

# EDS 231 Topic 2: Text Data in R Assignment

Mia Forsline

4/6/2022

## Set up: Load necessary packages

## 2. Connect to the New York Times API and send a query

- Query phrase: “bark beetle”

```
#create an object called x with the results of our query ("bark beetle")  
# the fromJSON flatten the JSON object, then convert to a data frame  
  
key = "zwtPmtpUJiMGpFgJXz8maQyr3cx6YFdS"  
  
x <- fromJSON("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=bark+beetle&api-key=zwtPmtpUJiMGpFgJXz8maQyr3cx6YFdS",  
              flatten = TRUE) #the string following "key=" is your API key  
  
#convert x to a dataframe  
x <- x %>%  
  data.frame()
```

## Examine a piece of text

- the data object has a variable called “response.docs.snippet” that contains a short excerpt, or “snippet” from the article.
- check the snippet to ensure that the query picked out articles containing the correct query word (“bark beetle”)

```
x$response.docs.snippet[9]
```

## Set some parameters for a bigger query

- searching for articles from 1990 - 2022

```
term <- "bark+beetle" # Need to use + to string together separate words  
begin_date <- "19900101" #start date: Jan 1, 1990  
end_date <- "20220101" #end date: Jan 1, 2022  
key = "zwtPmtpUJiMGpFgJXz8maQyr3cx6YFdS"  
  
#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=bark+beetle&begin_date=19900101&end_d"
```

Obtain multiple pages of query results

- retrieved 35 pages of results
- retrieved 358 articles

```
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)
}
```

```
## 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

## Retrieving page 21

## Retrieving page 22

## Retrieving page 23

## Retrieving page 24

## Retrieving page 25

## Retrieving page 26

## Retrieving page 27

## Retrieving page 28

## Retrieving page 29

## Retrieving page 30

## Retrieving page 31

## Retrieving page 32

## Retrieving page 33

## Retrieving page 34

## Retrieving page 35

```
class(nytSearch)
```

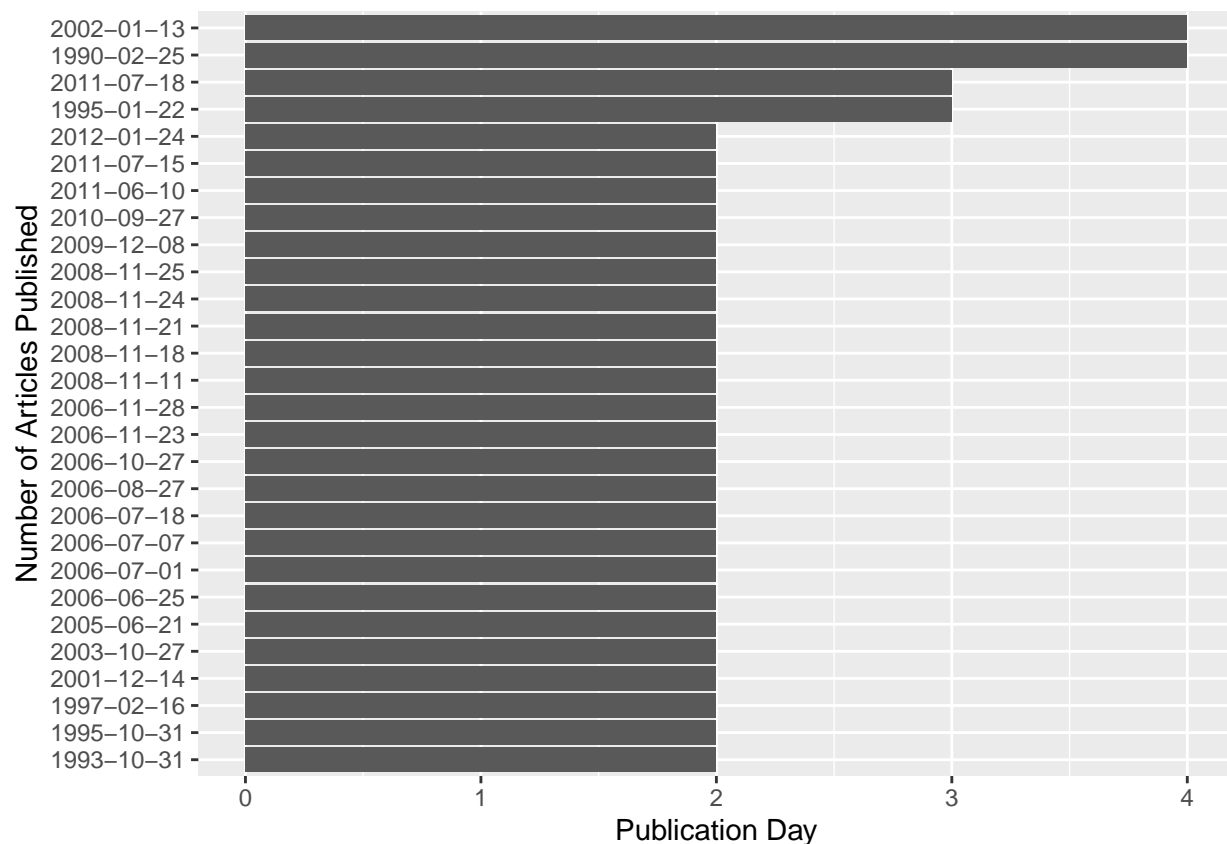
```
## [1] "data.frame"
```

```
#need to bind the pages and create a tibble from nytDat
```

```
nytDat <- rbind_pages(pages)
```

### 3. Visualize publications per 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() +  
  labs(y = "Publication Day",  
       x = "Number of Articles Published")
```



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.

- unnest the words from the first paragraph of each article

```
#names(nytDat)

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

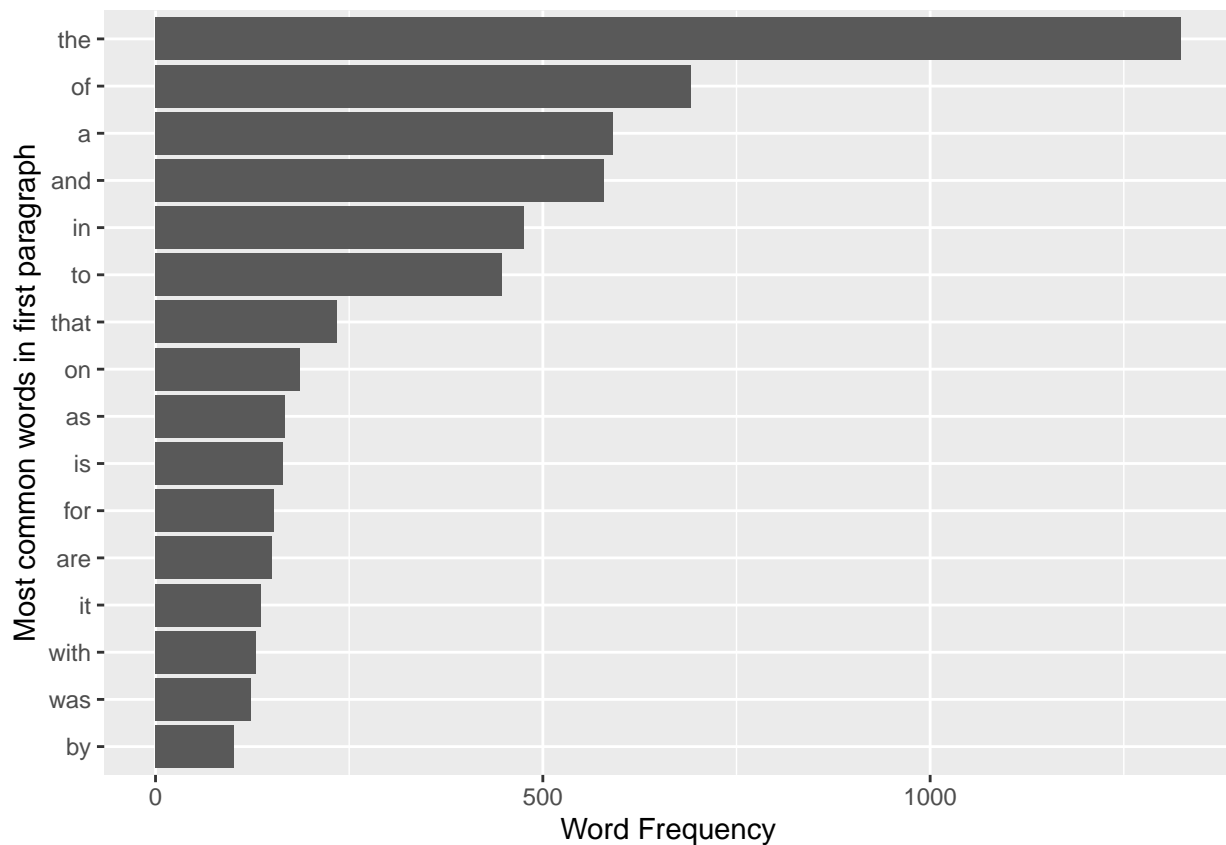
tokenized <- nytDat %>%
  unnest_tokens(word, paragraph)

#tokenized[,34]
```

### 3. Visualize word frequency plots using the first paragraph of each article

- note that the most common words (occurring more than 100 times) are not at all meaningful

```
tokenized %>%
  count(word, sort = TRUE) %>%
  filter(n > 100) %>% #illegible with all the words displayed
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = "Most common words in first paragraph",
       x = "Word Frequency")
```



Since this visual is not very helpful, we transform the corpus in various ways to help filter out less meaningful words from our visual

1. incorporate stop words
2. stem tribe words
3. remove numbers
4. remove possessive words

```
#call common stop words from a lexicon
data(stop_words)
stop_words
```

```
## # A tibble: 1,149 x 2
##   word      lexicon
##   <chr>    <chr>
## 1 a       SMART
## 2 a's     SMART
## 3 able    SMART
## 4 about   SMART
## 5 above   SMART
## 6 according SMART
## 7 accordingly SMART
## 8 across   SMART
## 9 actually SMART
## 10 after   SMART
## # ... with 1,139 more rows
```

```
#use an anti-join to remove the stop words from our tokenized object
tokenized <- tokenized %>%
  anti_join(stop_words)
```

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

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

clean_tokens <- str_replace_all(tokenized$word, "fringe[a-z,A-Z]*", "fringe")

#check to see if the transformation worked
# %>%
# as.data.frame()
# clean_tokens$.

clean_tokens <- str_remove_all(clean_tokens, "[:digit:]")

#check to see if the transformation worked
# %>%
# as.data.frame()
# clean_tokens$.

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

```

clean_tokens <- gsub("[^A-z]s", "", clean_tokens)

# %>%
#   as.data.frame()
# clean_tokens$.

#Put clean tokens into the `tokenized` dataframe

tokenized$clean <- clean_tokens

```

## Visualize word frequency after transforming/cleaning

- remove the empty strings
- limit the graph to only include words that occur over 20 times

```

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

#reassign
tokenized <- tib

#visualize again
tokenized %>%
  count(clean, sort = TRUE) %>%
  filter(n > 20) %>%
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = "Most common words in first paragraphs",
       x = "Word Occurrence Frequency") +
  theme_classic()

# ggsave(filename = "words_par.jpg",
#         width = 4,
#         height = 3,
#         units = c("in"),
#         dpi = 300)

```

## 4. Visualize publications per day ?

## 4. Visualize word frequency using headlines

```

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

tokenized_h <- nytDat %>%
  unnest_tokens(word, headline)

#tokenized_h[,34]

```

## Transform headline words

```
#remove stop words
tokenized_h <- tokenized_h %>%
  anti_join(stop_words)

## Joining, by = "word"

#stem tribe words
clean_tokens <- str_replace_all(tokenized_h$word, "land[a-z,A-Z]*", "land")

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

clean_tokens <- gsub("'s", '', clean_tokens)

tokenized_h$clean <- clean_tokens

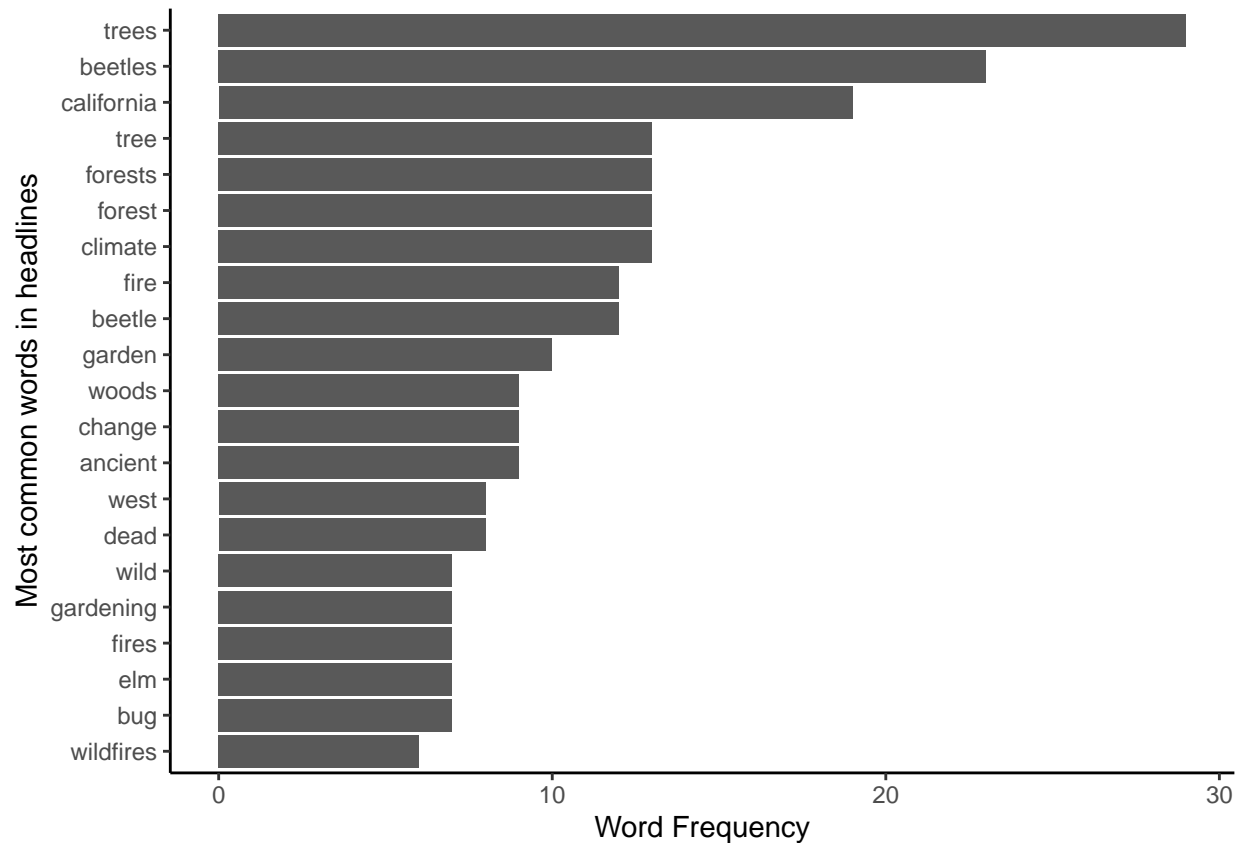
#remove the empty strings
tib <- subset(tokenized_h, clean!="")

#reassign
tokenized_h <- tib
```

## 4. Visualize common words based on headlines

```
tokenized_h %>%
  count(clean, sort = TRUE) %>%
  filter(n > 5) %>%
  mutate(clean = reorder(clean, n)) %>%
  ggplot(aes(n, clean)) +
  geom_col() +
  labs(y = NULL) +
  theme_classic() +
  labs(y = "Most common words in headlines",
       x = "Word Frequency")
```





```
# ggsave(filename = "words_headlines.jpg",
#         width = 4,
#         height = 3,
#         units = c("in"),
#         dpi = 300)
```

#### 4. Comparing word frequencies of NY Times articles' first paragraphs and headlines

The most immediate difference between the word frequency plots using either NY Times articles' first paragraphs or headlines is that the plot visualizing word frequency of first paragraphs has much larger word frequencies. For example, the most common word "trees" is used approximately 75 times in first paragraphs while the other plot shows the word "trees" was found in headlines only approximately 30 times. This makes sense given that paragraphs are much longer than headlines, so it's just more likely for a common word like "trees" to be used.

Both plots shared exact words such as "trees," "beetle(s)," and "climate." However, there were some unique top words. For instance, headlines reported "elm" and "ancient" while first paragraphs reported "york" and "pine."

The overall data distribution shapes looked very similar with one or two extremely common words and a gentle slope downward towards less frequent words.