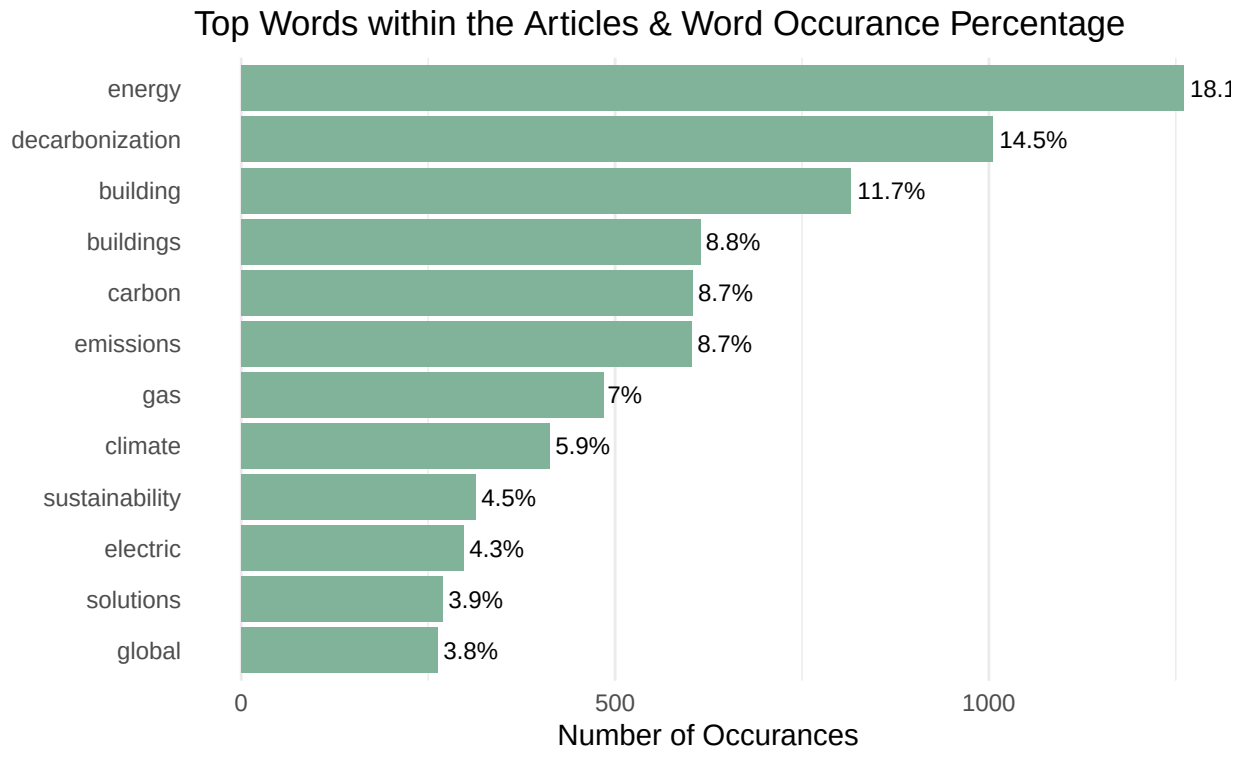

```
# plotting the distribution of polarity scores graph
ggplot(sent_article, aes(x = polarity)) +
  geom_histogram(col = "#004e69",
                 fill = "#086379",
                 bins = 40) +
  theme_minimal() +
```

```
  labs(title = "Distribution of Polarity Scores for the Articles",
       y = "Sentiment Score",
       x = NULL)
```



Plotting Top Words Occurance Numbers & Percentage

```
# plotting the top words used
p_top_words <- text_words_clean |>
  count(word, sort = TRUE) |>
  filter(n > 250) |>
  mutate(word = reorder(word, n),
         percent = paste0(round((n / sum(n)) * 100, 1), "%")) |>
  ggplot(aes(n, word)) +
  geom_col(fill = "#81b29a") +
  geom_text(aes(label = percent), hjust = -0.1, size = 3) +
  theme_minimal() +
  labs(x = "Number of Occurances \n",
       y = NULL,
       title = "Top Words within the Articles & Word Occurance Percentage",
       caption = "Displaying the occurance percentage for the word \n in all of the articles excluding s
  theme(panel.grid.major.y = element_blank())  # remove major x-axis lines

p_top_words
```



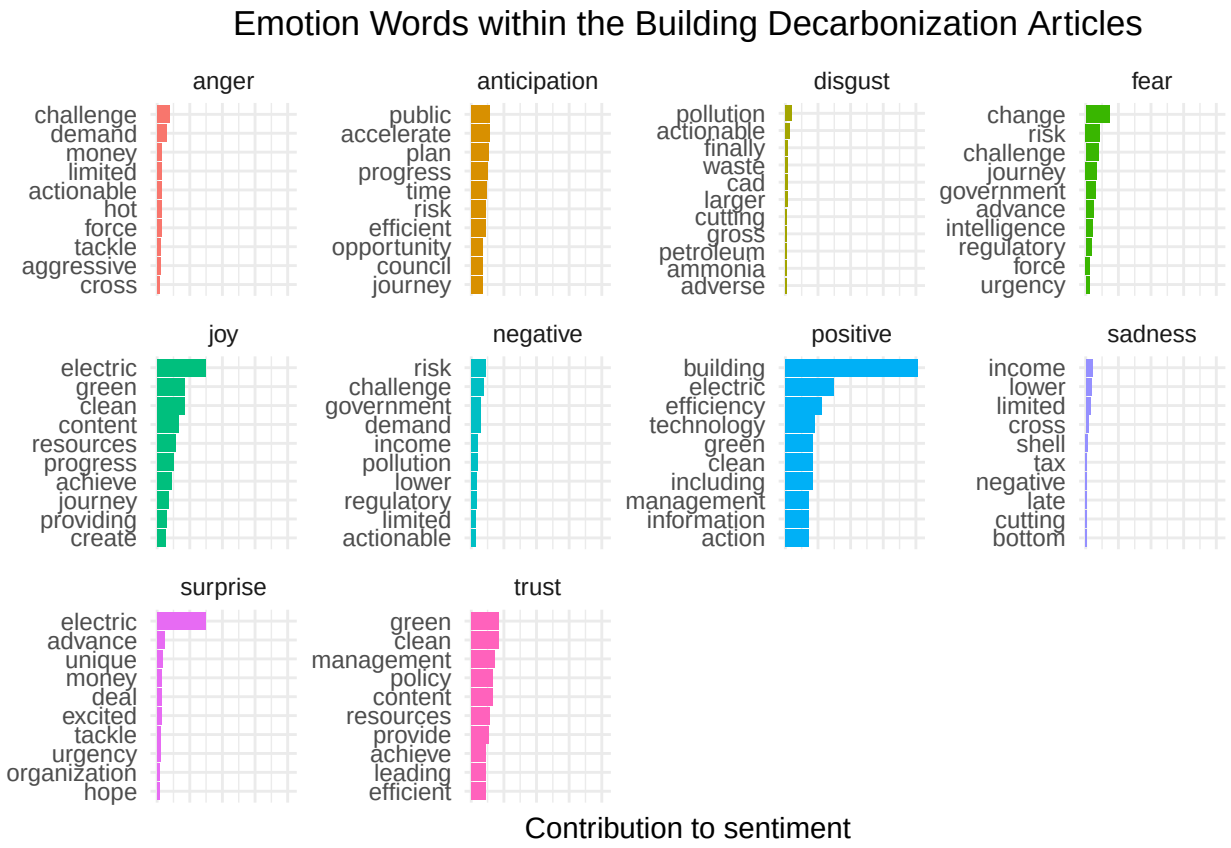
Displaying the occurance percentage for the word in all of the articles excluding stop words.

NRC Emotion Word Analysis (7)

```
# loading in the nrc sentiment words
nrc_sent <- get_sentiments("nrc")

# joining the sentiment words to the total text words
nrc_word_counts <- text_words_clean |>
  inner_join(nrc_sent) |>
  count(word, sentiment, sort = T) |>
  ungroup()

# grouping by sentiment and saving this to be grouped by sentiment
sent_counts_nrc <- text_words_clean |>
  group_by(ID) |>
  inner_join(nrc_sent) |>
  group_by(sentiment) |>
  count(word, sentiment, sort = T)
```

```
# plotting the top words for each emotion from all of the articles
sent_counts_nrc |>
  group_by(sentiment) |>
  slice_max(n, n = 10) |>
  ungroup() |>
  mutate(word = reorder(word, n)) |>
```

```
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") + # plot for each emotion
  labs(x = "Contribution to sentiment",
       y = NULL,
       title = "Emotion Words within the Building Decarbonization Articles") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank())
```

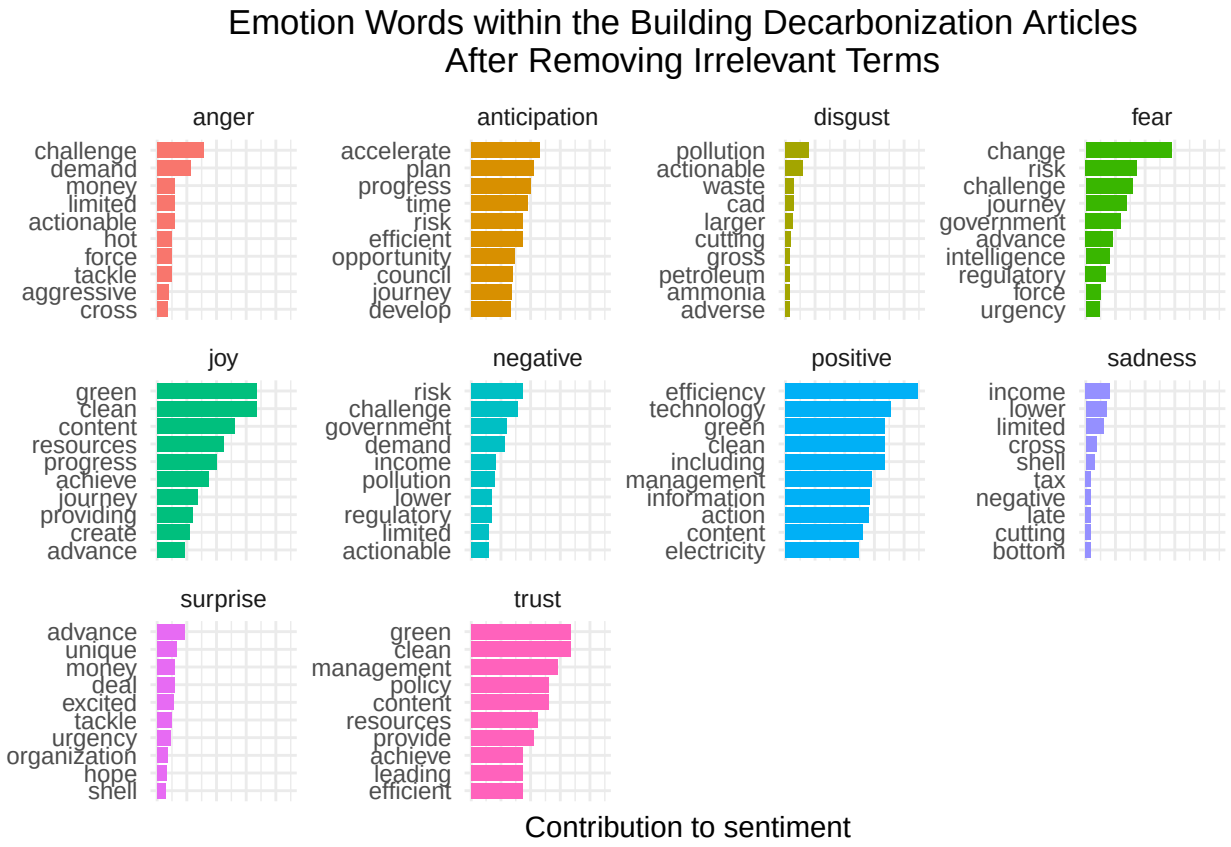


Removing Misleading NRC Words

```
# creating a dataframe of words to remove
nrc_remove <- tibble(
  word = c("building", "electric", "finally", "public"),
  sentiment = c(0, 0, 0, 0))

# removing the words listed above from the nrc words
nrc_clean <- nrc_sent |>
  anti_join(nrc_remove, by = "word")
```

```
# updating the sentiment counts with the irrelevant words taken out
sent_counts_nrc <- text_words_clean |>
```

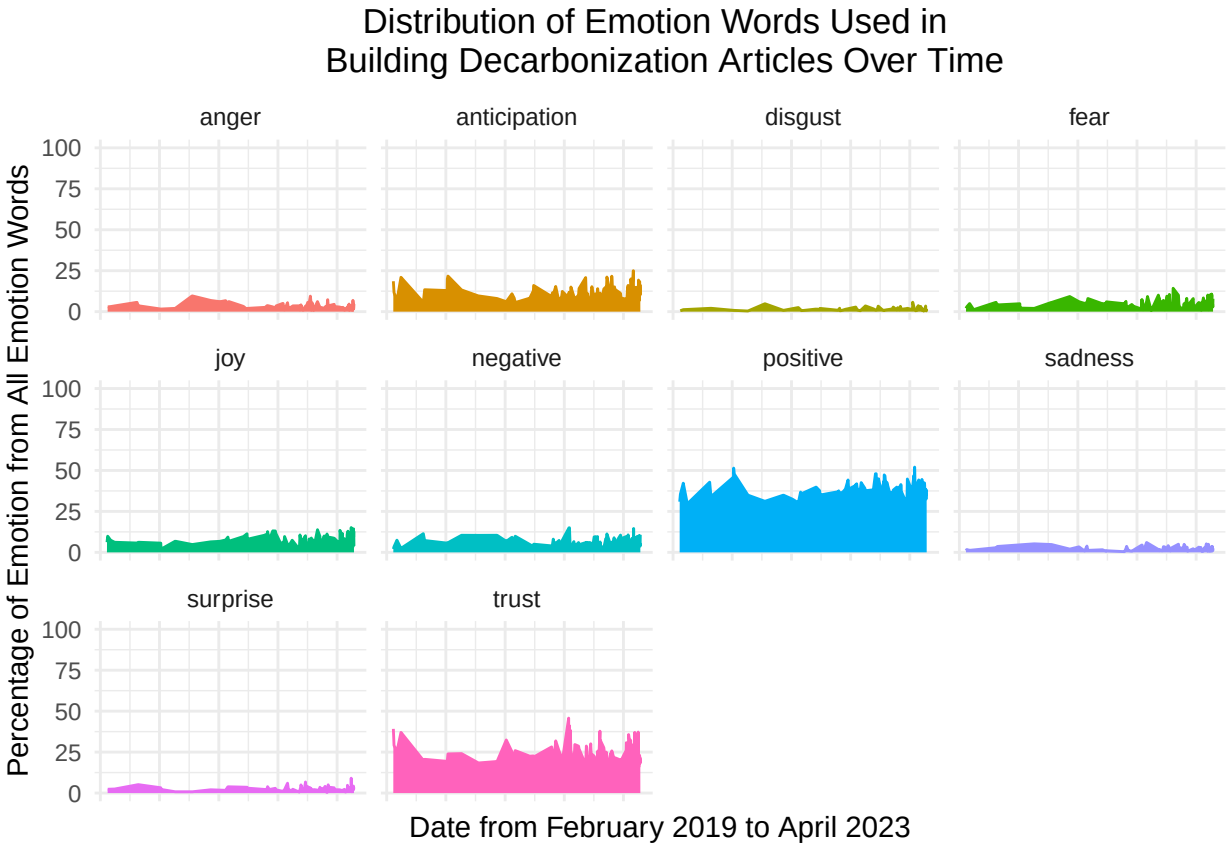
```
  group_by(ID) |>
  inner_join(nrc_clean) |>
  group_by(sentiment) |>
  count(word, sentiment, sort = T)

# plotting with the new words taken out
sent_counts_nrc |>
  group_by(sentiment) |>
  slice_max(n, n = 10) |>
  ungroup() |>
  mutate(word = reorder(word, n)) |>
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") + # plot for each emotion
  labs(x = "Contribution to sentiment",
       y = NULL,
       title = "Emotion Words within the Building Decarbonization Articles \n After Removing Irrelevant
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank())
```



Plotting the amount of nrc emotion words as a percentage of all the emotion words used each day (8)

```
# selecting the date and ID column from the metadata
article_date <- meta |>
  select(ID, Date)

# joining the date articles data
text_words_date <- full_join(text_words_clean, article_date, by = "ID")

# grouping by sentiment said per day
sent_counts_date <- text_words_clean |>
  group_by(ID) |>
  inner_join(nrc_clean) |>
  full_join(article_date, by = "ID") |>
  group_by(sentiment, Date) |>
  count(sentiment)

# getting total words for the day
total_words_date <- sent_counts_date |>
  group_by(Date) |>
  summarize(total_words = sum(n))

# adding total emotional words for the day as a column and calculating the percentage
sent_counts_date <- sent_counts_date |>
  full_join(total_words_date, by = "Date")

# adding the percentage column
sent_counts_date <- sent_counts_date |>
  mutate(percent = round(((n/total_words)*100), 2),) |>
  mutate(Date = as.Date(Date, format = "%B %d, %Y"))
```

```
# plotting sentiment change over time
sent_counts_date |>
  ggplot(aes(Date, percent, color = sentiment, fill = sentiment, group = sentiment)) +
  geom_area(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "fixed") +
  labs(x = "Date from February 2019 to April 2023",
       y = "Percentage of Emotion from All Emotion Words",
       title = "Distribution of Emotion Words Used in \n Building Decarbonization Articles Over Time") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank()) +
  ylim(c(0, 100))
```



How does the distribution of emotion words change over time? Can you think of any reason this would be the case?

RESPONSE: It appears that the distribution of emotion words used in the building decarbonization articles tends to be more variable as time advances. This is shown through more rigid percentages of the emotions overtime. This is more apparent in the trust, joy, anger and anticipation graphs. This could be because as time advances more articles have been published about building decarbonization. More articles published allows for a greater variability of emotions published about the subject.

However, it doesn't appear that there is any major changes overtime for emotions in articles that contain the word "building decarbonization". I think this is because the term itself is more industry-focused jargon. This analysis can inform us that other key words may be better suited to assess the change in emotions about the decarbonization of buildings.