Week 2 Lab: Text Analysis in R

Steven Cognac

4/12/2022

Build connection to the New York Times API

```
term <- "wetland" # Need to use + to string together separate words
begin_date <- "20190120"
end_date <- "20220401"
key <- read.delim(here("02_NYT_assignment/nyt_api_key.txt"), header = FALSE)

#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="")
```

Run search query on term "wetland" in the NYT api

Because the NYT api only allows a max return of 10 results at a time, we have to uses the pages query to paginate through our results.

```
#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)
# }
```

We can now bind those pages together into single data frame

```
# nyt_wetland <- rbind_pages(pages)
```

Write/Read .csv file

To avoid having to redo the query each time, let's download our results and save as a .csv file

```r
# export dataframe as csv as backup
# df <- apply(nyt_wetland, MARGIN = 2, FUN = as.character)
# write.csv(df, "data/nyt_wetland.csv", row.names = TRUE)

# read in data so I don't have to re-download
nyt_wetland <- read.csv(here("02_NYT_assignment/data/nyt_wetland.csv"), header = TRUE)

# we received 380 articles from our search with 34 columns
dim(nyt_wetland)
```

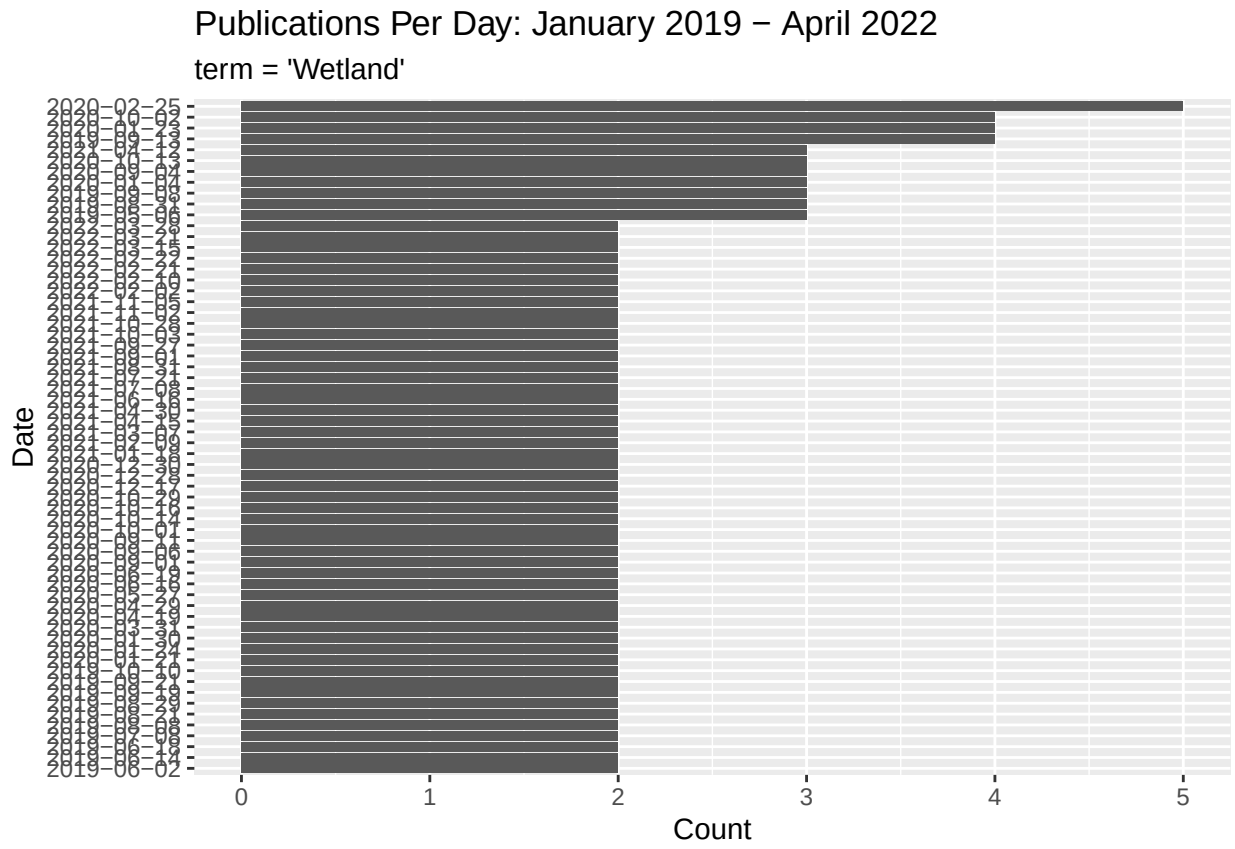
```
## [1] 380  34
```

```r
# names of columns
names(nyt_wetland)
```

```
##  [1] "X"
##  [2] "status"
##  [3] "copyright"
##  [4] "response.docs.abstract"
##  [5] "response.docs.web_url"
##  [6] "response.docs.snippet"
##  [7] "response.docs.lead_paragraph"
##  [8] "response.docs.print_section"
##  [9] "response.docs.print_page"
## [10] "response.docs.source"
## [11] "response.docs.multimedia"
## [12] "response.docs.keywords"
## [13] "response.docs.pub_date"
## [14] "response.docs.document_type"
## [15] "response.docs.news_desk"
## [16] "response.docs.section_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"
## [34] "response.docs.subsection_name"
```

Publications per day chart >=3 hits

```
nyt_wetland %>%
  mutate(pubDay=gsub("T.*","",response.docs.pub_date)) %>%
  group_by(pubDay) %>%
  summarise(count=n()) %>%
  filter(count >= 2) %>%
  ggplot(aes(y = reorder(pubDay, count), x = count)) +
  geom_bar(aes(), stat="identity") +
  labs(title = "Publications Per Day: January 2019 - April 2022",
       subtitle = "term = 'Wetland'",
       x = "Count",
       y = "Date")
```



Graph Type of News Articles

```
# switch to pages
nyt_wetland %>%
  group_by(response.docs.type_of_material) %>%
  summarize(count=n()) %>%
  mutate(percent = (count / sum(count))*100) %>%
```

```
  ggplot() +
  geom_bar(aes(x = percent,
               y = response.docs.type_of_material,
               fill=response.docs.type_of_material),
           stat = "identity") +
  labs(title = "Articles Types: January 2019 - April 2022",
       subtitle = "term = 'Wetland'",
       caption = paste0("NYT stories in search criteria,  N=", nrow(nyt_wetland)))
```

Frequency Plots

```
# the 21st column represents the article title
headline <- names(nyt_wetland)[21]
headline
```

```
## [1] "response.docs.headline.main"
```

```
headline_tokenized <- nyt_wetland %>%
  unnest_tokens(word, headline)

# the 7th column represents lead paragraph
lead_paragraph <- names(nyt_wetland)[7]
lead_paragraph
```

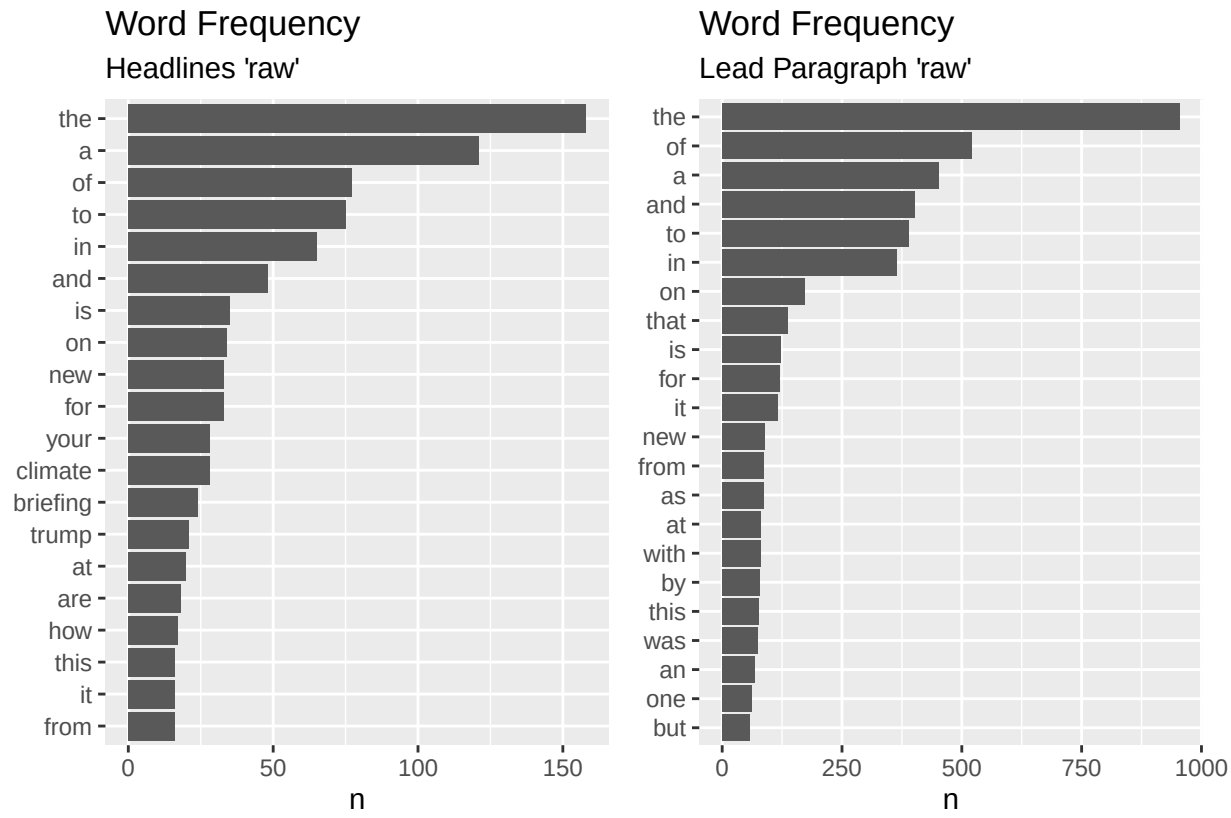
```
## [1] "response.docs.lead_paragraph"
```

```
lead_tokenized <- nyt_wetland %>%
  unnest_tokens(word, lead_paragraph)
```

Raw view of the word frequency plots using the article headline and lead paragraph

```
# article title word frequency plot raw
headline_freq <- headline_tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 15) %>% #illegible with all the words displayed
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Headlines 'raw'",
       y = NULL)

# lead paragraph word frequency plot raw
lead_freq <- lead_tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 55) %>% #illegible with all the words displayed
```

```
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Lead Paragraph 'raw'",
       y = NULL)

headline_freq + lead_freq
```



In our raw Headline and Lead Paragraph frequency plots we see words like 'the', 'a', and other filler words as our top hits.

Let's grab some stop_words to remove fluff in the headline

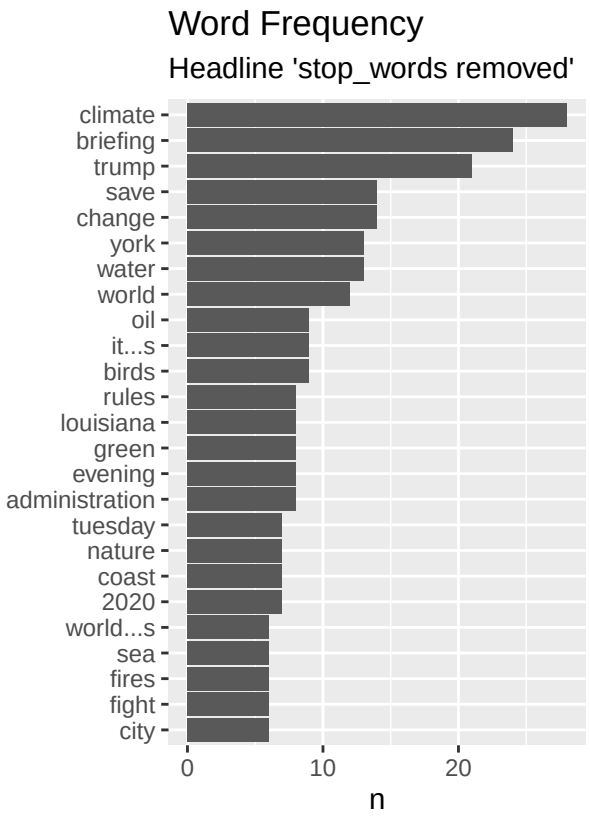
```
data(stop_words)

# removes all the stop_words from the columns in the headline_tokenized words
headline_tokenized <- headline_tokenized %>%
  anti_join(stop_words)
```

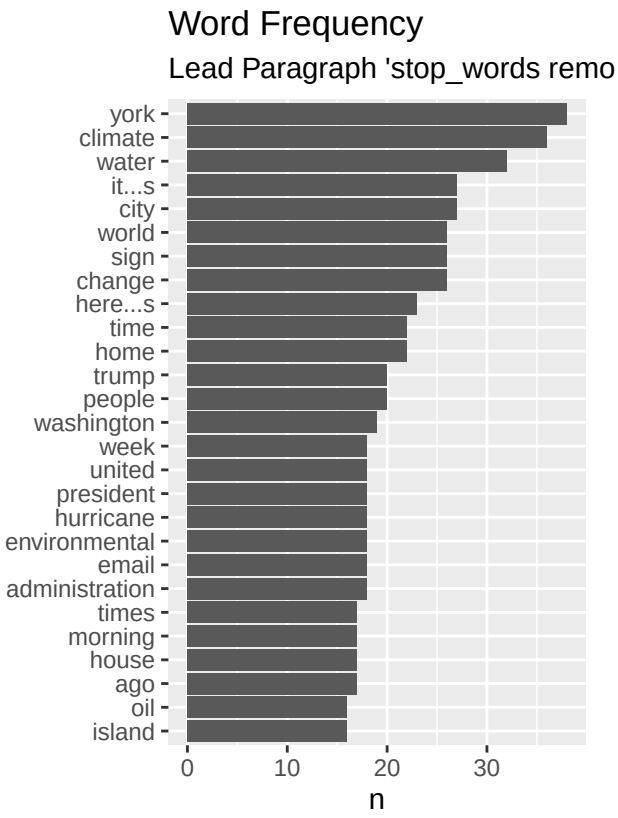
```
## Joining, by = "word"
```

```r
# removes all the stop_words from the columns in the lead_tokenized words
lead_tokenized <- lead_tokenized %>%
  anti_join(stop_words)
```

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

```r
headline_freq2 <- headline_tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Headline 'stop_words removed'",
       y = NULL)

lead_freq2 <- lead_tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Lead Paragraph 'stop_words removed'",
       y = NULL)


headline_freq2 + lead_freq2
```





These plots look better but not great. The word **'climate'** is likely tied to **'climate change'** and contractions are present in both, which isn't very helpful for inferring any meaning from frequency plots. Let's remove those filler words.

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

#Remove "change", "here's" from word bank
clean_tokens <- str_remove_all(headline_tokenized$word, "change")
clean_tokens <- str_remove_all(clean_tokens, "here's")
clean_tokens <- str_remove_all(clean_tokens, "it")

# replace climate with climate change
clean_tokens <- str_replace_all(clean_tokens,"world[a-z,A-Z]*","world") #stem tribe words
clean_tokens <- str_replace_all(clean_tokens,"climate[a-z,A-Z]*","climate change")

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

# removes apostrophe s
clean_tokens <- gsub("'s", '', clean_tokens)
```

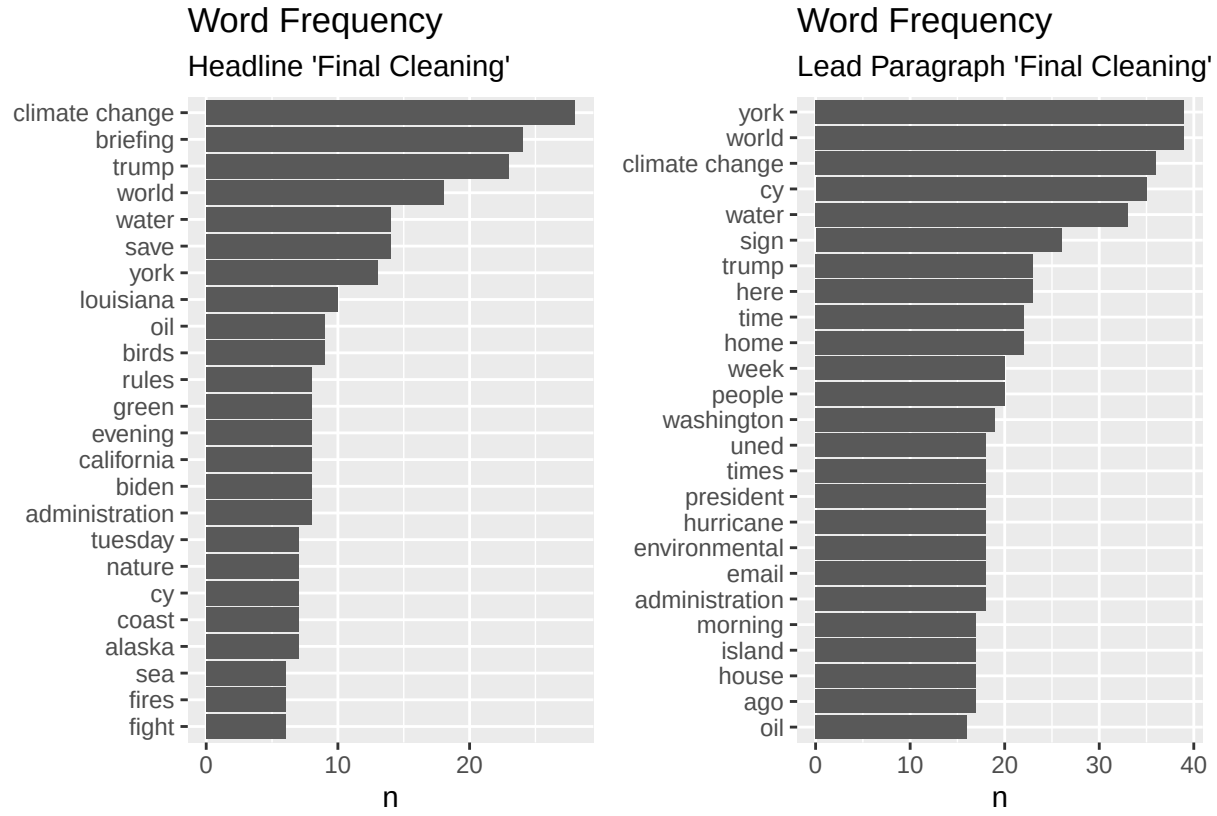
Let's do the same for the lead paragraph

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

#Remove "change", "here's" from word bank
clean_tokens2 <- str_remove_all(lead_tokenized$word, "change")
clean_tokens2 <- str_remove_all(clean_tokens2, "here's")
clean_tokens2 <- str_remove_all(clean_tokens2, "$it$")

# replace climate with climate change
clean_tokens2 <- str_replace_all(clean_tokens2,"world[a-z,A-Z]*","world") #stem tribe words
clean_tokens2 <- str_replace_all(clean_tokens2,"climate[a-z,A-Z]*","climate change")

#remove all numbers
clean_tokens2 <- str_remove_all(clean_tokens2, "[:digit:]")

# removes apostrophe s
clean_tokens2 <- gsub("'s", '', clean_tokens2)
clean_tokens2 <- gsub(",", '', clean_tokens2)
clean_tokens2 <- gsub("it", '', clean_tokens2)
```

```
# add cleaned words to stop_words
headline_tokenized$clean <- clean_tokens
lead_tokenized$clean <- clean_tokens2

#remove the empty strings
tib <-subset(headline_tokenized, clean!="")
tib2 <- subset(lead_tokenized, clean!="")

#reassign
headline_tokenized <- tib
lead_tokenized <- tib2
```

```
#plot the updated word frequency plot
headline_freq3 <- headline_tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 5) %>% #illegible with all the words displayed
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Headline 'Final Cleaning'",
       y = NULL)


lead_freq3 <- lead_tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 15) %>% #illegible with all the words displayed
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(title = "Word Frequency",
       subtitle = "Lead Paragraph 'Final Cleaning'",
       y = NULL)
```

```
headline_freq3 + lead_freq3
```



In these updated Headline frequency plot we have words like 'climate change' briefing', 'trump', and 'world' topping our lists. In our updated Lead Paragraph frequency plot we have words like 'york', 'world', 'climate change' and 'city' topping our list.