Text Data in R

Alex Clippinger

04/12/2022

The purpose of this R Markdown document is to query text data from the New York Times database via their API. Then, basic string manipulations, text data wrangling, and basic visualizations will be created to analyze the data.

Setup

```
knitr::opts_chunk$set(echo = TRUE)

library(jsonlite) #convert results from API queries into R-friendly formats
library(tidyverse)
library(readr)
library(lubridate)
library(tidytext) #text data management and analysis
library(ggplot2) #plot word frequencies and publication dates
```

Data

For this exercise, I chose to use the search term “methane” to select articles related to methane emissions, policy, public opinion, etc.

Create API Query

```
term <- "methane" # Use + to string together separate words
begin_date <- "20210120"
end_date <- Sys.Date()
api_key <- read_file("api_key.txt")

#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="")
```

Retrieve Data via API

The following code chunk retrieves the data through the API. The data is stored at the end in order to prevent the need to run this code multiple times.

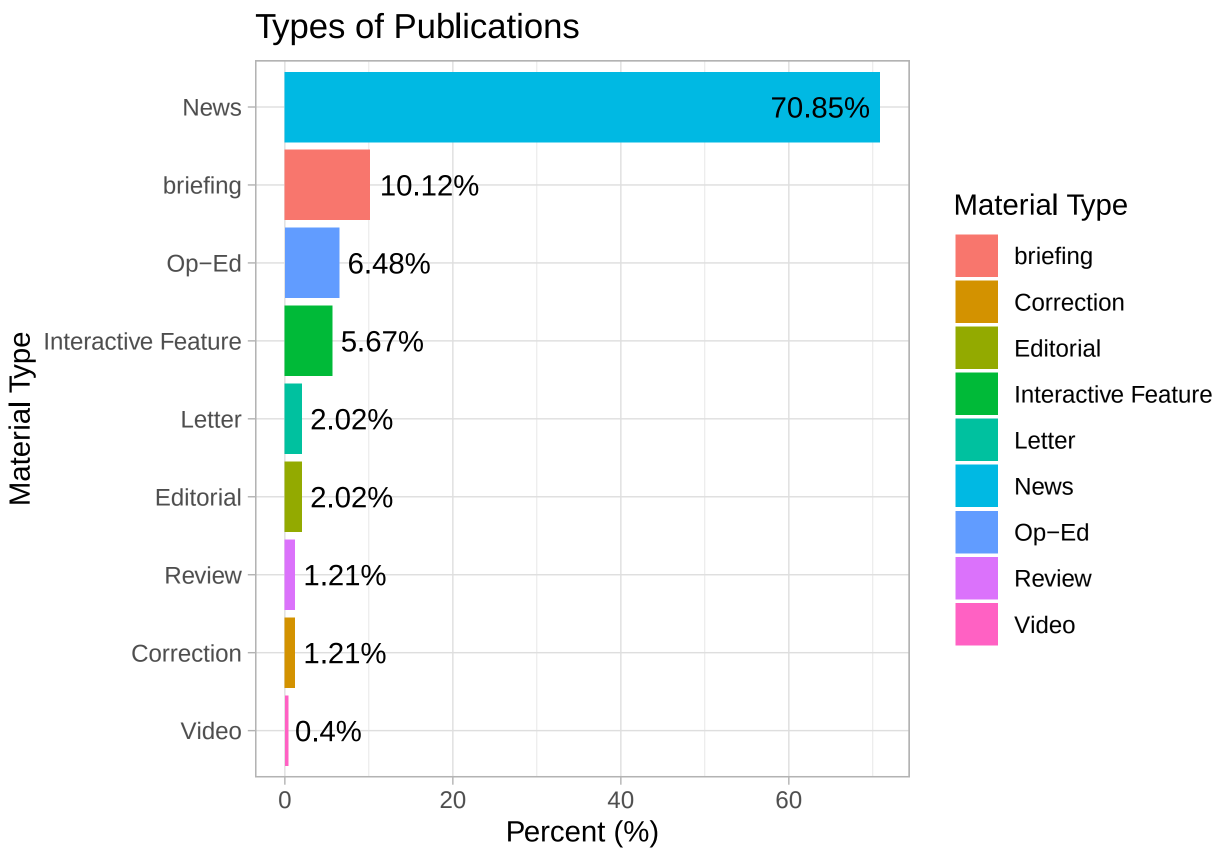
```r
#this code allows for obtaining multiple pages of query results
initialQuery <- fromJSON(baseurl)

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)
}

# bind the pages and create a tibble
nytDat <- rbind_pages(pages)

# save to file for faster knitting
saveRDS(nytDat, "nytDat.rds")
```

```r
nytDat <- readRDS("nytDat.rds")
```

Data Summary

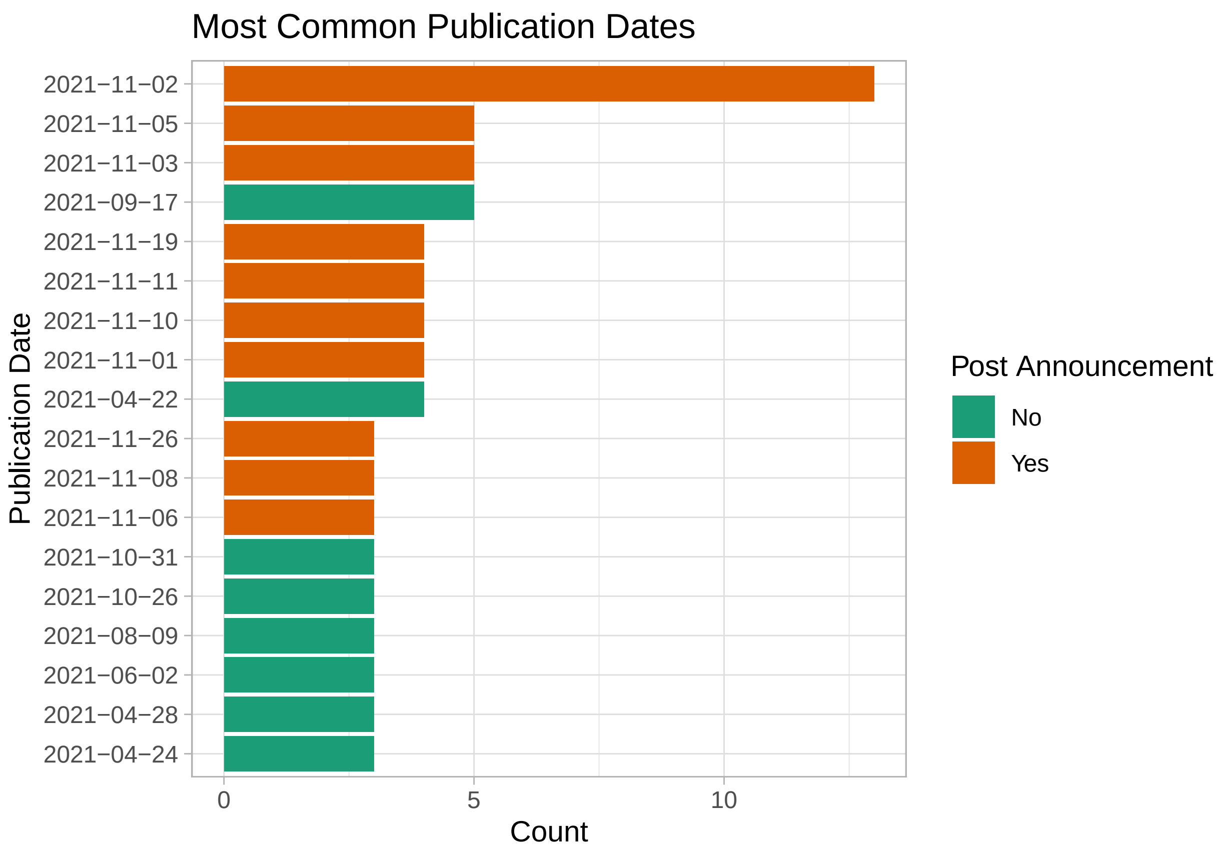
Overview

```r
nytDat_articles <- nytDat %>%
  rename(`Material Type` = response.docs.type_of_material) %>%
  group_by(`Material Type`) %>%
  summarize(count=n()) %>%
  mutate(percent = (count / sum(count))*100)

ggplot(data = nytDat_articles,
       aes(y=percent, x=reorder(`Material Type`, percent), fill=`Material Type`)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_light() +
  labs(y="Percent (%)", x="Material Type", title="Types of Publications") +
  geom_text(aes(label = ifelse(percent<25, paste0(round(percent, 2),"%"), ""), hjust = -.1)) +
  geom_text(aes(label = ifelse(percent>=25, paste0(round(percent, 2),"%"), ""), hjust = 1.1))
```



Most of the pages retrieved were news articles, which made up over 70% of the results.

```
nytDat_dates <- nytDat %>%
  mutate(pubDay=gsub("T.*","",response.docs.pub_date)) %>%
  group_by(pubDay) %>%
  summarise(count=n()) %>%
  filter(count >= 3) %>%
  mutate(`Post Announcement`=factor(ifelse(year(pubDay)==2021 & month(pubDay)==11, "Yes", "No")))

ggplot(nytDat_dates) +
  geom_bar(aes(x=reorder(pubDay, count), y=count, fill=`Post Announcement`),
           stat="identity") +
  scale_fill_brewer(palette="Dark2") +
  coord_flip() +
  theme_light() +
  labs(x="Publication Date", y="Count", title="Most Common Publication Dates")
```

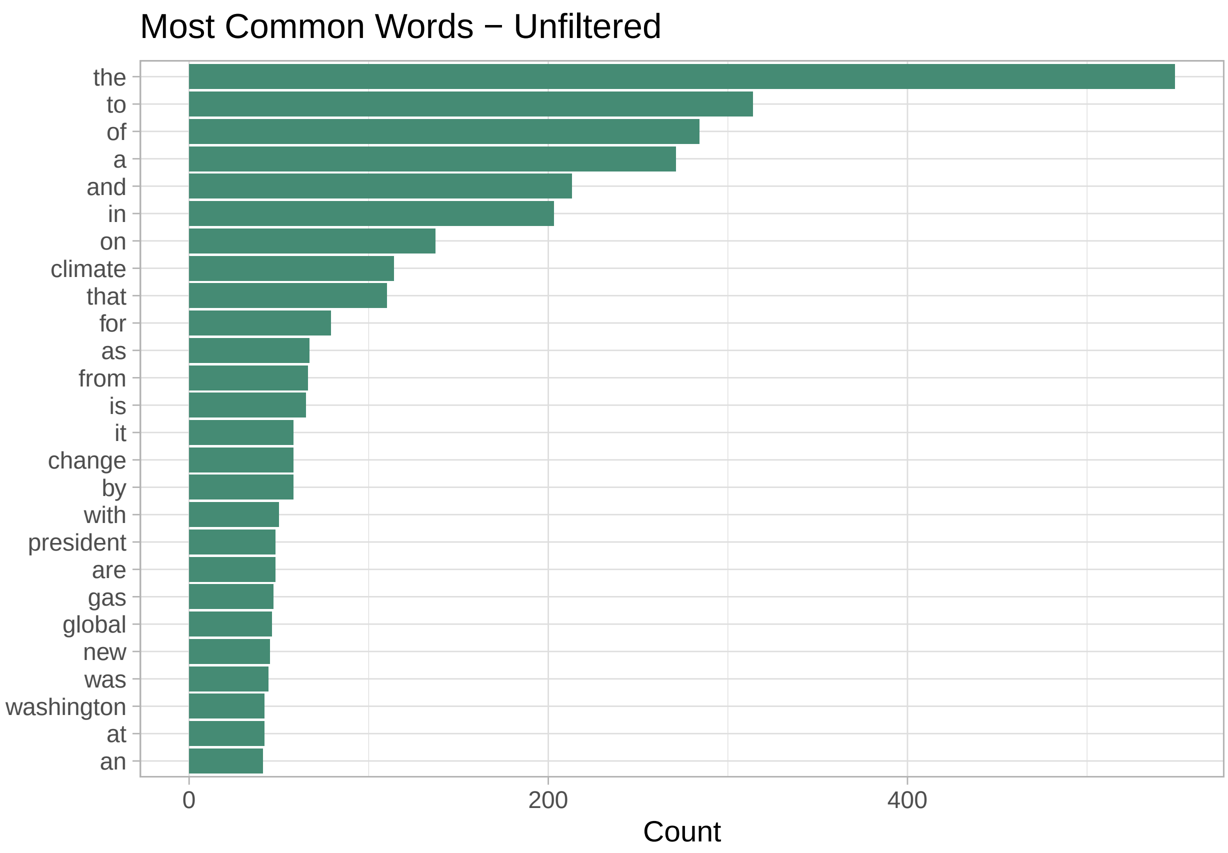


The plot above is dominated by publications in November of 2021. On November 2nd, the President announced an action plan targeting reductions in methane emissions in the Oil and Gas sector. His speech at the COP26 climate summit in Glasgow, Scotland can be viewed here.

First Paragraph Data

Since the New York Times doesn't provide access to full articles through the API, below we will use only the first paragraph of each article to summarize the most common words used.

```
paragraph <- names(nytDat)[6] #The 6th column, "response.doc.lead_paragraph"

tokenized <- nytDat %>%
  unnest_tokens(word, paragraph)

tokenized_summary <- tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 40) %>% #illegible with all the words displayed
  mutate(word = reorder(word, n))

ggplot(data = tokenized_summary, aes(n, word)) +
  geom_col(fill="aquamarine4") +
  labs(y = NULL, x = "Count", title="Most Common Words - Unfiltered") +
  theme_light()
```

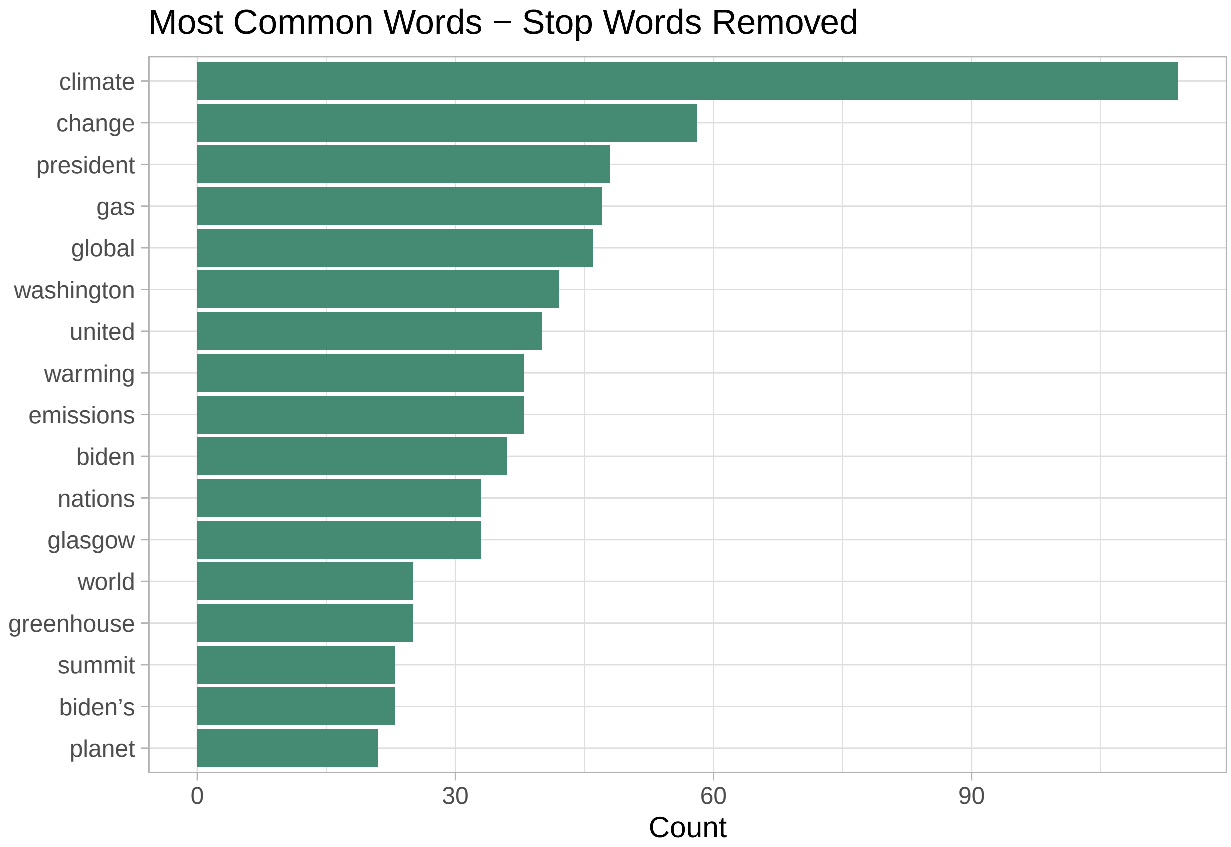


Clearly, words such as “the”, “to”, “of”, and “a” make up the top of the list, but are largely uninformative. Below, we remove stop words from the onix, SMART, and snowball lexicons.

```
data(stop_words)

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

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

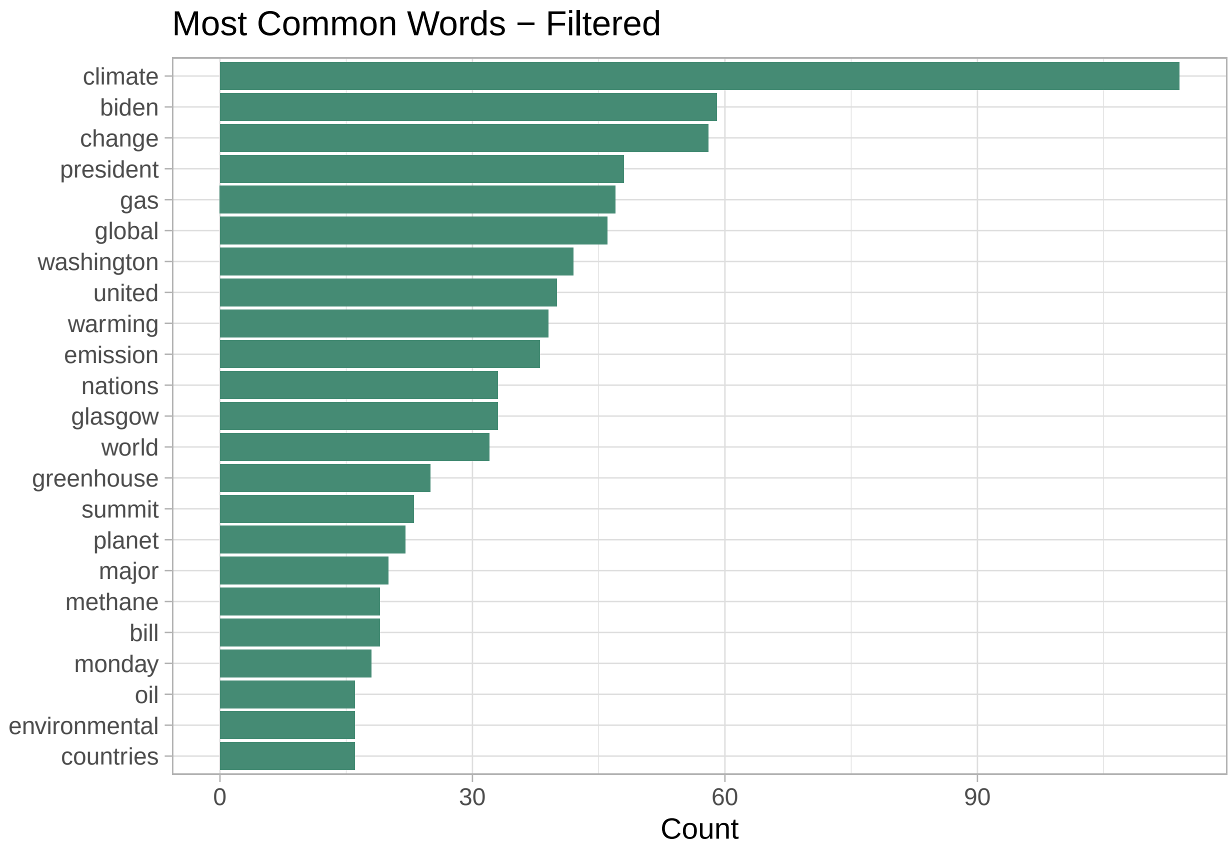
```
tokenized_filtered <- tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 20) %>%
  mutate(word = reorder(word, n))

ggplot(data = tokenized_filtered, aes(n, word)) +
  geom_col(fill="aquamarine4") +
  labs(y = NULL, x = "Count", title="Most Common Words - Stop Words Removed") +
  theme_light()
```



The top words are now more informative. For instance, the top word, “climate”, is the primary topic of discussion when dealing with methane, which is a greenhouse gas (“greenhouse” and “gas” also appear in the list). The second word, “change”, is used in this context to describe “climate change”.

Still, we can clean the text data further to remove stop words such as “it” that were not captured with the default lexicons. Additionally, we can combine cases such as “Biden” and “Biden’s”.

```
clean_tokens <- str_replace_all(tokenized$word,"emission[a-z,A-Z]*","emission") #stem emissions
clean_tokens <- str_remove_all(clean_tokens, "[:digit:]") #remove all numbers
clean_tokens <- str_remove_all(clean_tokens, "[*]day") # remove days of the week
clean_tokens <- gsub("’s", '', clean_tokens) # remove possessive

tokenized$clean <- clean_tokens

#remove empty strings
tib <-subset(tokenized, !(clean %in% c("", ".")))

#reassign
tokenized <- tib

tokenized_cleaned_v2 <- tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 15) %>%
  mutate(clean = reorder(clean, n))

ggplot(data = tokenized_cleaned_v2, aes(n, clean)) +
  geom_col(fill="aquamarine4") +
  labs(y = NULL, x = "Count", title="Most Common Words - Filtered") +
  theme_light()
```



After cleaning the data, "biden" sees an increase in count, largely due to the aggregation of the possessive "biden's". This is intuitive, as indicated earlier that the majority of the articles written during this time period were following the President's methane emissions action plan announcement.

Headline Data

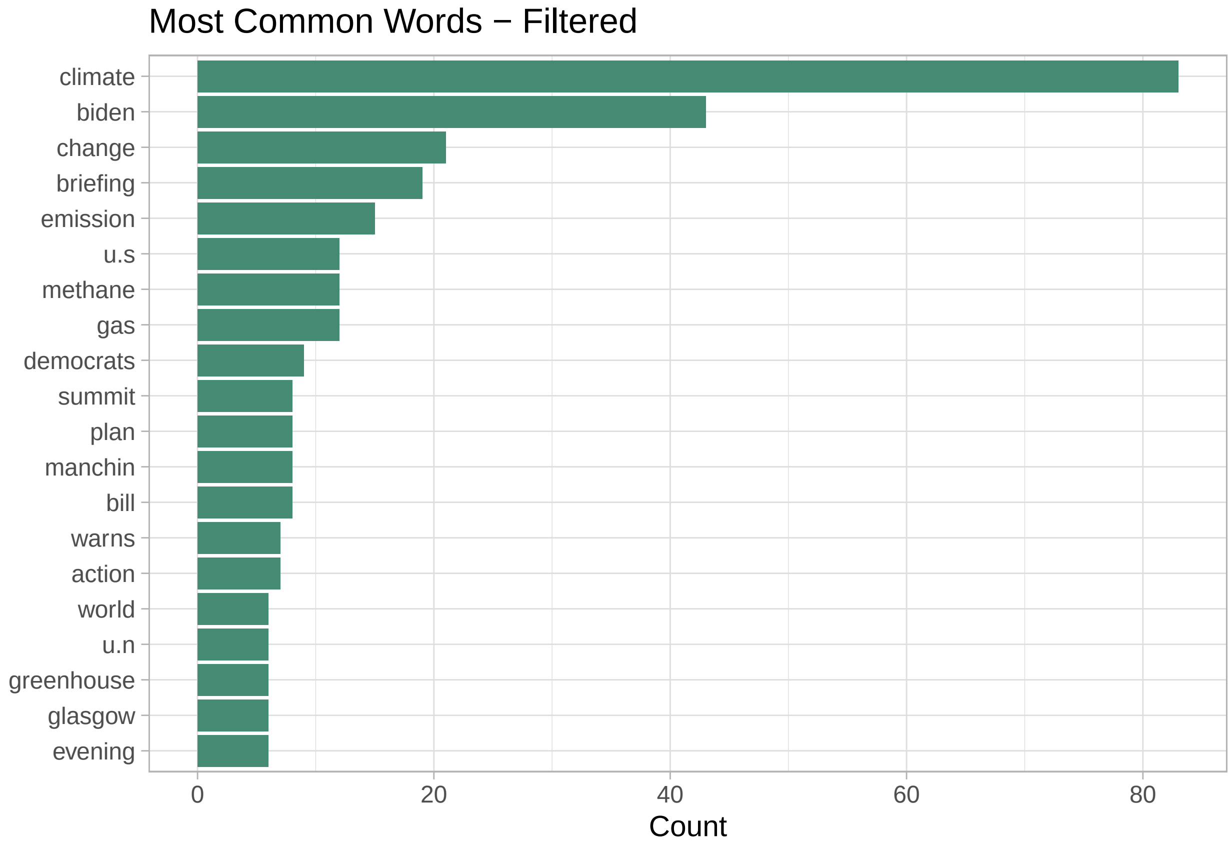
Let's do the same exercise using headlines instead of first paragraph's.

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

tokenized <- nytDat %>%
  unnest_tokens(word, paragraph)

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

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

```
tokenized_filtered <- tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n))
```

```r
clean_tokens <- str_replace_all(tokenized$word,"emission[a-z,A-Z]*","emission") #stem emissions
clean_tokens <- str_remove_all(clean_tokens, "[:digit:]") #remove all numbers
clean_tokens <- str_remove_all(clean_tokens, "[*]day") # remove days of the week
clean_tokens <- gsub("’s", '', clean_tokens) # remove possessive

tokenized$clean <- clean_tokens

#remove empty strings
tib <-subset(tokenized, !(clean %in% c("", ".", "it")))

#reassign
tokenized <- tib

tokenized_cleaned_v2 <- tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(clean = reorder(clean, n))

ggplot(data = tokenized_cleaned_v2, aes(n, clean)) +
  geom_col(fill="aquamarine4") +
  labs(y = NULL, x = "Count", title="Most Common Words - Filtered") +
  theme_light()
```



We see that the top 3, “climate”, “Biden”, and “change”, are the same as the analysis covering the paragraph data. However, there are noticeable differences in the list. Words such as “briefing”, “methane”, and “emission” have made a huge jump. This makes sense because many publications likely cover President

Biden's press briefing, in which he outlined new rules and regulations for methane emissions. An additional interesting adition is Manchin referring to Senator Manchin, a known supporter of the oil, gas, and coal industries.

Conclusion

In summary, this exercise demonstrated how to access New York Times article data from their API and quickly analyze results using basic text analysis functions and visualizations. For the search term used above, "methane", many of the published articles in the past year were found to revolve around President Biden's November 2021 global pledge to reduce methane gas.