Lab 1

Allie Cole

4/6/2022

```
#now pull in from teh API

t <- fromJSON("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=IPCC+report&begin_date=2021012
#the string following key= is the API key
#?q is what you are querying for

class(t) #what is teh class of object t
```

```
## [1] "list"
```

```
t <- t %>%
  data.frame() #make it a data frame

#how big is it?
dim(t)
```

```
## [1] 10 33
```

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

```
##  [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"
```

```
t$response.docs.snippet[9] #this will change
```

```
## [1] "Three green-minded entrepreneurs reveal why sustainability is the first step in fighting the cl
```

```
#assign a snippet to x to use as fodder for stringr functions.  You can follow along using the sentence

x <- "Her nomination as secretary of the interior is historic, but as the first Native cabinet member, 

tolower(x) #lowercase
```

```
## [1] "her nomination as secretary of the interior is historic, but as the first native cabinet member
```

```
str_split(x, ','); str_split(x, 't') #split into multiple strings by comma and t
```

```
## [[1]]
## [1] "Her nomination as secretary of the interior is historic"
## [2] " but as the first Native cabinet member"
## [3] " she would have to strike a delicate balance."
```

```
## [[1]]
##  [1] "Her nomina"                "ion as secre"
##  [3] "ary of "                   "he in"
##  [5] "erior is his"              "oric, bu"
##  [7] " as "                      "he firs"
##  [9] " Na"                       "ive cabine"
## [11] " member, she would have " "o s"
## [13] "rike a delica"             "e balance."
```

```
str_replace(x, 'historic', 'without precedent') #replace
```

```
## [1] "Her nomination as secretary of the interior is without precedent, but as the first Native cabin
```

```
str_replace(x, ' ', '_') #first one, replace space with _
```

```
## [1] "Her_nomination as secretary of the interior is historic, but as the first Native cabinet member
```

```
#how do we replace all of them?
str_replace_all(x, ' ', '_')
```

```
## [1] "Her_nomination_as_secretary_of_the_interior_is_historic,_but_as_the_first_Native_cabinet_member
```

```
str_detect(x, 't'); str_detect(x, 'tive') ### is pattern in the string? T/F
```

```
## [1] TRUE
```

```
## [1] TRUE
```

```
str_locate(x, 't'); str_locate_all(x, 'as')
```

```
##      start end
## [1,]    11  11
```

```
## [[1]]
##      start end
## [1,]    16  17
## [2,]    62  63
```

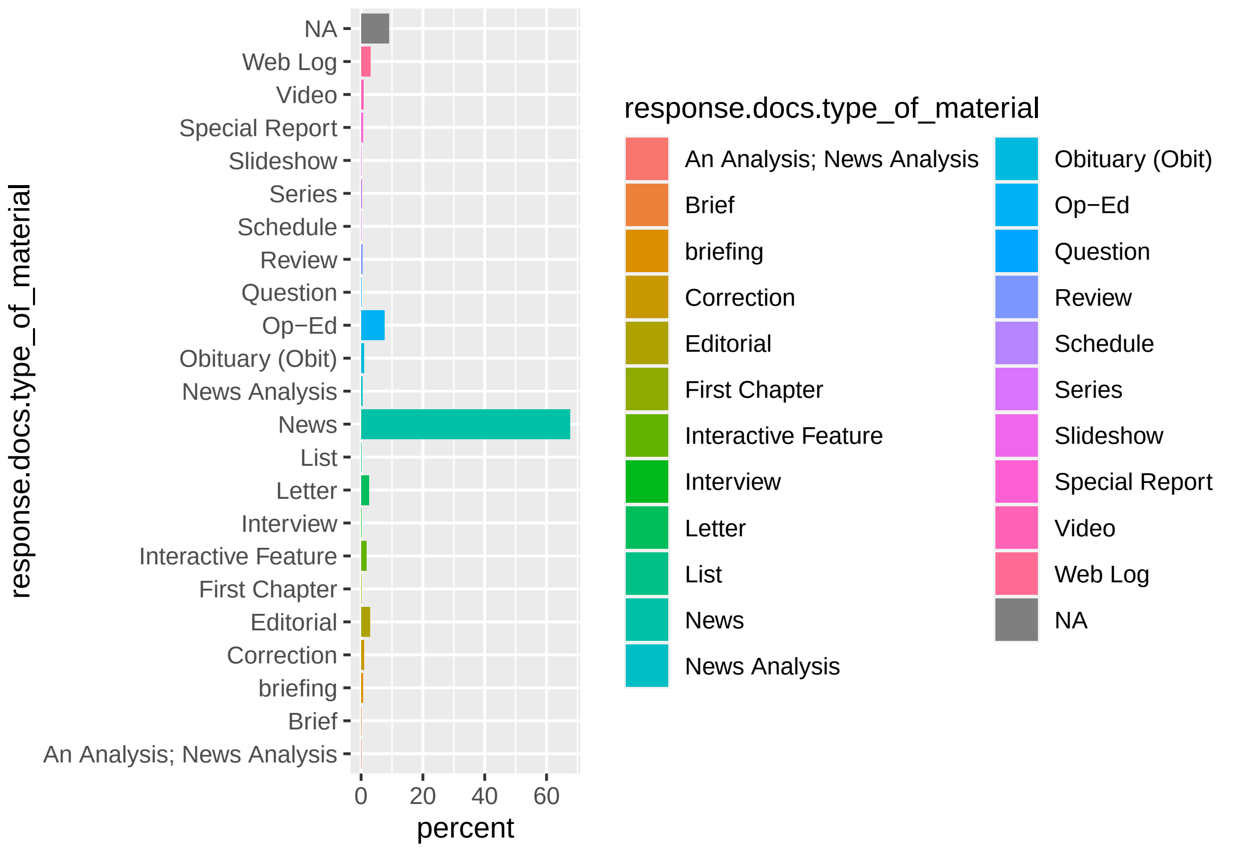
```
term <- "Intergovernmental+Panel+on+Climate+Change" # Need to use + to string together separate words
begin_date <- "20050120"
end_date <- "20220401"

#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=","ZxgXK99oAK0QZG5WXmPazQj7EtiQ5foK", sep="")

#examine our query url
baseurl
```

```
## [1] "http://api.nytimes.com/svc/search/v2/articlesearch.json?q=Intergovernmental+Panel+on+Climate+Cha
```

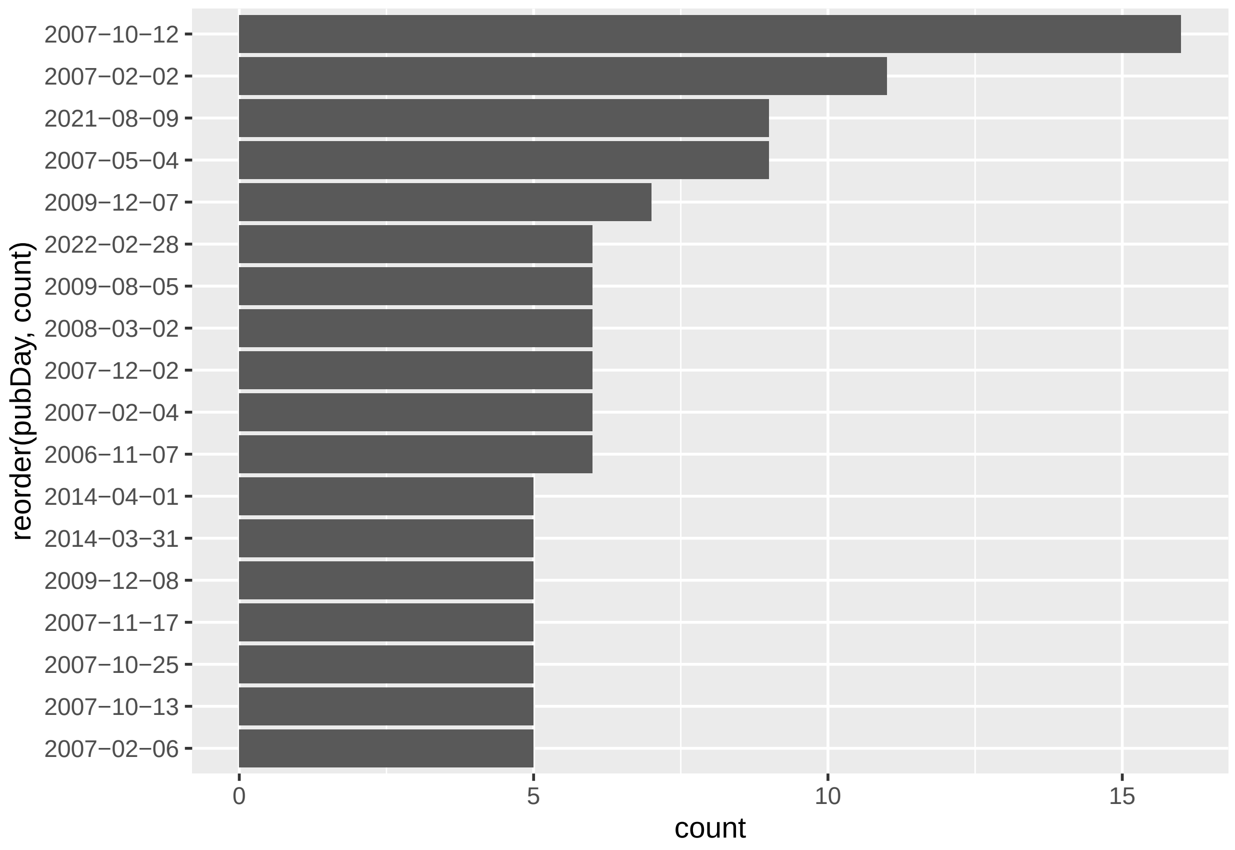
This is commented out so that I can knit the document without the query running again

```
# #this code allows for obtaining multiple pages of query results
#  initialQuery <- fromJSON(baseurl)
# maxPages <- round((initialQuery$response$meta$hits[1] / 10)-1)
#
#
# #looping through so you can get more pages, because NYT gives us one page (ten articles) at a time
#
# 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) #6 allows for it to not time out
# }
# class(nytSearch)
#
#
# nytdat <- rbind_pages(pages)
#  saveRDS(object = nytdat,
#          file = "nytDat.RData")
```

```
nytdat <- readRDS("nytDat.RData")

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
```

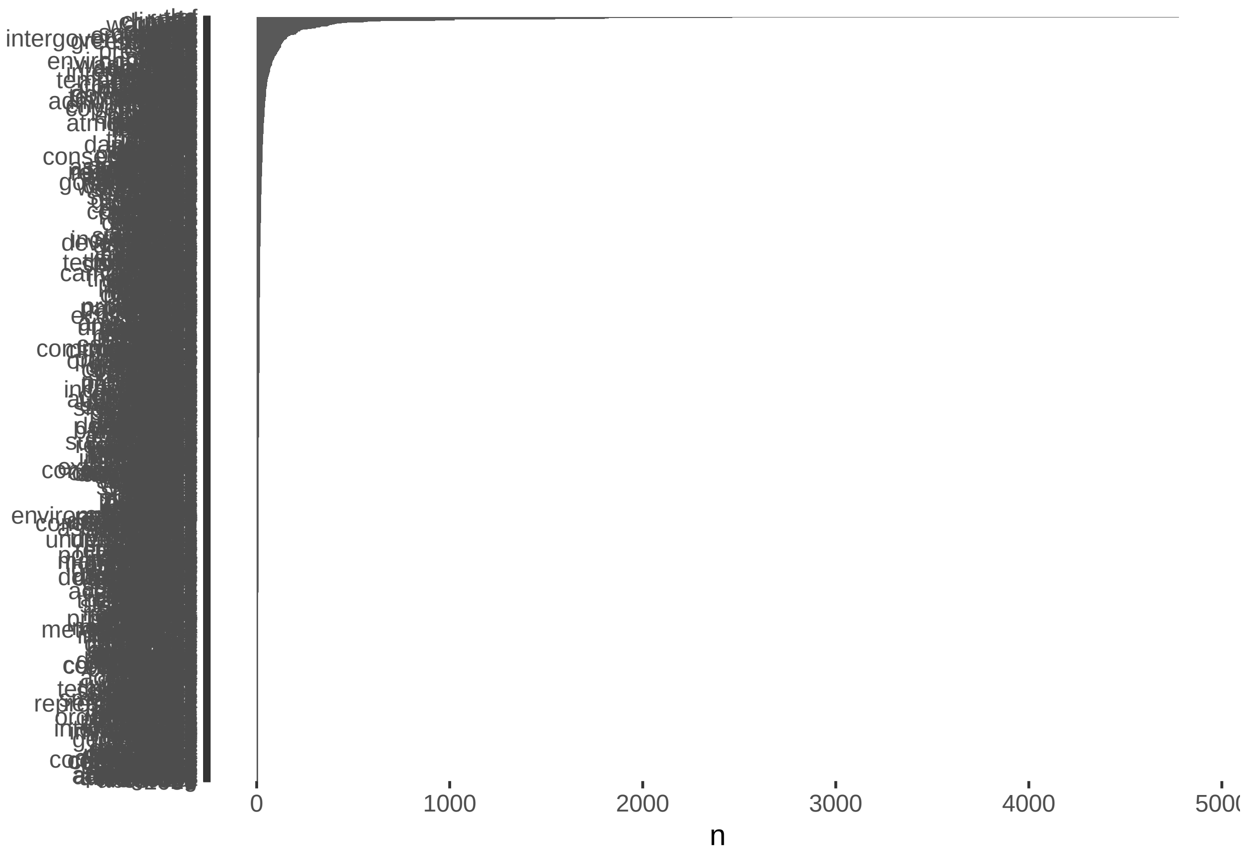


```
#change the number of pubs on a day becase there were a ton
nytdat %>%
  mutate(pubDay=gsub("T.*","",response.docs.pub_date)) %>%
  group_by(pubDay) %>%
  summarise(count=n()) %>%
  filter(count >= 5) %>%
  ggplot() +
  geom_bar(aes(x=reorder(pubDay, count), y=count), stat="identity") + coord_flip()
```

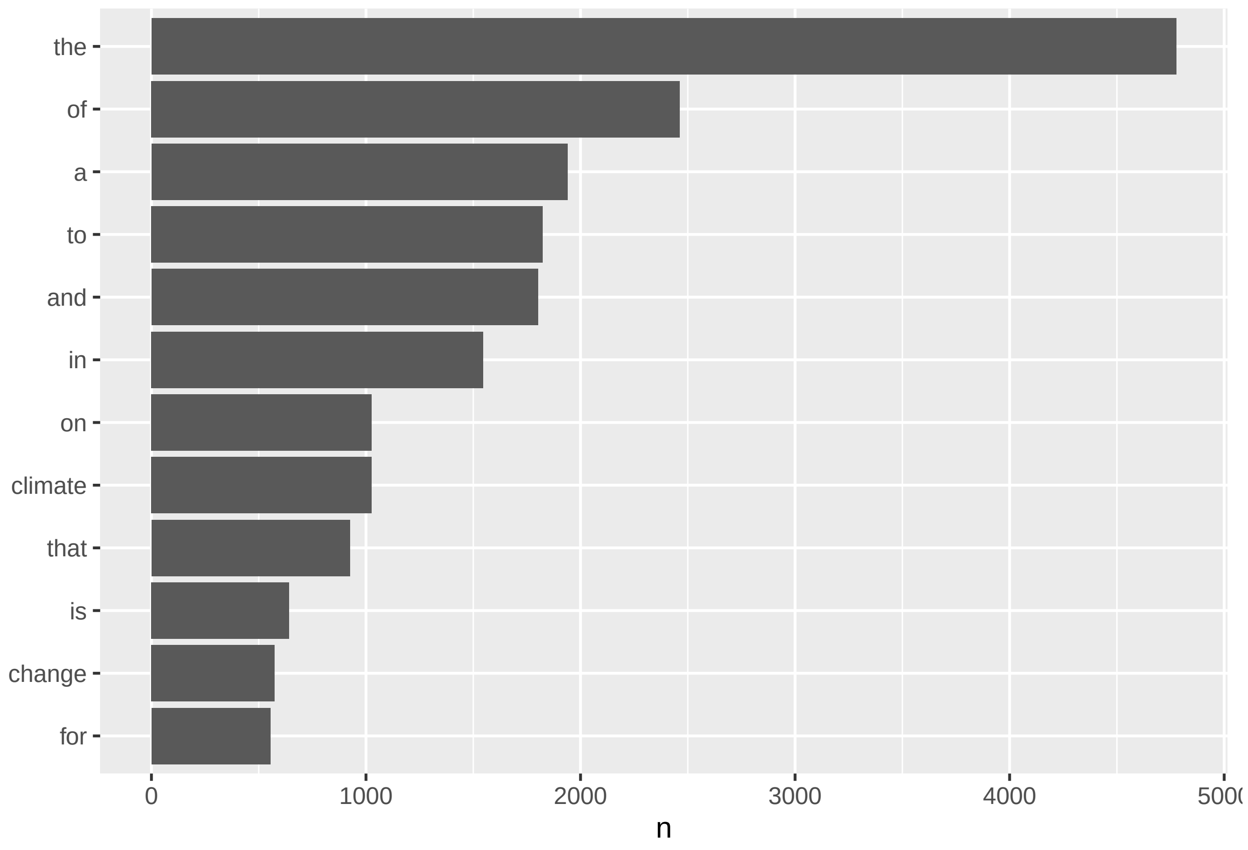


Now we want the first paragraph because thats all nyt gives us

```
paragraph <- names(nytdat)[6] #The 6th column, "response.doc.lead_paragraph", is the one we want here.
tokenized <- nytdat %>%
  unnest_tokens(word, paragraph) #convert text to tidy text format, taking paragrah in and unnest to wo

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



```
#lets run this again but up the word count!

tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 500) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```

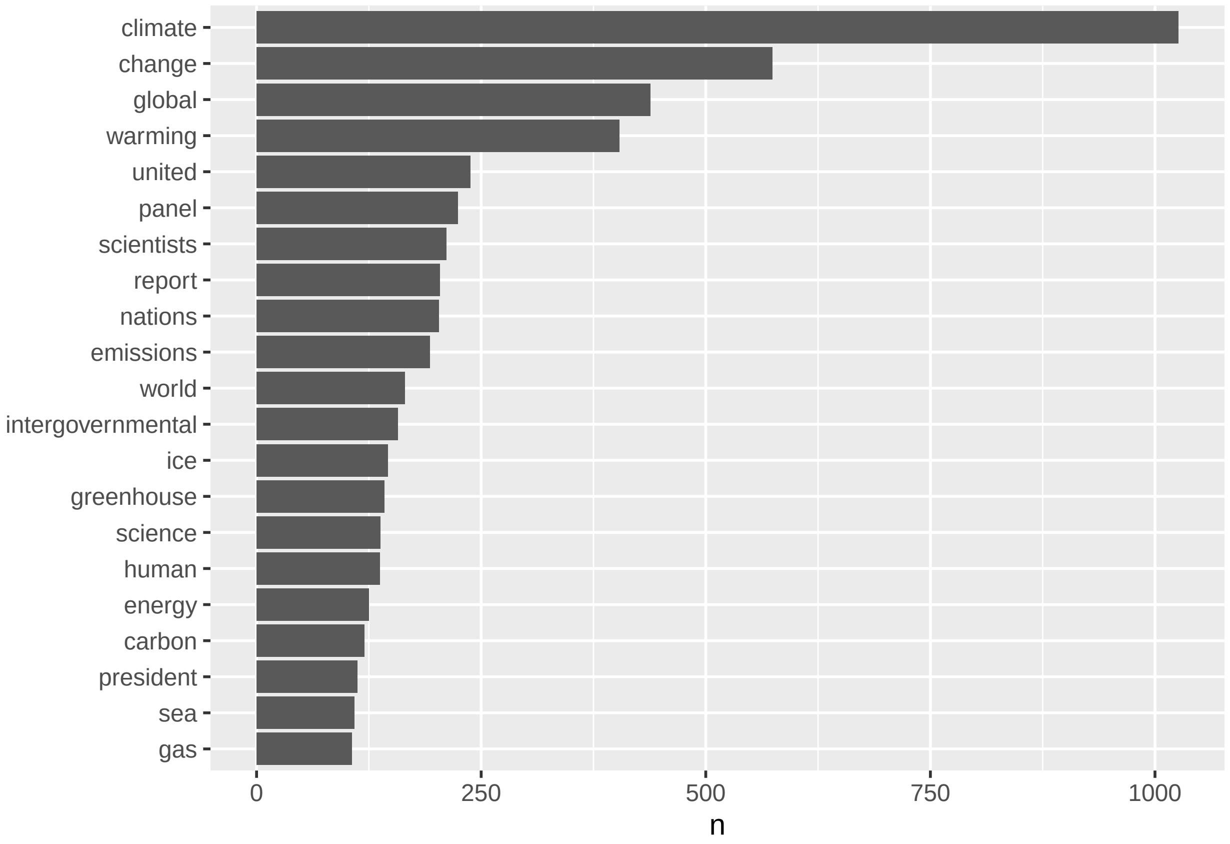


Now we need to get rid of words we don want

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

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

```
#then we can play around with the word count again to make sure its all goood after removing the stop w
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 100) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



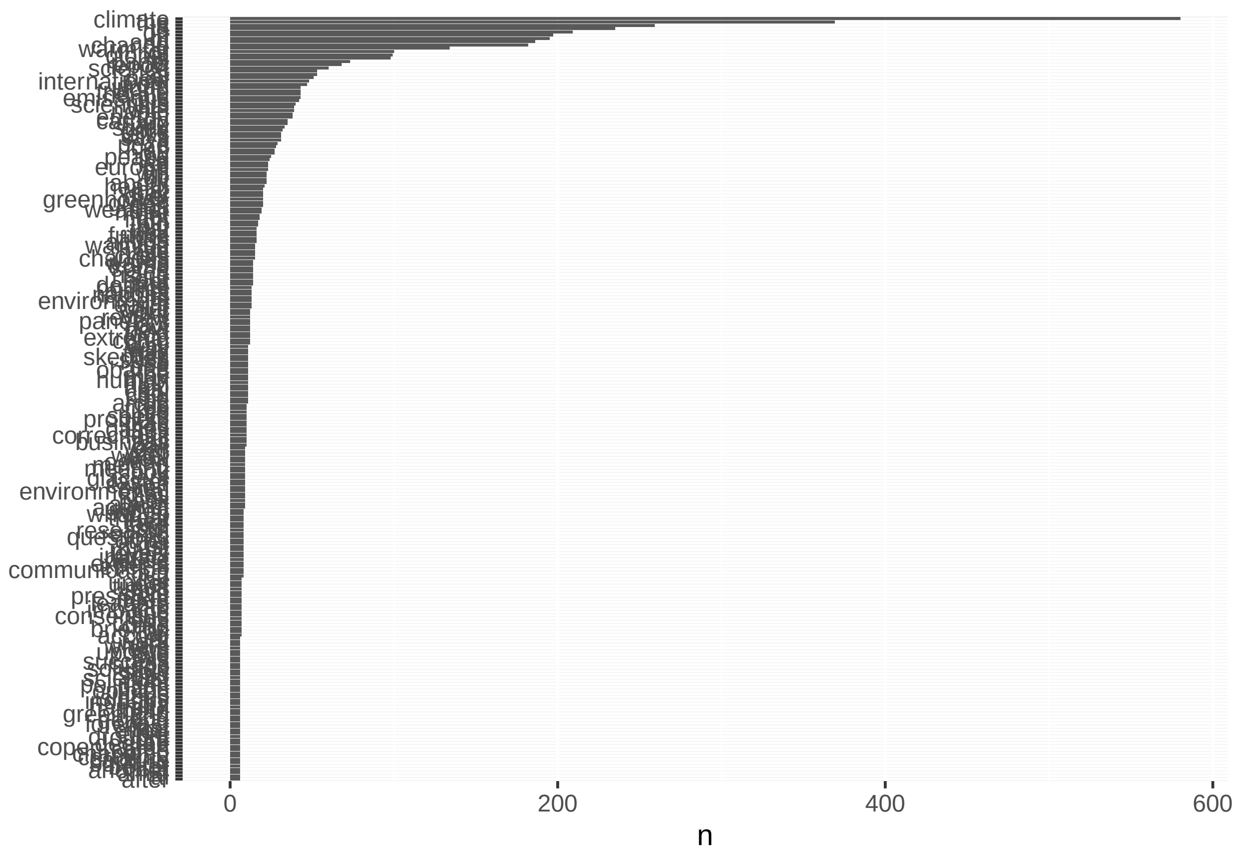
Now removing some words that show up a lot but we dont care about

We know that climate change is important but the name of the report is the Intergovernmental Panel on Climate Change, So we don't really need those words.

```
#inspect the list of tokens (words) but commented out so we dont get an insane list
#tokenized$word

clean_tokens <- str_remove_all(string = tokenized$word,
                               pattern = "intergovernmental")

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "panel")

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "climate")

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "change")

clean_tokens <- str_remove_all(string = clean_tokens,
                               pattern = "i.p.c.c")

#clean_tokens <- str_remove_all(string = clean_tokens,
                              # pattern = " ") #trying to remove the spaces that happened from the abov


clean_tokens <- str_remove_all(clean_tokens, "[:digit:]") #remove all numbers
clean_tokens <- gsub(pattern = "'s", # remove "'s" and replace them with nothing
                     replacement = '',
                     x = clean_tokens)


#check that worked but commented out so we dont get an insane list
#clean_tokens

tokenized$clean <- clean_tokens

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

#reassign
tokenized <- tib

paragraph_plot <- tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 10) %>%
  slice(1:30) %>% # get the top 30 words
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL,
       title = "Paragraphs")
```

Now we do all that over again for the headlines!

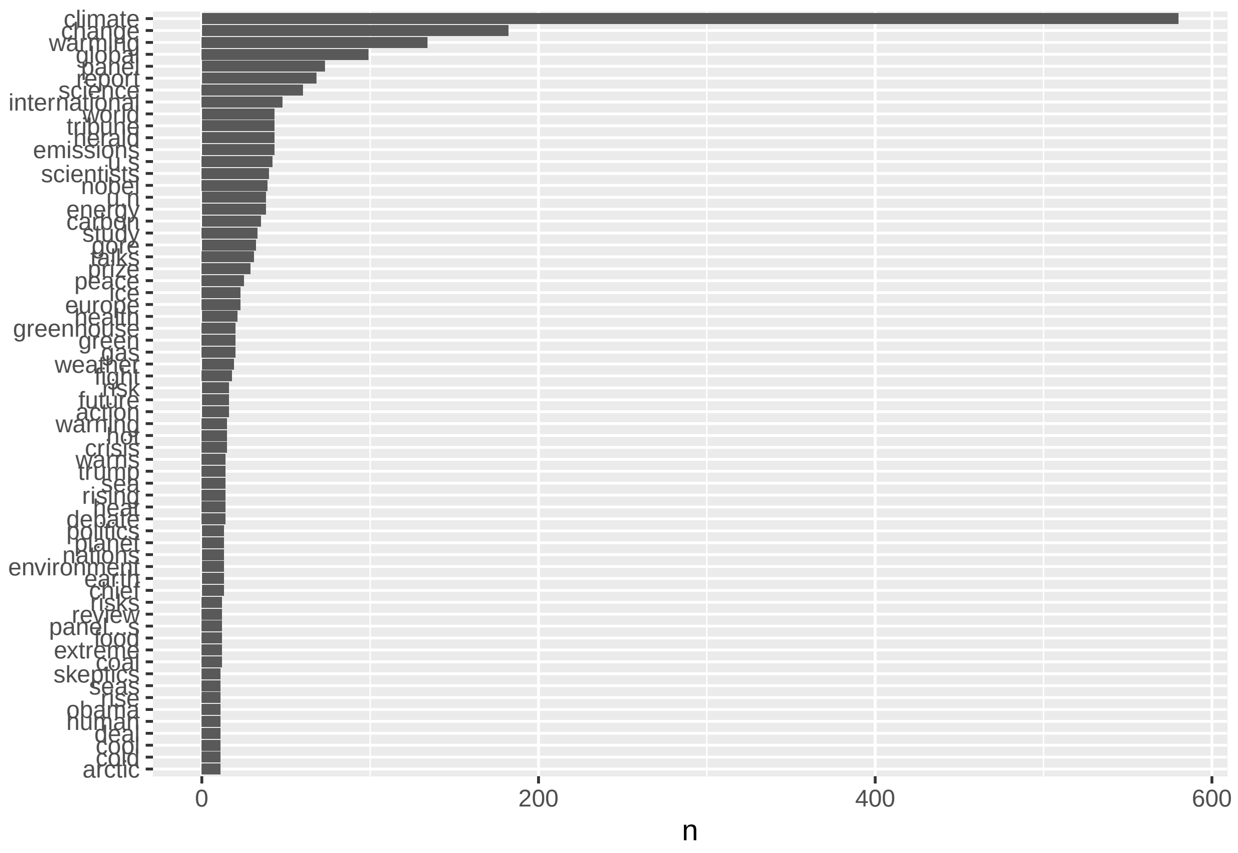
```
headline <- names(nytdat)[20] #The 20th column, "response.docs.headline.main", is the one we want here.
tokenized_2 <- nytdat %>%
  unnest_tokens(word, headline) #convert text to tidy text format, taking paragrah in and unnest to wor

tokenized_2 %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



```
tokenized_2 <- tokenized_2 %>%
  anti_join(stop_words)
```

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

```
#then we can play around with the word count again to make sure its all goood after removing the stop w
tokenized_2 %>%
  count(word, sort = TRUE) %>%
  filter(n > 10) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```

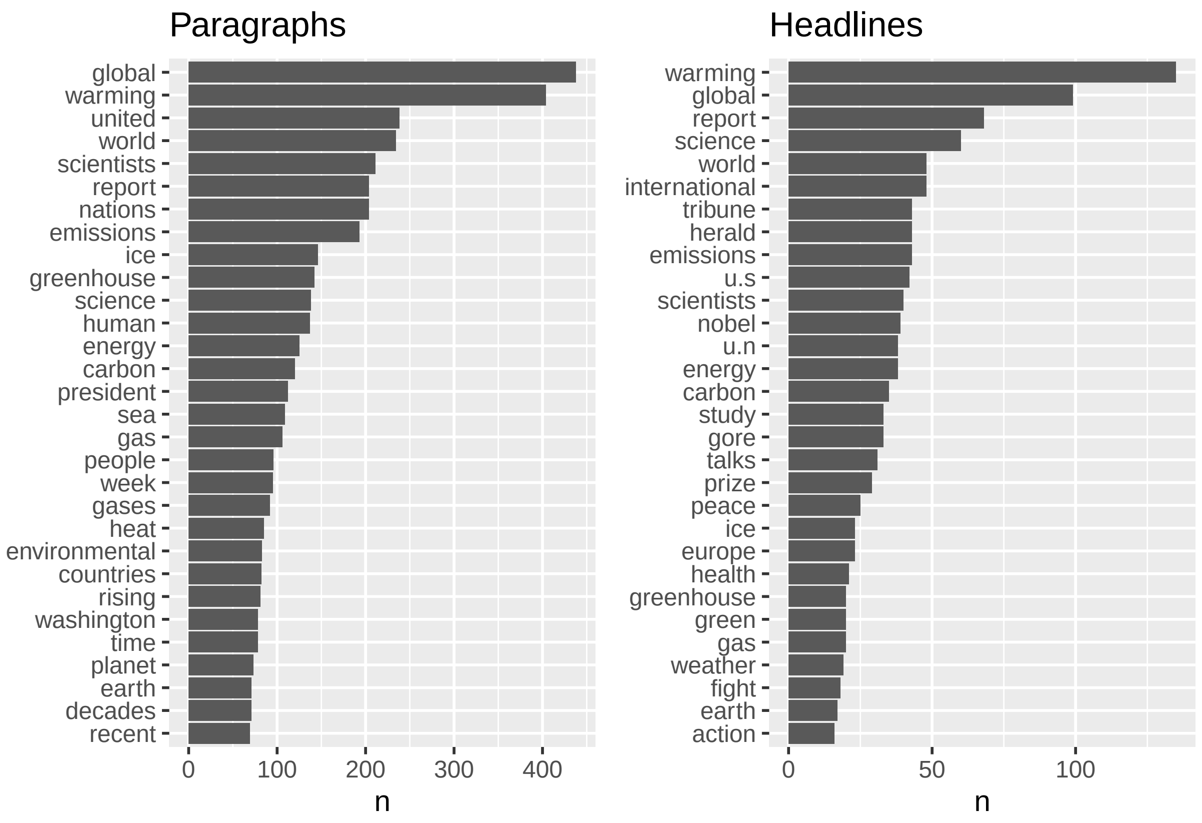


```r
#inspect the list of tokens (words)
#tokenized_2$word

clean_tokens_2 <- str_remove_all(string = tokenized_2$word,
                                 pattern = "intergovernmental")

clean_tokens_2 <- str_remove_all(string = clean_tokens_2,
                                 pattern = "panel")

clean_tokens_2 <- str_remove_all(string = clean_tokens_2,
                                 pattern = "climate")

clean_tokens_2 <- str_remove_all(string = clean_tokens_2,
                                 pattern = "change")

clean_tokens_2 <- str_remove_all(string = clean_tokens_2,
                                 pattern = "i.p.c.c")

#clean_tokens <- str_remove_all(string = clean_tokens,
                               # pattern = " ") #trying to remove the spaces that happened from the abov


clean_tokens_2 <- str_remove_all(clean_tokens_2, "[:digit:]") #remove all numbers

clean_tokens_2 <- gsub(pattern = "'s", # remove "'s" and replace them with nothing
                       replacement = '',
                       x = clean_tokens_2)
```

```
#check that worked
#clean_tokens_2

tokenized_2$clean <- clean_tokens_2

#remove the empty strings
tib <-subset(tokenized_2, clean!="")

#reassign
tokenized_2 <- tib

headline_plot <- tokenized_2 %>%
  count(clean, sort = TRUE) %>%
  filter(n > 10) %>%
  slice(1:30) %>% # get the top 30 words
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL,
       title = "Headlines")
```

Now we can look at the figure side by side to compare them!

```
paragraph_plot + headline_plot
```



There doesn't seem to be much difference in the words used within the paragraphs and the headlines, however the number of times those words are used is different. I do find it interesting that the word "action" is found on the headline list, i imagine it is for a “call to action” or telling people they need to take action, however it is not found in the paragraph. I wonder why not?