Assignment 1: Text Data in R Using NYT API

Halina Do-Linh

04/10/2022

For my assignment, I chose the key word "dinosaur" to query from the NYT API.

From this API call, I received 10 articles with 33 variables. I also had to convert the object from a list to a data frame.

```
# create an object t with the results of query ("dinosaur")
t <- fromJSON(paste0("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=",
                     "dinosaur",
                     "&api-key=", "mrb3Gx3ed9yP1KDVybtq9F2yEXWIgAga"),
                     flatten = TRUE)

class(t) # t is a list object

# convert t to df
t <- t %>% data.frame()

# inspect data
dim(t) # 10 x 33

# what variables are we working with?
names(t) # 33 total
```

Here I changed the API call to obtain more articles/data. I decided to query based on the dates 2021/01/01 to 2022/04/09, so a little more than one year's worth of data.

```
term <- "dinosaur"
begin_date <- "20210101"
end_date <- "20220409"

# construct the query url using API operators
baseurl <- paste0("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=", term,
                  "&begin_date=", begin_date,
                  "&end_date=", end_date,
                  "&facet_filter=true&api-key=","mrb3Gx3ed9yP1KDVybtq9F2yEXWIgAga")

# examine our query url
baseurl
```

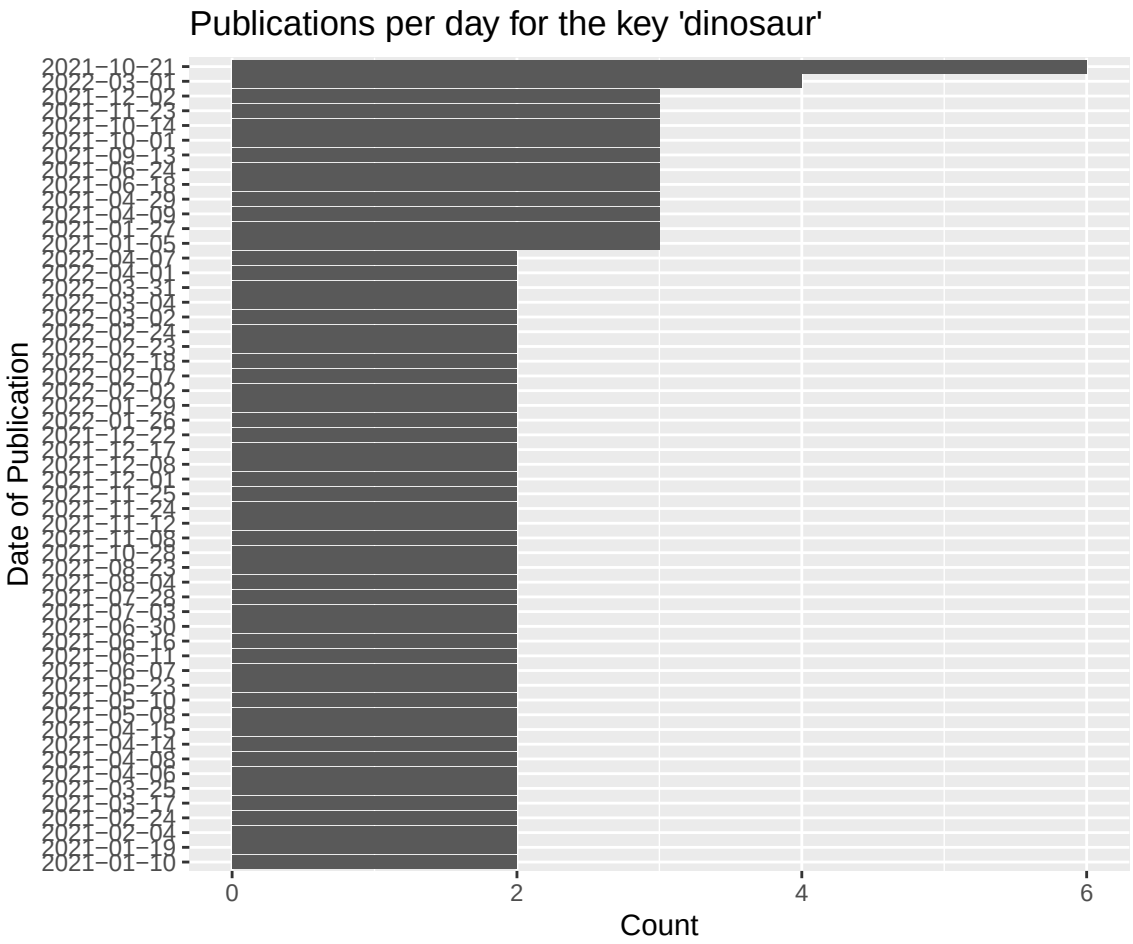
Now that I have my query URL, I can obtain the results. The `for loop` iterates over each of the 23 pages in `maxPages`. It turns each list into a dataframe, and then saves the results in an object `pages`.

Lastly, combine the list of data frames into a single data frame using `rbind_pages()`.

```
# this code allows for obtaining multiple pages of query results
initialQuery <- fromJSON(baseurl)
maxPages <- round((initialQuery$response$meta$hits[1] / 10) - 1) # 242 hits is 23 max pgs

# create an empty list
pages <- list()
for(i in 0:maxPages){
  nytSearch <- fromJSON(paste0(baseurl, "&page=", i), flatten = TRUE) %>% data.frame()
  message("Retrieving page ", i)
  pages[[i + 1]] <- nytSearch
  Sys.sleep(6)
}

class(nytSearch)

# need to bind the pages and create a tibble from nytDat
nytDat <- rbind_pages(pages)
dim(nytDat) # 240 x 33
```

Now that I have my larger query in the format I want, I can create some visuals. This is a visual of the publications per day for the key "dinosaur".

```
nytDat %>%
  mutate(pubDay = gsub("T.*" ,"", response.docs.pub_date)) %>%
  group_by(pubDay) %>%
  summarise(count = n()) %>%
  filter(count >= 2) %>%
  ggplot() +
  geom_bar(aes(x = reorder(pubDay, count),
               y = count),
           stat="identity") +
  coord_flip() +
  labs(
    title = "Publications per day for the key 'dinosaur'",
    x = "Date of Publication",
    y = "Count"
  )
```

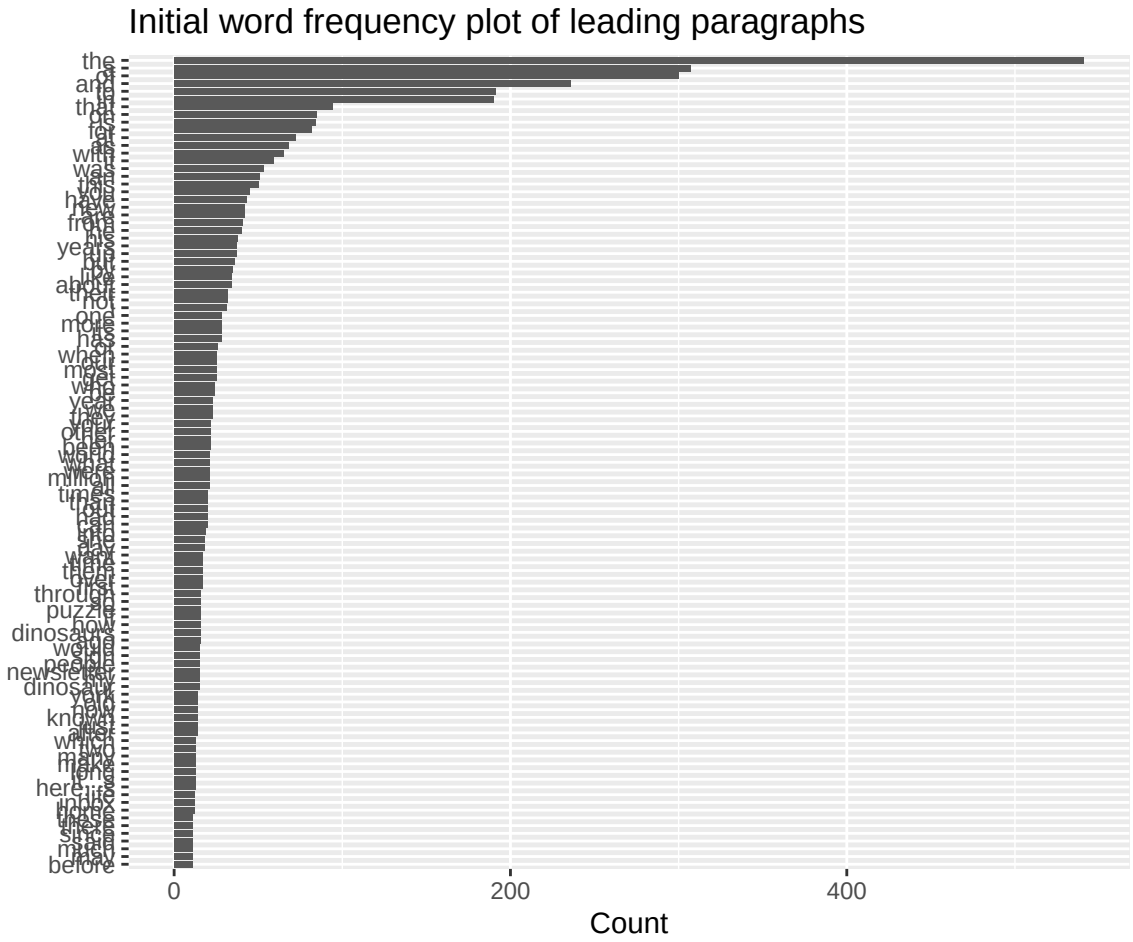


To create a word frequency plot using the first paragraph, I first need to tokenize the `response.doc.lead_paragraph`.

```
paragraph <- names(nytDat)[6] # response.doc.lead_paragraph
tokenized <- nytDat %>%
  unnest_tokens(word, paragraph)

tokenized[,34] # results of words variable
```

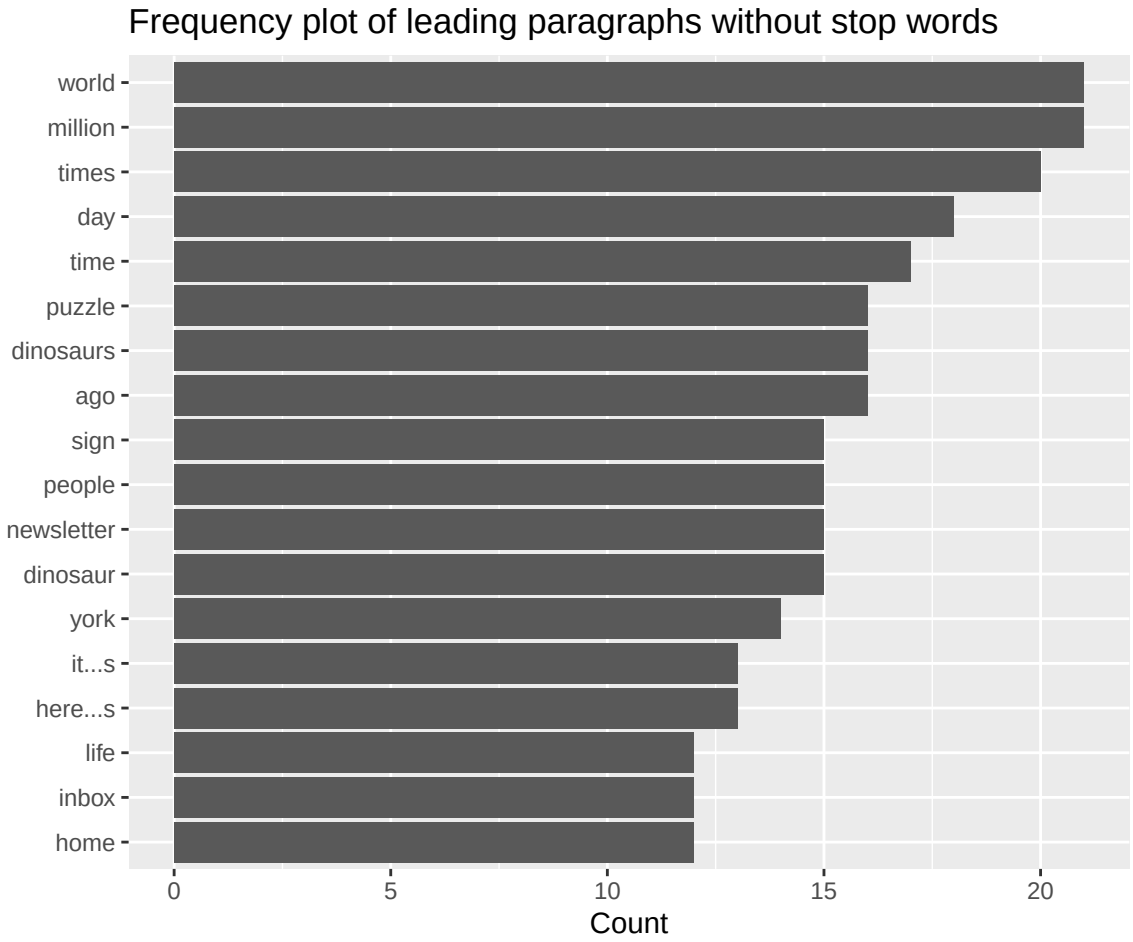
Here I create an initial plot of the tokenized paragraphs. This plot includes words with a frequency of at least 10. The big takeaway from the plot is that it's not informative because stop words are still included.

```
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 10) %>% # illegible with all the words displayed
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Initial word frequency plot of leading paragraphs",
       y = NULL,
       x = "Count",)
```



Here is the plot again, but without the stop words. However, I am also seeing the word "dinosaurs" and "it's" coming up, so I need to stem dinosaur (and time which I found out in a later plot which had time and times coming up), and remove possessive words.

```
tokenized <- tokenized %>%
  anti_join(stop_words)

tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 10) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Frequency plot of leading paragraphs without stop words",
       y = NULL,
       x = "Count")
```



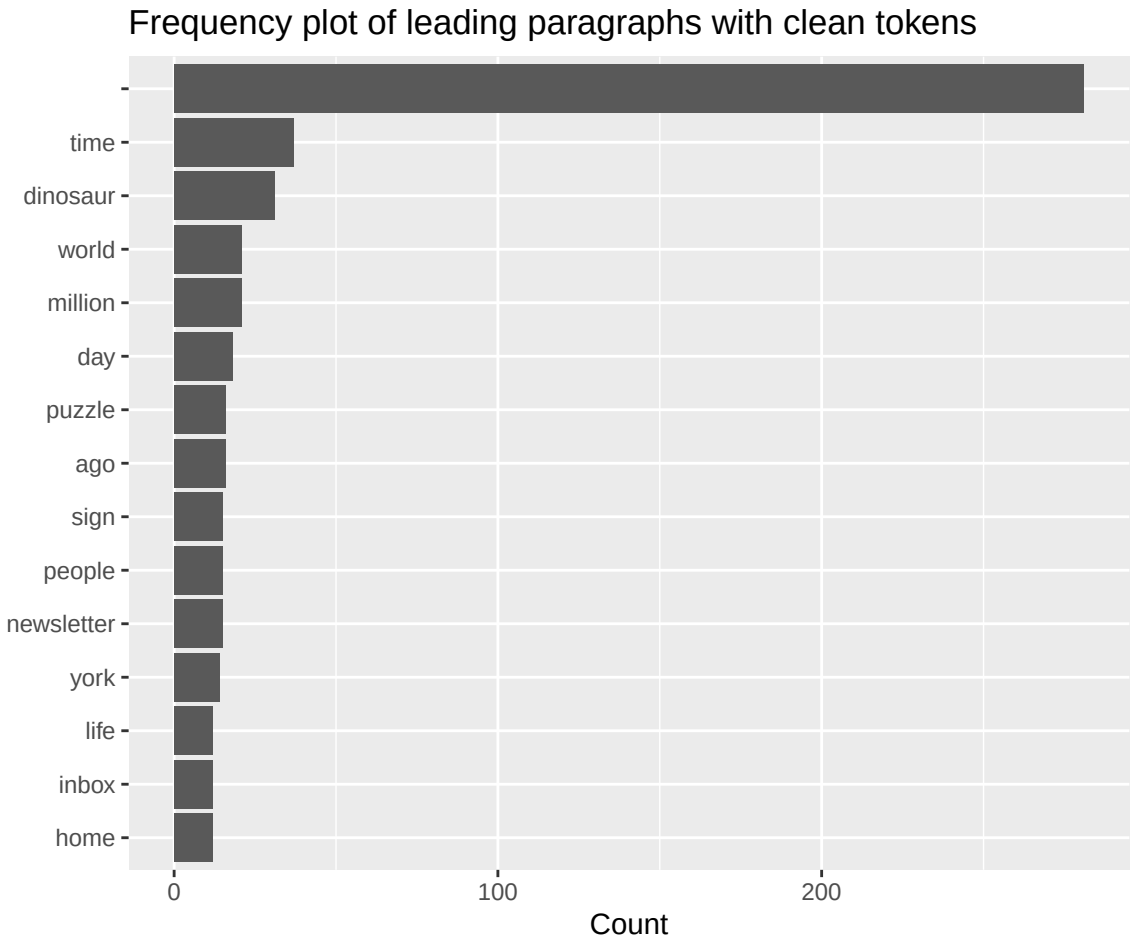
Here I am removing apostrophes, numbers, and stemmed the key term dinosaur.

```
# stem dinosaur words
clean_tokens <- str_replace_all(tokenized$word, "dinosaur[a-z, A-Z]*", "dinosaur")
# stem time words
clean_tokens <- str_replace_all(clean_tokens, "time[a-z, A-Z]*", "time")
# remove words with an apostrophe
clean_tokens <- str_remove_all(clean_tokens, "\\w+[:punct:]\\w+")
# remove all numbers
clean_tokens <- str_remove_all(clean_tokens, "[:digit:]")

# add clean tokens to df
tokenized_clean <- tokenized %>%
  mutate(clean = clean_tokens)
```

Here is my word frequency plot for the leading paragraphs after I created `clean_tokens`. The big takeaway here is that I need to remove all the empty strings.

```
tokenized_clean <- tokenized_clean %>%
  count(clean, sort = TRUE) %>%
  filter(n > 10) %>%
  mutate(clean = reorder(clean, n))
```

```
tokenized_clean_plot <- tokenized_clean %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(title = "Frequency plot of leading paragraphs with clean tokens",
       y = NULL,
       x = "Count")

tokenized_clean_plot
```

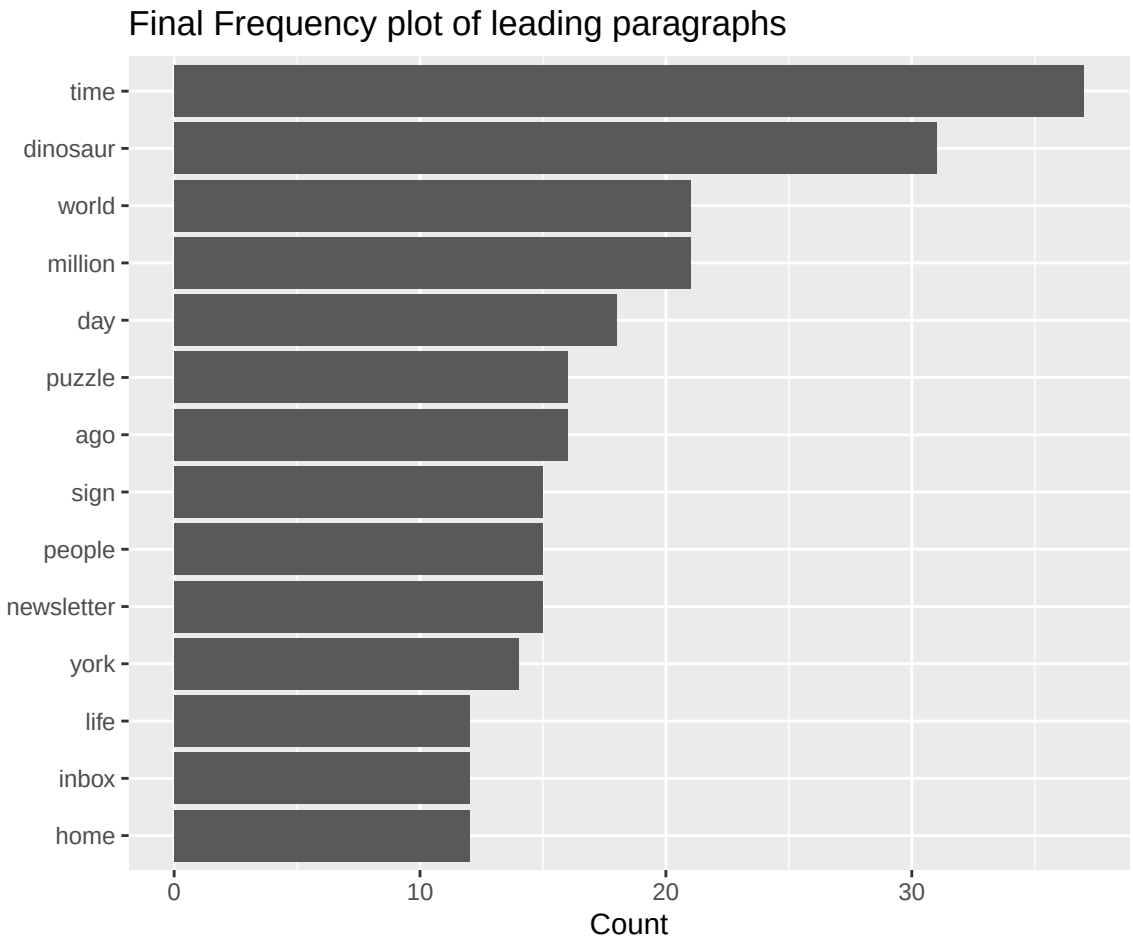


Here is my final word frequency plot for the leading paragraphs after I created `clean_tokens`. The big takeaway here is that I need to remove all the empty strings.

```
# remove empty strings
tib <- subset(tokenized_clean, clean != "")
tokenized_clean_tib <- tib

# create final df
tokenized_clean_final <- tokenized_clean_tib %>%
  mutate(clean = reorder(clean, n))
```

```
# plot
final_tokenized_plot <- tokenized_clean_final %>%
```

```
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(title = "Final Frequency plot of leading paragraphs",
       y = NULL,
       x = "Count")

final_tokenized_plot
```

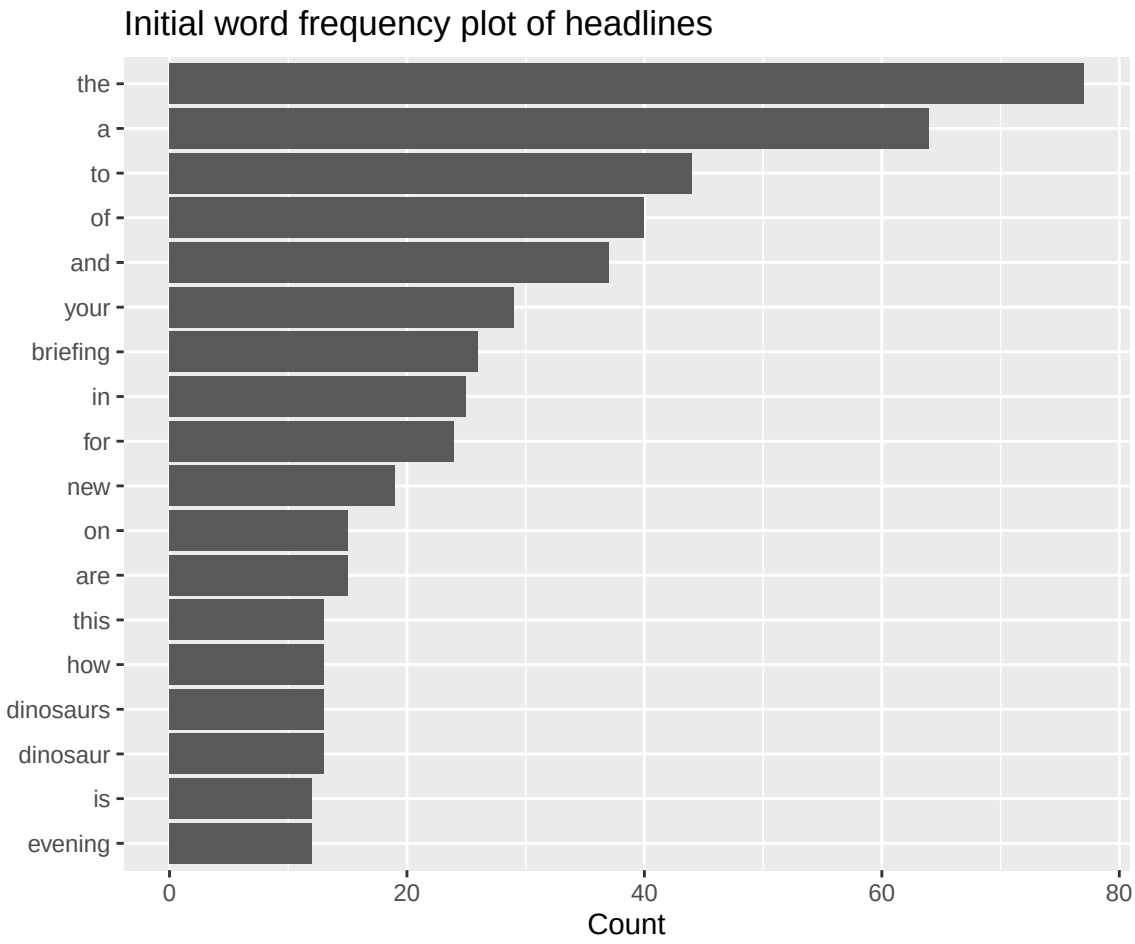


Here I am going to recreate the word frequency plots using the headline variable (`response.docs.headline.main`).

```
headlines <- names(nytDat)[21] # "response.docs.headline.main"

# tokenizing the headlines
tokenized_hl <- nytDat %>%
  unnest_tokens(output = word, # column to be created
                input = headlines) # column that is split

tokenized_hl[,34]
```

Here is my initial plot after I tokenized the headlines.

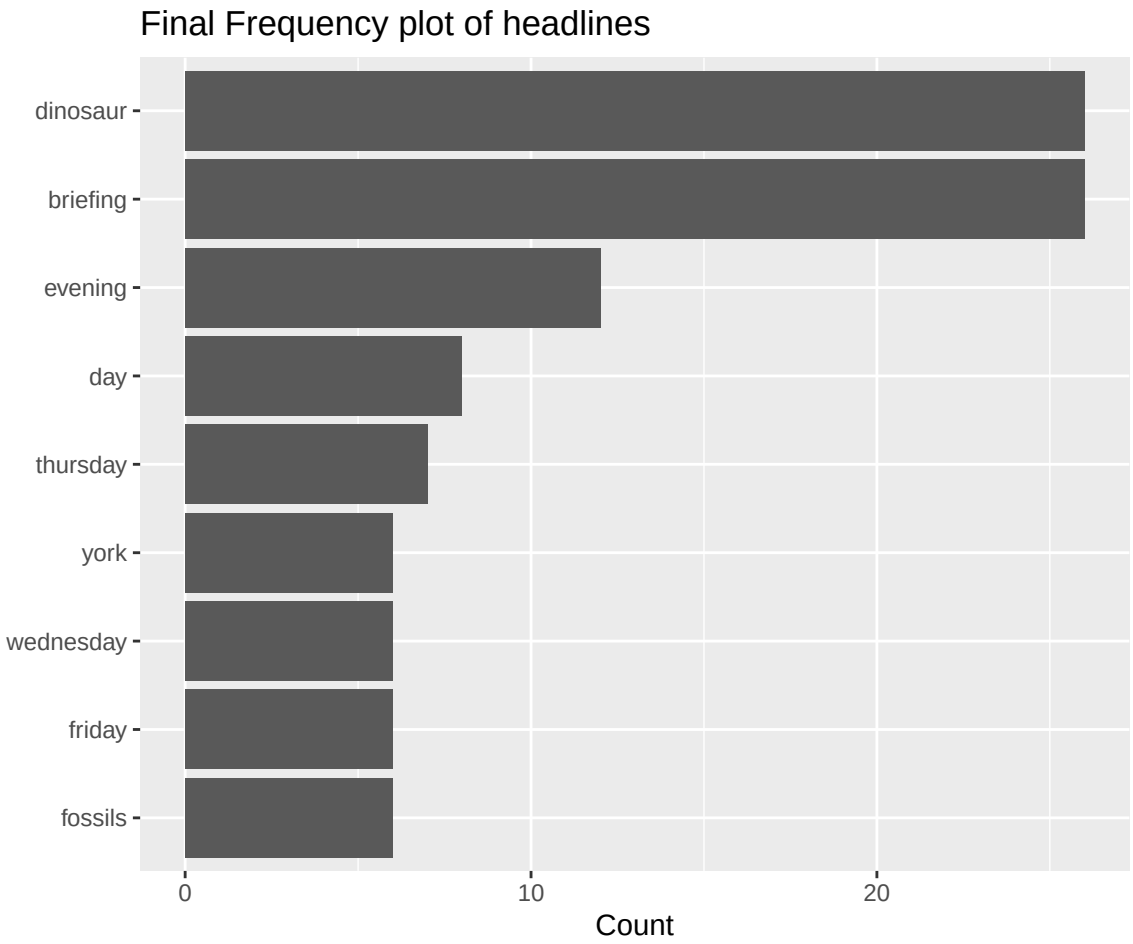
```
tokenized_hl %>%
  count(word, sort = TRUE) %>%
  filter(n > 10) %>% # illegible with all the words displayed
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Initial word frequency plot of headlines",
       y = NULL,
       x = "Count")
```



Here is my final plot after I have added the stop words and completed some of the necessary transformations to the corpus.

```
# added stop words
tokenized_hl <- tokenized_hl %>%
  anti_join(stop_words)

## CLEAN TOKENS ##
# stem dinosaur words
clean_tokens <- str_replace_all(tokenized_hl$word, "dinosaur[a-z, A-Z]*", "dinosaur")
# remove words with an apostrophe
clean_tokens <- str_remove_all(clean_tokens, "\\w+[:punct:]\\w+")
# remove all numbers
```

```r
clean_tokens <- str_remove_all(clean_tokens, "[:digit:]")

# add clean tokens to df
tokenized_hl_clean <- tokenized_hl %>%
  mutate(clean = clean_tokens)

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

# final clean df
tokenized_hl_clean_final <- tokenized_hl_clean %>%
  count(clean, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(clean = reorder(clean, n))

# final plot
final_tokenized_hl_plot <- tokenized_hl_clean_final %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(title = "Final Frequency plot of headlines",
       y = NULL,
       x = "Count")

final_tokenized_hl_plot
```



When I compare the final frequency plot of the leading paragraphs with the final frequency plot of the headlines, I do see quite a difference. For one, the top word is different between the two. After I stemmed "time", this word took the leading spot in the leading paragraphs plot. In fact, the only words in common between the two plots are: dinosaur, day, and york. And I'm not sure that "york" is a very relevant word here. I was also confused why so many days of the week were coming up in the headlines plot. When I took a closer look at the data, I found that many of the headlines were titled something like "Your Thursday Briefing". It appears that many dinosaur related articles are placed in these briefings.