EDS 231 Week 2 Lab

Wylie Hampson

4/11/2022

This code chunk uses the NYT API to search for articles. I decided to use the term "Electric Vehicles" and look for all articles between 01/01/2012 and 04/01/2022.

```r
term <- "Electric+Vehicles" # Need to use + to string together separate words
begin_date <- "20120101"
end_date <- "20220401"

key <- "rCsHXgvrixcSaNg6ardzy3HgWFAQcAYu"

#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=", key, sep="")

#examine our query url
baseurl
```

```
## [1] "http://api.nytimes.com/svc/search/v2/articlesearch.json?q=Electric+Vehicles&begin_date=20120101
```

This code retrieves multiple pages of articles from my search query. I decided to limit the search to 20 pages because the query returned a huge amount of results.

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

pages <- list()
for(i in 0:20){
  nytSearch <- fromJSON(paste0(baseurl, "&page=", i), flatten = TRUE) %>% data.frame()
  message("Retrieving page ", i)
  pages[[i+1]] <- nytSearch
  Sys.sleep(6)
}
```

```
## Retrieving page 0

## Retrieving page 1

## Retrieving page 2

## Retrieving page 3
## Retrieving page 4

## Retrieving page 5

## Retrieving page 6

## Retrieving page 7

## Retrieving page 8

## Retrieving page 9

## Retrieving page 10

## Retrieving page 11

## Retrieving page 12

## Retrieving page 13

## Retrieving page 14

## Retrieving page 15

## Retrieving page 16

## Retrieving page 17

## Retrieving page 18

## Retrieving page 19

## Retrieving page 20
```

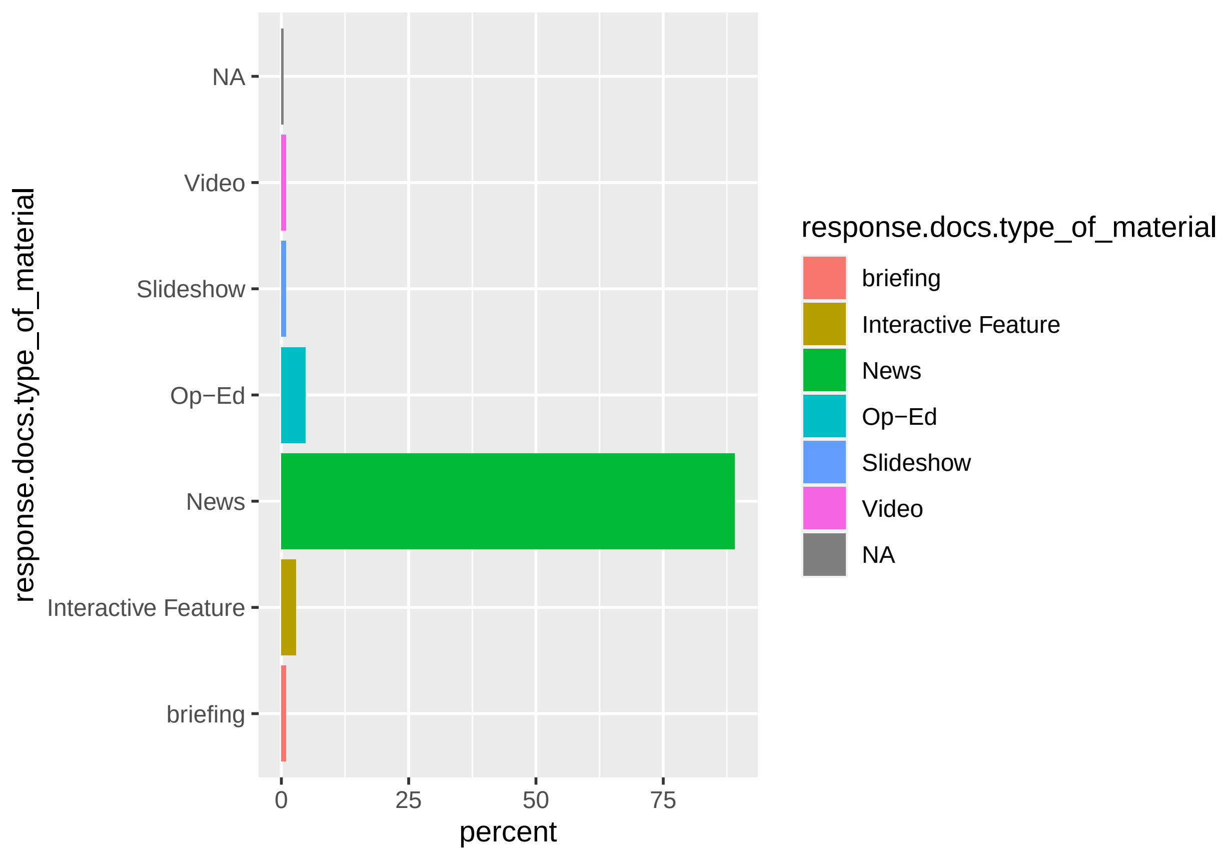
```
class(nytSearch)
```

```
## [1] "data.frame"
```

```
#need to bind the pages and create a tibble from nytDa
nytDat <- rbind_pages(pages)
```

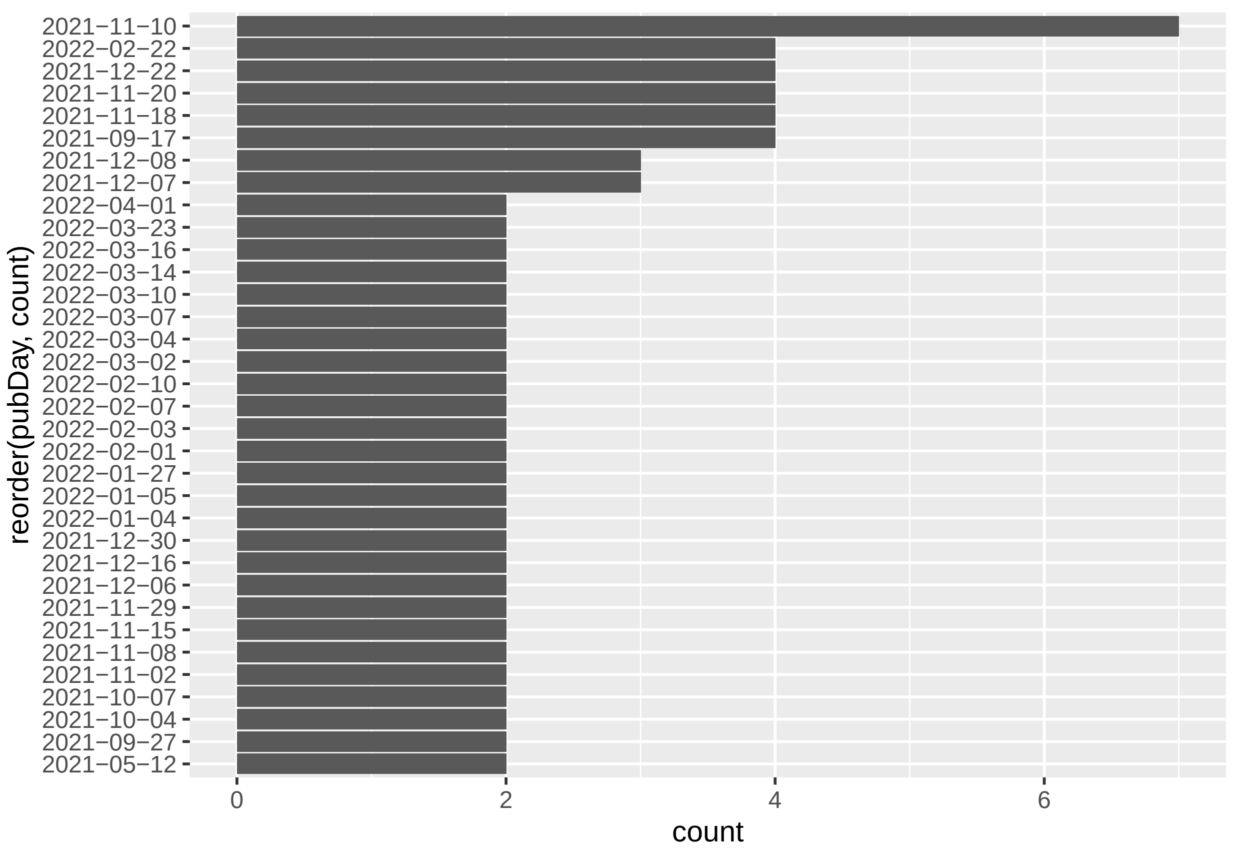
This code chunk creates a plot that shows the percentage of what types of media the search term comes up in. The most common one being News media.

```
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
```



This code chunk plots the number of articles published per date, so we can see which days published the most relevant articles on the search. We see 11/10/2021 had 7 articles about electric vehicles published that day.

```
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()
```



```
names(nytDat)
```

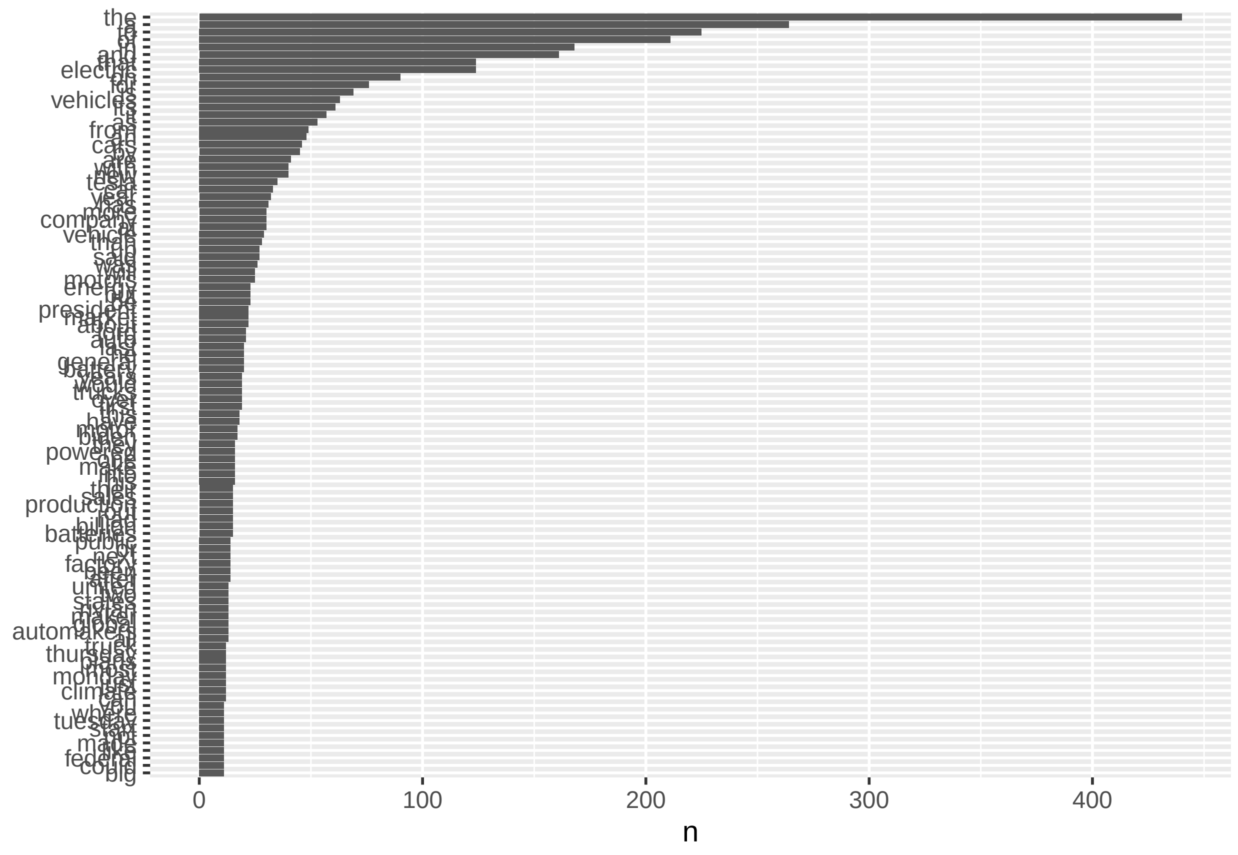
```
##  [1] "status"
##  [2] "copyright"
##  [3] "response.docs.abstract"
##  [4] "response.docs.web_url"
##  [5] "response.docs.snippet"
##  [6] "response.docs.lead_paragraph"
##  [7] "response.docs.print_section"
##  [8] "response.docs.print_page"
##  [9] "response.docs.source"
## [10] "response.docs.multimedia"
## [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.type_of_material"
## [17] "response.docs._id"
## [18] "response.docs.word_count"
## [19] "response.docs.uri"
## [20] "response.docs.subsection_name"
## [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"
## [34] "response.docs.slideshow_credits"
```

This code chunk creates a column to put each word from the lead paragraphs of the articles into its own row.

```
paragraph <- names(nytDat)[6] #The 6th column, "response.doc.lead_paragraph", is the one we want here.
tokenized <- nytDat %>%
  unnest_tokens(word, paragraph)

#tokenized[,35]
```

This code chunk takes all of the words from the word column and creates a histogram. However, notice that stop words are still included, and there are too many words on it to read.

```
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(y = NULL)
```



Here we pull in the stop words so that we can remove them.

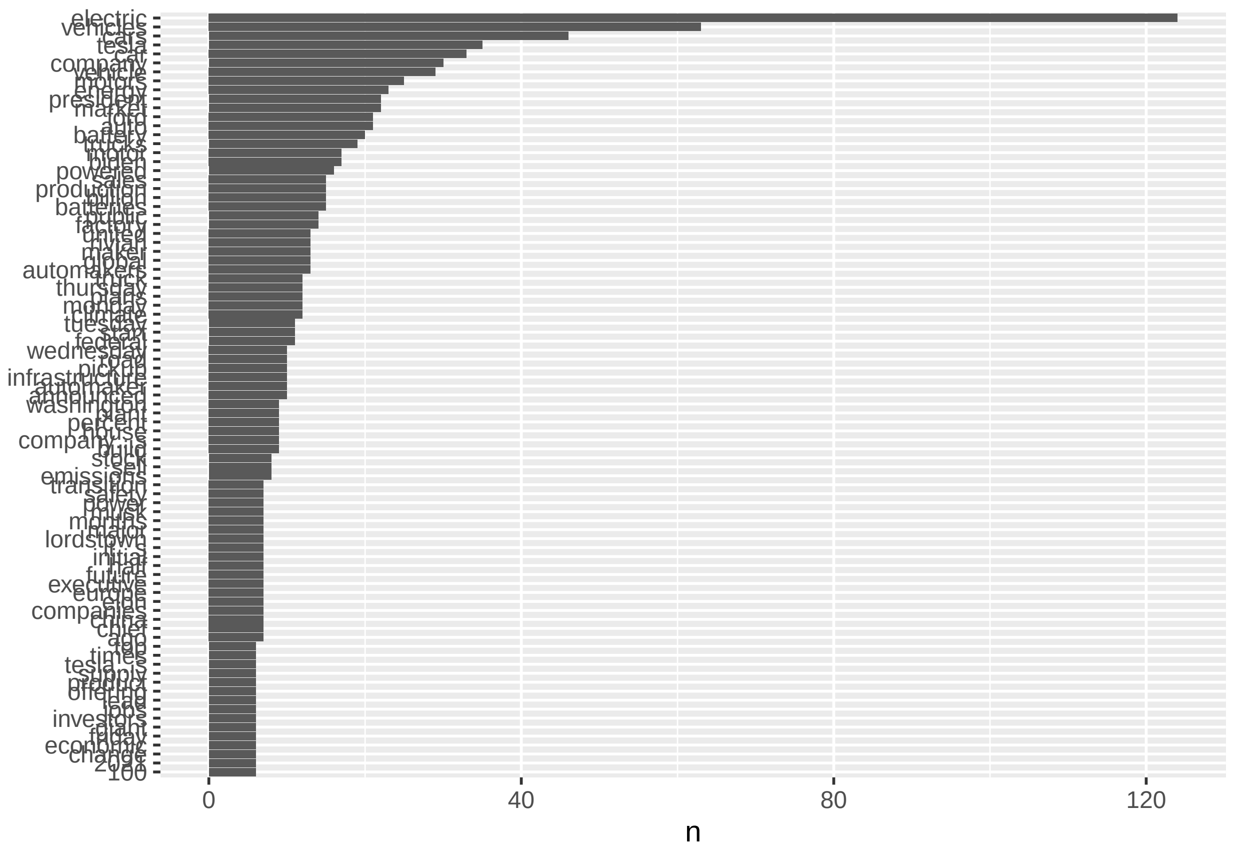
```
data(stop_words)
# stop_words
```

These are the results without stop words included. Still too many words to read, and there are other issues.

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

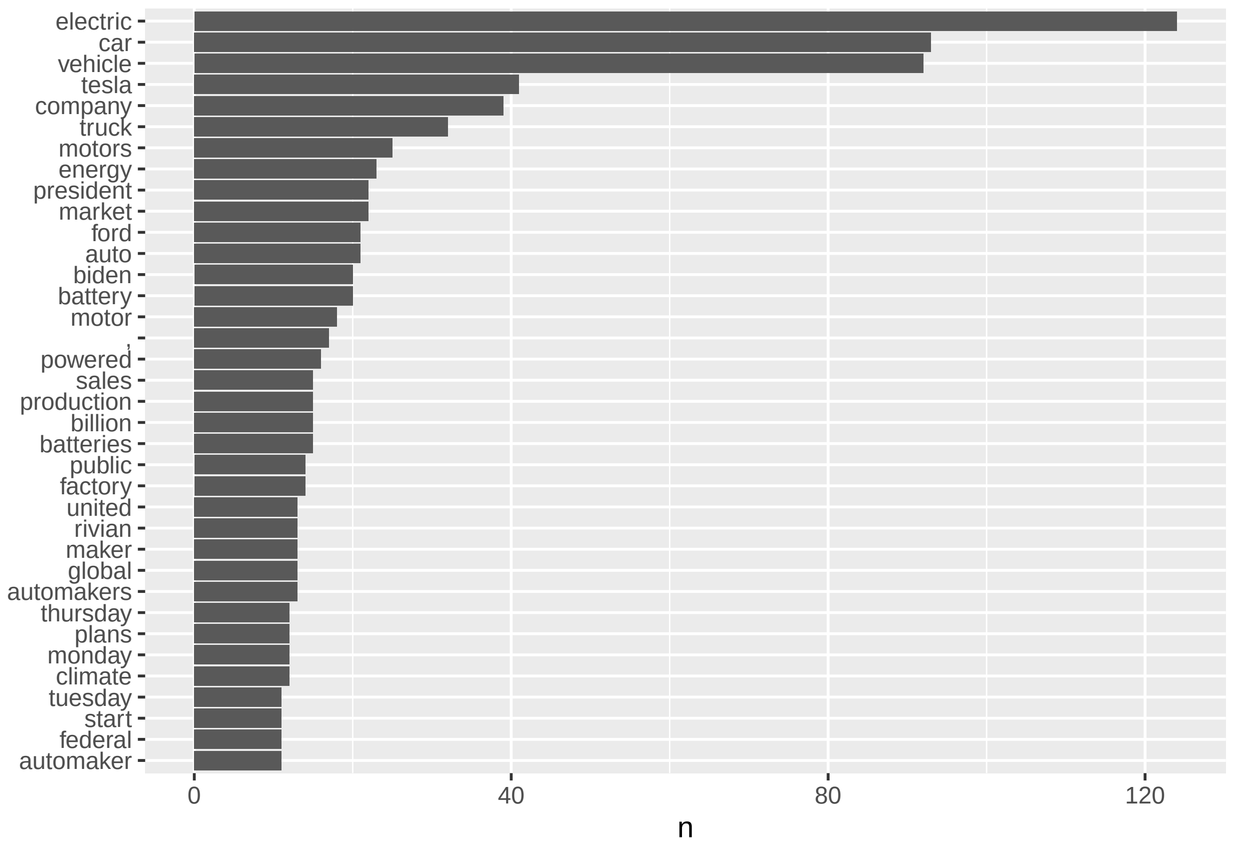
```
## Joining, by = "word"
```

```
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



This code cleans up the above plot by getting rid of number strings, plural versions of words, and words that end in "'s".

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

clean_tokens <- str_remove_all(tokenized$word, "[:digit:]") #remove all numbers

# These lines replace plural words with their singular version
clean_tokens <- str_replace_all(clean_tokens,"car[a-z,A-Z]*","car")
clean_tokens <- str_replace_all(clean_tokens,"vehicle[a-z,A-Z]*","vehicle")
clean_tokens <- str_replace_all(clean_tokens,"truck[a-z,A-Z]*","truck")

clean_tokens <- gsub("’s", '', clean_tokens) # remove all words ending in "'s"

tokenized$clean <- clean_tokens

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

#reassign
tokenized <- tib

#try again
tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 10) %>%
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL)
```

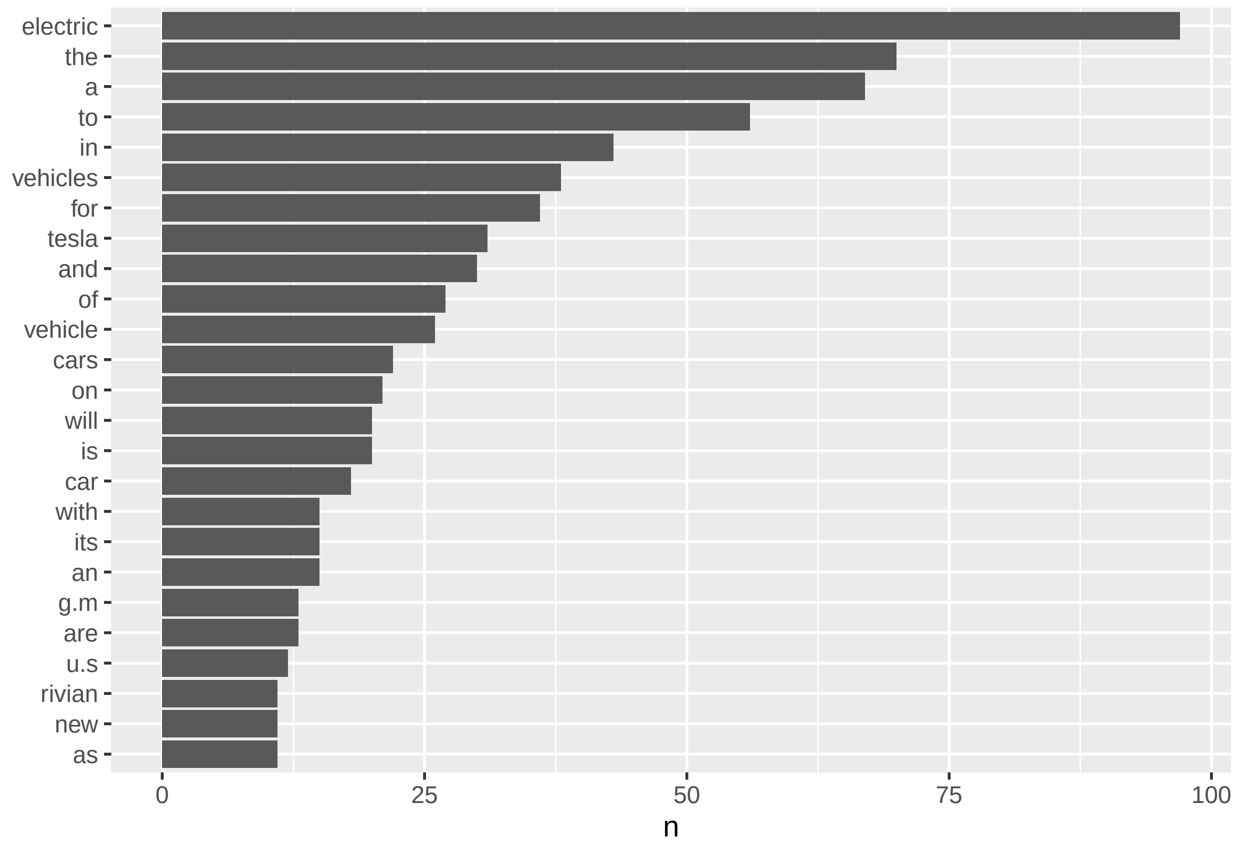


Now let's do the same thing using headlines instead of lead paragraphs.

```
headline <- names(nytDat)[21] #The 6th column, "response.doc.lead_paragraph", is the one we want here.
tokenized <- nytDat %>%
  unnest_tokens(word, headline)

# tokenized[,35]
```

With the stop words 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(y = NULL)
```

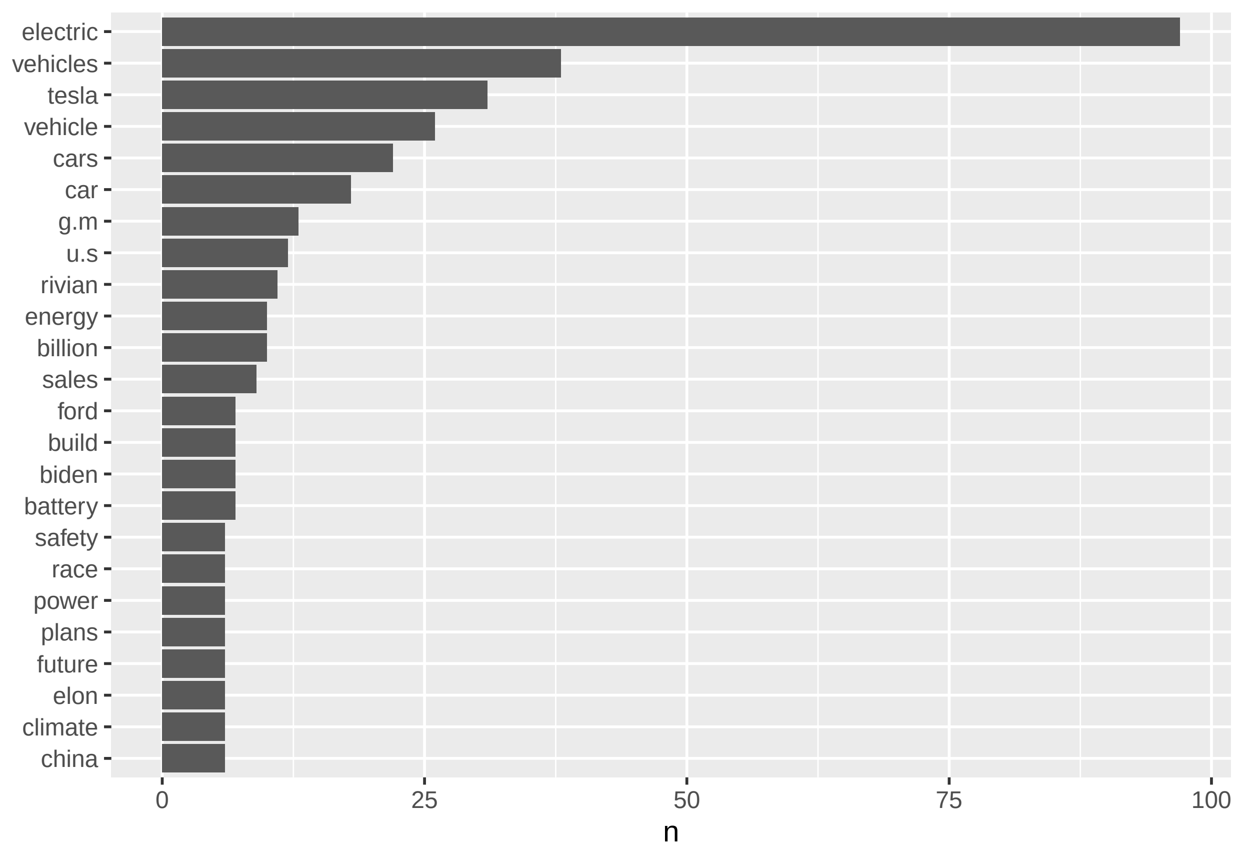


Before taking out numbers, plurals, and words ending in "'s".

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

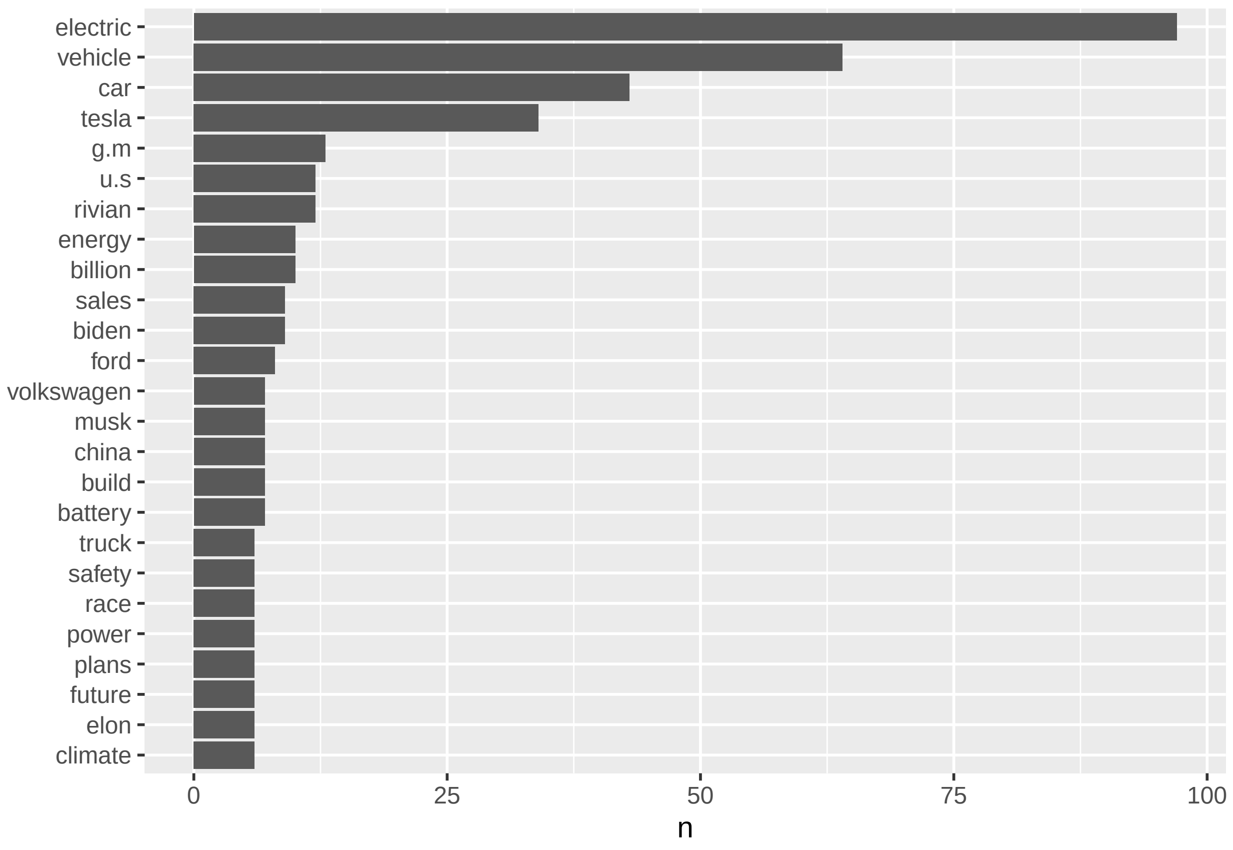
```
## Joining, by = "word"
```

```
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



Final plot.

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

clean_tokens <- str_remove_all(tokenized$word, "[:digit:]") #remove all numbers

# These lines replace plural words with their singular version
clean_tokens <- str_replace_all(clean_tokens,"car[a-z,A-Z]*","car")
clean_tokens <- str_replace_all(clean_tokens,"vehicle[a-z,A-Z]*","vehicle")
clean_tokens <- str_replace_all(clean_tokens,"truck[a-z,A-Z]*","truck")

clean_tokens <- gsub("'s", '', clean_tokens) # remove all words ending in "'s"

tokenized$clean <- clean_tokens

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

#reassign
tokenized <- tib

#try again
tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL)
```



We can see that when we look at the headlines instead of the lead paragraphs we do get a slightly different order of words, however many of the words are the same in both groups. Some things I found interesting though is that the headlines seemed to mention specific companies such as G.M. and Rivian and the lead paragraphs did not, other than for Tesla. Something else interesting was the the lead paragraphs had more mention of Biden where as for the headlines that is lower on the list. For the headlines I lowered the number of times a word needs to appear to show up on the plot, which makes sense because there are going to be less words overall in the headline list.