# week_2_lab

Joe DeCesaro

4/10/2022

```
library(jsonlite) #convert results from API queries into R-friendly formats
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'tidyr' was built under R version 4.1.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
```

```
library(tidytext) #text data management and analysis
```

```
## Warning: package 'tidytext' was built under R version 4.1.1
```

```
library(ggplot2) #plot word frequencies and publication dates
library(patchwork) # stitch plots together
```

```
## Warning: package 'patchwork' was built under R version 4.1.1
```

## Query about renewable energy since 2021

```
api_key <- "kx8lL2TlBIFIgBShOs1eFATbWPgHHWX2"
term <- "renewable+energy" # Need to use + to string together separate words
begin_date <- "20200101" # YYYYMMDD
end_date <- "20220401" # close to present

#construct the query url using API operators
```

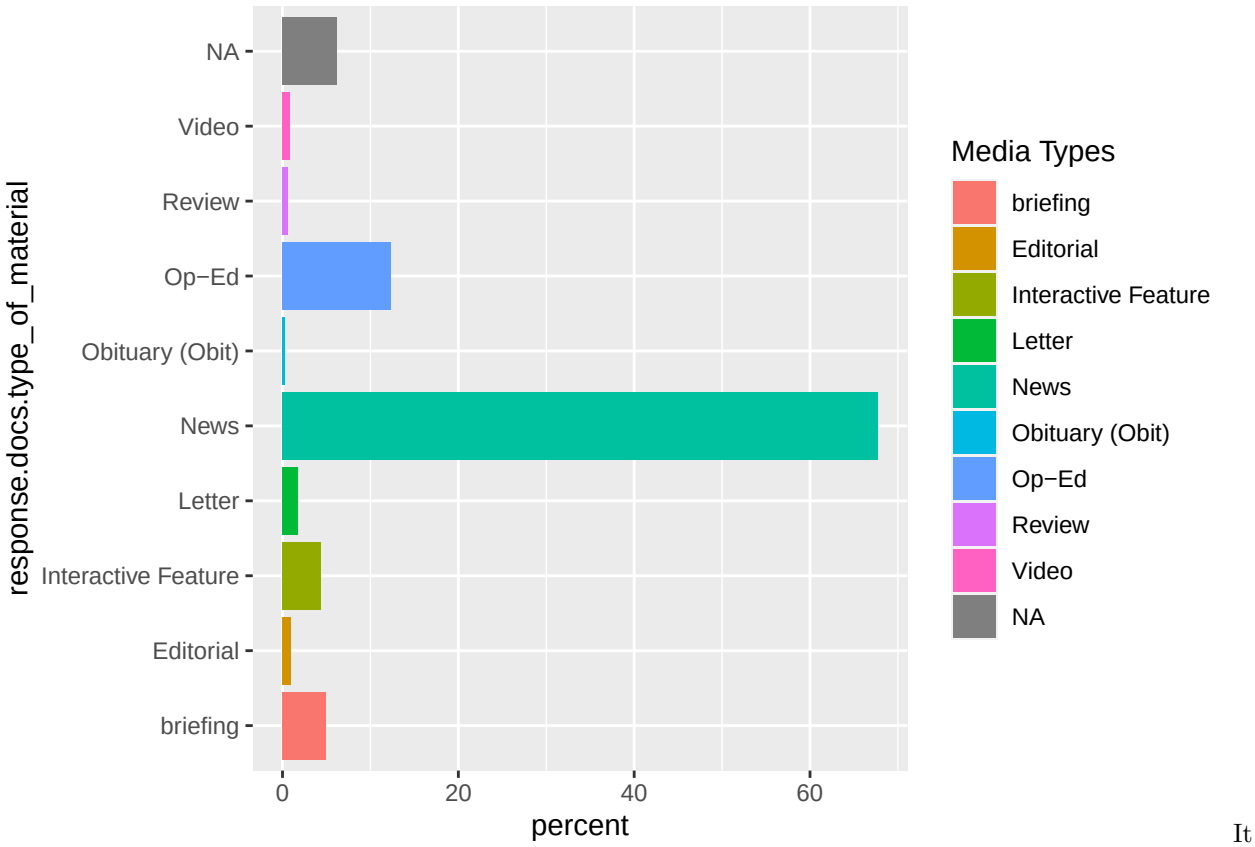
```
baseurl <- paste0("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=",term,
                  "&begin_date=",begin_date,
                  "&end_date=",end_date,
                  "&facet_filter=true&api-key=",api_key,
                  sep="")
```

After making our query we will use a for loop to retrieve the maximum number of information that we can.

```
#this code allows for obtaining multiple pages of query results
initialQuery <- fromJSON(baseurl)

maxPages <- round((initialQuery$response$meta$hits[1] / 10)-1)

# 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) # keeps you from hitting limit for AP
# }
# class(nytSearch)

#need to bind the pages and create a tibble from nytDa
# nytDat <- rbind_pages(pages)
# saveRDS(object = nytDat,
#         file = "data/nytDat.RData") # save the file so don't have to query every time
```

Next, lets make a quick plot using ggplot to see what media types have the words “renewable+energy”.

```
nytDat <- readRDS(file = "data/nytDat.RData") # read in saved file

nytDat %>%
  group_by(response.docs.type_of_material) %>%
  summarize(count=n()) %>%
  mutate(percent = (count / sum(count))*100) %>%
  ggplot() +
  geom_bar(aes(y=percent,
               x=response.docs.type_of_material,
               fill=response.docs.type_of_material),
           stat = "identity") +
  coord_flip() +
  labs(fill = "Media Types")
```




It looks like these are mostly news articles. Next lets see if the spread of publication dates is mostly even.

```
# 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()
```

## Let's analyze the leading paragraphs

The New York Times doesn't make full text of the articles available through the API. But we can use the first paragraph of each article.

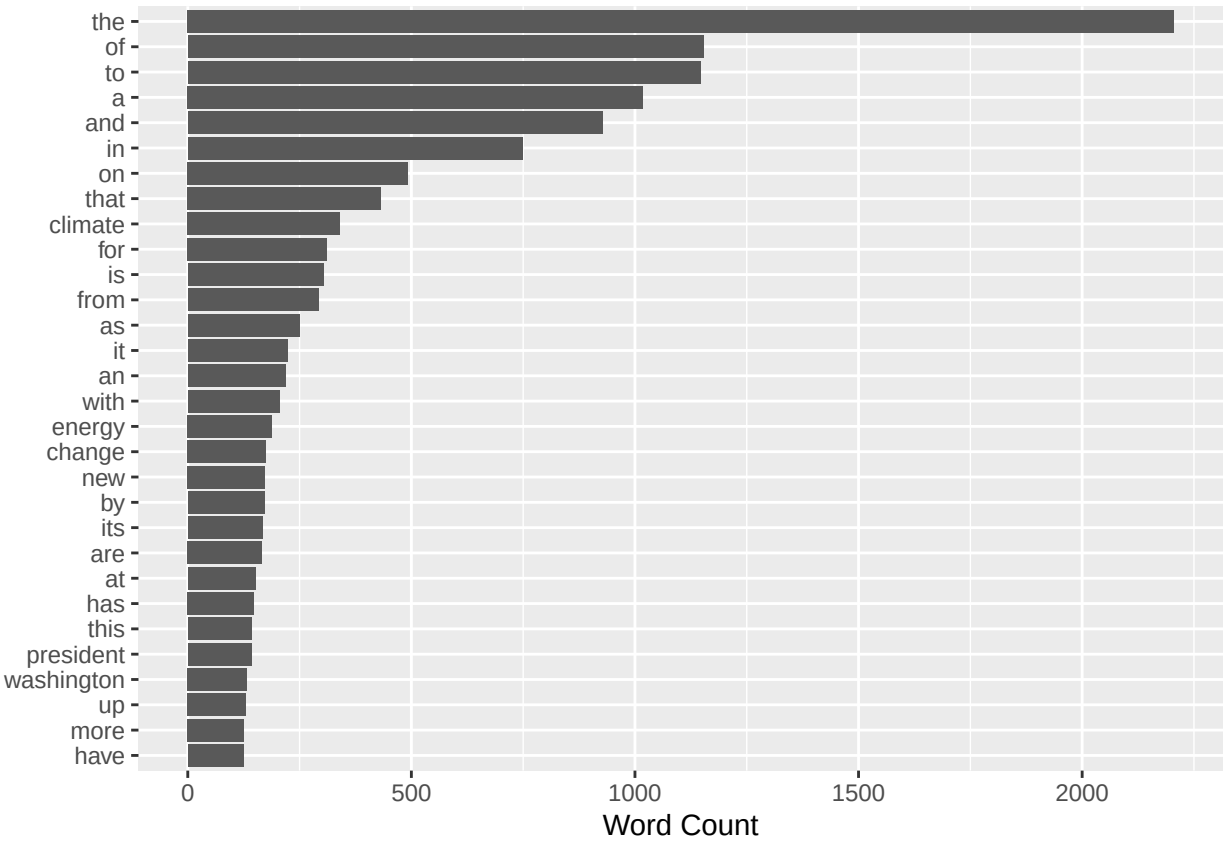
```
# look at variable names
names(nytDat)
```

```
##  [1] "status"
##  [2] "copyright"
##  [3] "response.docs.abstract"
##  [4] "response.docs.web_url"
```

```
##  [5] "response.docs.snippet"
##  [6] "response.docs.lead_paragraph"
##  [7] "response.docs.print_section"
##  [8] "response.docs.print_page"
##  [9] "response.docs.source"
## [10] "response.docs.multimedia"
## [11] "response.docs.keywords"
## [12] "response.docs.pub_date"
## [13] "response.docs.document_type"
## [14] "response.docs.news_desk"
## [15] "response.docs.section_name"
## [16] "response.docs.subsection_name"
## [17] "response.docs.type_of_material"
## [18] "response.docs._id"
## [19] "response.docs.word_count"
## [20] "response.docs.uri"
## [21] "response.docs.headline.main"
## [22] "response.docs.headline.kicker"
## [23] "response.docs.headline.content_kicker"
## [24] "response.docs.headline.print_headline"
## [25] "response.docs.headline.name"
## [26] "response.docs.headline.seo"
## [27] "response.docs.headline.sub"
## [28] "response.docs.byline.original"
## [29] "response.docs.byline.person"
## [30] "response.docs.byline.organization"
## [31] "response.meta.hits"
## [32] "response.meta.offset"
## [33] "response.meta.time"
```

For now, we want the 6th column, which is named "response.doc.lead_paragraph".

```
paragraph <- names(nytDat)[6] #The 6th column, "response.doc.lead_paragraph", is the one we want here.
tokenized <- nytDat %>%
  unnest_tokens(word, paragraph) #take paragraphs in and un-nest to word level (1 row for each word in

tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>% #illegible with all the words displayed
  mutate(word = reorder(word, n)) %>%
  slice(1:30) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL,
       x = "Word Count")
```

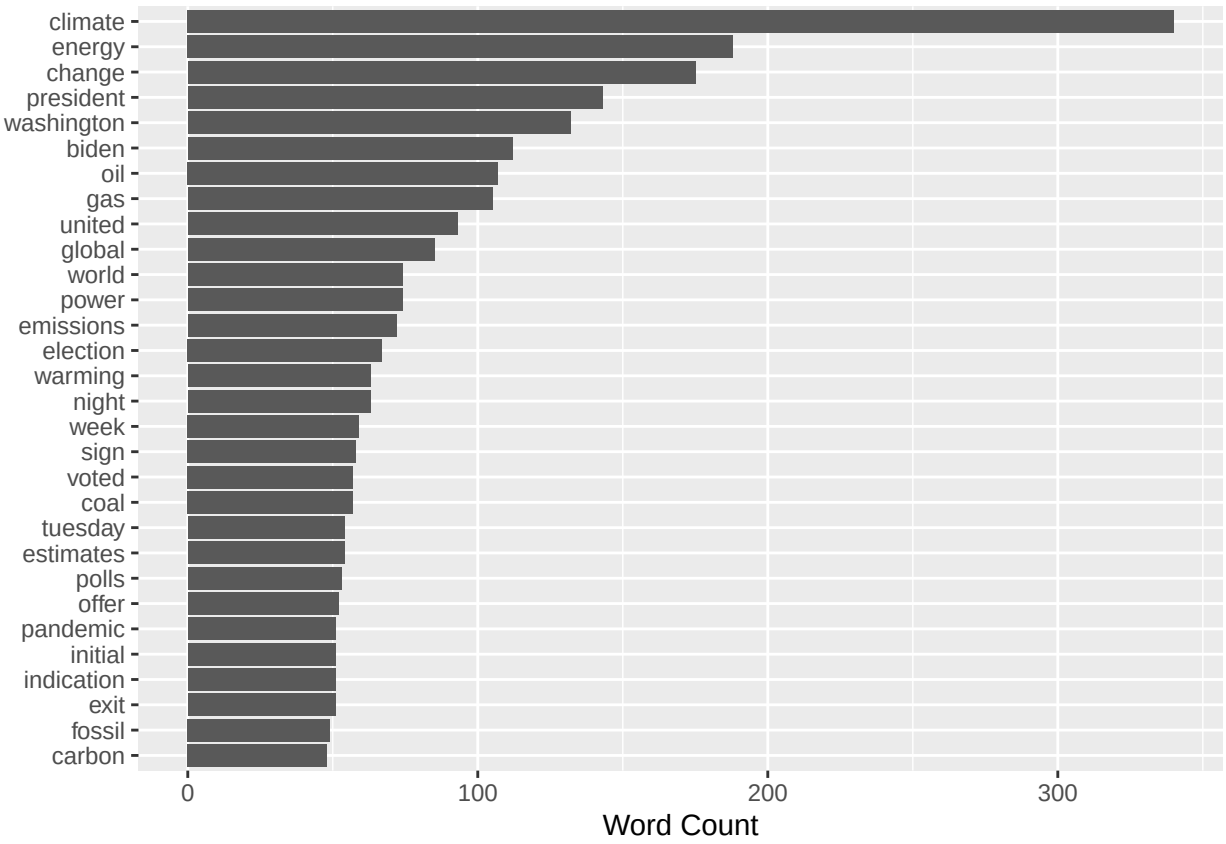



This is not very helpful as it contains a lot of stop words, lets clean it up.

```
# 'tidytext' has a list of common stopwords in English
data(stop_words) # pull stop_word data object from `tidytext` package

tokenized <- tokenized %>%
  anti_join(stop_words) # remove all rows that match a stopword
```

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

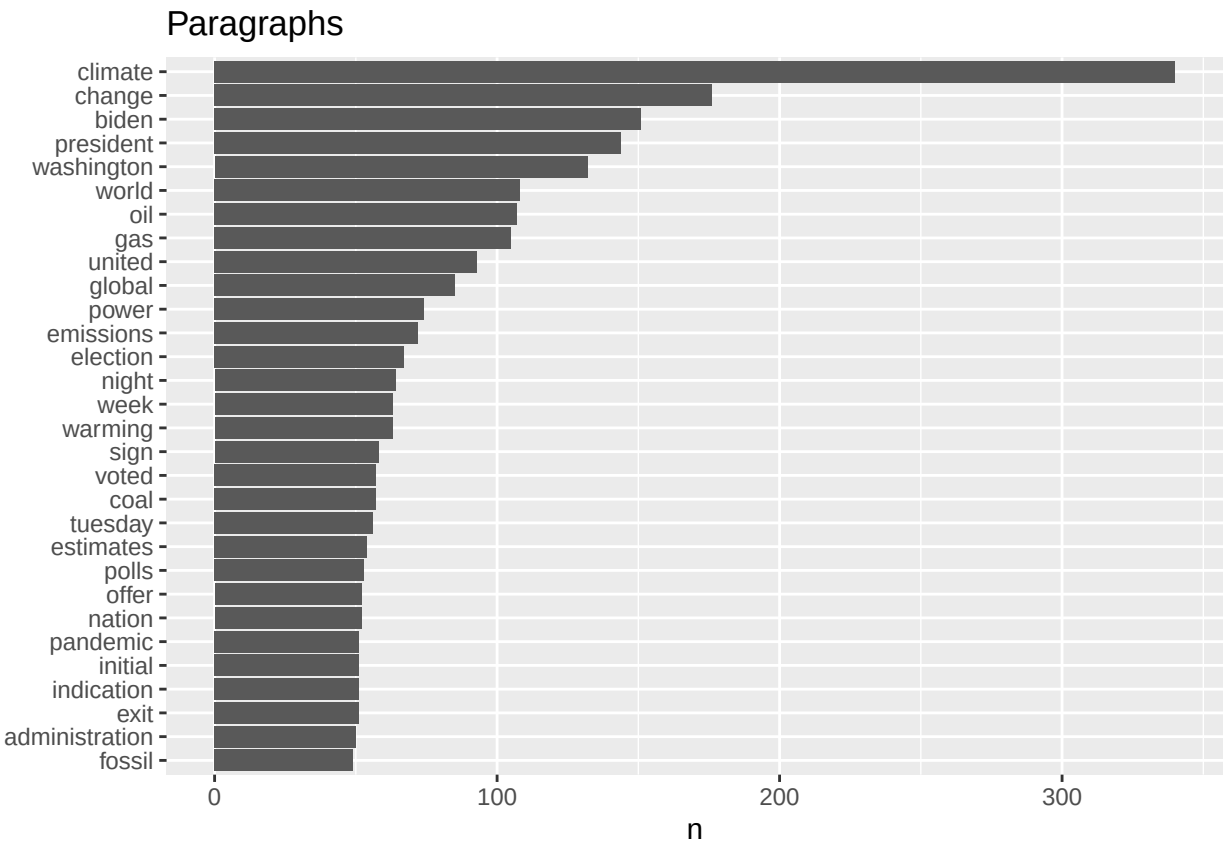
```
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n)) %>%
  slice(1:30) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL,
       x = "Word Count")
```




This is better but, lets work with it more to get more information out of the data. Lets start by making some words their stem, remove any numbers, and remove the word energy and renewable as we know that these articles discuss this topic.

```
#inspect the list of tokens (words)
# tokenized$word

clean_tokens <- str_replace_all(string = tokenized$word,
                                pattern = "energ[a-z,A-Z]*",
                                replacement = "energy") #stem tribe words

clean_tokens <- str_replace_all(string = tokenized$word,
                                pattern = "renewable[a-z,A-Z]*",
                                replacement = "renwable") #stem tribe words

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

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "energy") # remove energy because it will show up a lot

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "renewable") # remove renewable because it will show up a lot

clean_tokens <- gsub(pattern = "'s", # remove "'s" and replace them with nothing
                     replacement = '',
                     x = clean_tokens)
```

```
tokenized$clean <- clean_tokens # put the cleaned tokens into the `tokenized` df `clean` column

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

#reassign
tokenized <- tib

paragraph_plot <- tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 20) %>% #illegible with all the words displayed
  mutate(clean = reorder(clean, n)) %>%
  slice(1:30) %>% # get the top 30 words
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL,
       title = "Paragraphs")

paragraph_plot
```




It appears that renewable energy is most frequently discussed in articles that talk about climate change, US politics, and non-renewable energy. We could remove a few more words but I think this is fine as it is.

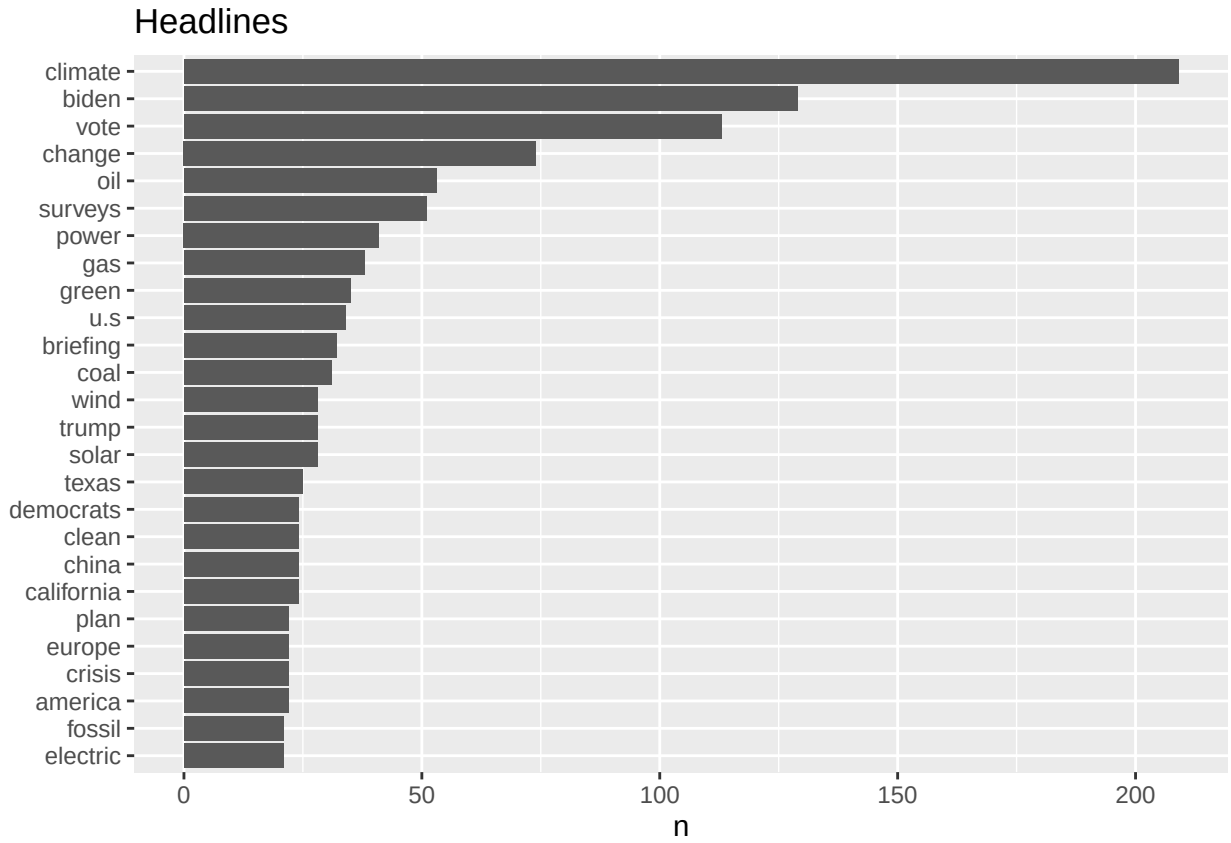
### Next let's look at the headlines and how they compare to the paragraphs

```
headline <- names(nytDat)[21] #The 21st column, "response.docs.headline.main", is the one we want here.
tokenized <- nytDat %>%
  unnest_tokens(word, headline) #take paragraphs in and un-nest to word level (1 row for each word in h

tokenized <- tokenized %>%
  anti_join(stop_words) # remove all rows that match a stopword
```

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

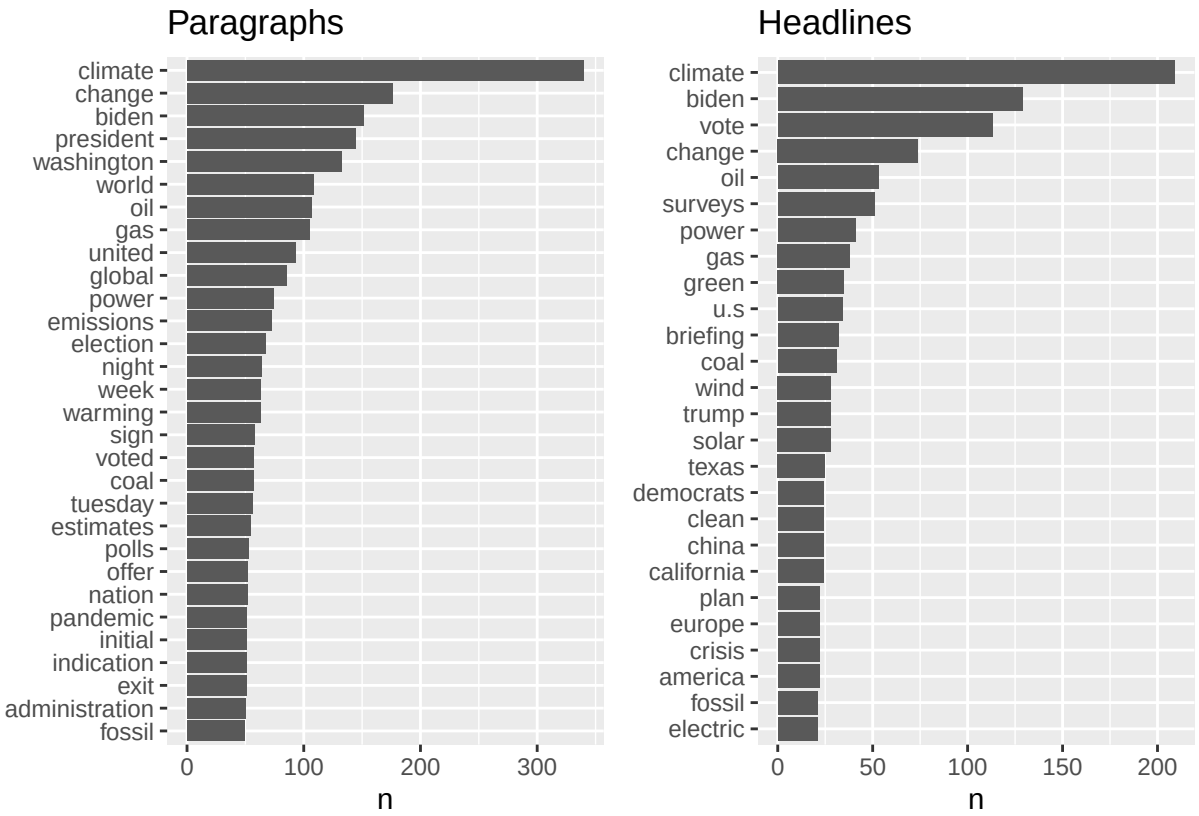
```
clean_tokens <- str_replace_all(string = tokenized$word,
                                pattern = "energ[a-z,A-Z]*",
                                replacement = "energy") #stem tribe words

clean_tokens <- str_replace_all(string = tokenized$word,
                                pattern = "renewable[a-z,A-Z]*",
                                replacement = "renwable") #stem tribe words

clean_tokens <- str_replace_all(string = tokenized$word,
                                pattern = "vot[a-z,A-Z]*",
                                replacement = "vote") #stem tribe words

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

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "energy") # remove energy because it will show up a lot

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "renewable") # remove renewable because it will show up a lot

clean_tokens <- gsub(pattern = "'s", # remove "'s" and replace them with nothing
                     replacement = '',
                     x = clean_tokens)

tokenized$clean <- clean_tokens # put the cleaned tokens into the `tokenized` df `clean` column

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

#reassign
tokenized <- tib

headline_plot <- tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 20) %>% #illegible with all the words displayed
  mutate(clean = reorder(clean, n)) %>%
  slice(1:30) %>% # get the top 30 words
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL,
```

```
       title = "Headlines")

headline_plot
```




It seems like there are a lot of shared words in the headlines and the paragraphs but lets look at them side by side.

```
paragraph_plot + headline_plot
```




Overall there are a lot of shared words between the paragraphs and headlines including climate, change, Biden and more. There is a noticeable increase in words like vote, surveys, crisis, and other.