

4Chan Scraper v2

Anon

2023-08-24

Load in the needed libraries.

```
library("rvest")
library("tidyverse")
library("ggplot2")
library("wordcloud")
library("tidytext")
```

This chunk is scraping all the internal links on each page of pol.

```
#Page 1 Scrape
pol_threads1 <- read_html("https://boards.4channel.org/pol/") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 2 Scrape
pol_threads2 <- read_html("https://boards.4channel.org/pol/2") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 3 Scrape
pol_threads3 <- read_html("https://boards.4channel.org/pol/3") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 4 Scrape
pol_threads4 <- read_html("https://boards.4channel.org/pol/4") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 5 Scrape
pol_threads5 <- read_html("https://boards.4channel.org/pol/5") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 6 Scrape
pol_threads6 <- read_html("https://boards.4channel.org/pol/6") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 7 Scrape
pol_threads7 <- read_html("https://boards.4channel.org/pol/7") %>%
  html_elements("a") %>%
```

```

html_attr('href')

#Page 8 Scrape
pol_threads8 <- read_html("https://boards.4channel.org/pol/8") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 9 Scrape
pol_threads9 <- read_html("https://boards.4channel.org/pol/9") %>%
  html_elements("a") %>%
  html_attr('href')

#Page 10 Scrape
pol_threads10 <- read_html("https://boards.4channel.org/pol/10") %>%
  html_elements("a") %>%
  html_attr('href')

```

Combining all the internal links into 1 data frame.

```

df_pol <- c(pol_threads1,
            pol_threads2,
            pol_threads3,
            pol_threads4,
            pol_threads5,
            pol_threads6,
            pol_threads7,
            pol_threads8,
            pol_threads9,
            pol_threads10)

```

Next, put the scraped links into a table (called tibble).

```
pol_table <- tibble(txt = df_pol)
```

Choosing all of the links that look like: “thread/this-is-a-thread”. More precisely, it’s the href internal links on pol that are named “thread” with a trailing title, separated by a slash.

```

df_links <- pol_table %>%
  filter(str_detect(txt, "(thread/[0-9]{6,}/[a-z]{1,})"))

```

The href internal links don’t have the appropriate full URL link, so it is added in the code block below. This is appending on “https://boards.4chan.org/pol/” before the “thread/this-is-a-thread” section.

```
df_links$txt <- paste("https://boards.4chan.org/pol/", df_links$txt, sep = "")
```

This code will “apply” the “read_html”, “html_elements”, and “html_text”, to each row in the data frame.

```

threads <- lapply(df_links$txt, function(x) {
  read_html(x) %>%
  html_text()})

```

Now, a table (called tibble) needs to be constructed for tidytext to work properly.

```
threads_tibble <- tibble(txt = threads)
```

Break up all of the sentences into single words.

```

tidy_pol <- threads_tibble %>%
  unnest_tokens(word, txt, format = "text")

```

Please excuse this enormous filter list. This cleans up the noise from scraping the entire HTML page. In the filter are other board names, and various text that detract from the words of interest. There can be more words added, but the most frequently appearing 200 words appear to be unaffected when using this filter list.

```
tidy_pol_fixed <- tidy_pol %>%
  filter(!word %in% stop_words$word
    & !word == "fucking"
    & !word == "https"
    & !word == "shit"
    & !word == "id"
    & !word == "anonymous"
    & !word == "wed"
    & !word == "kb"
    & !word == "var"
    & !word == "png"
    & !word == "mobile"
    & !word == "mb"
    & !word == "catalog"
    & !word == "settings"
    & !word == "display"
    & !word == "advertise"
    & !word == "pass"
    & !word == "bottom"
    & !word == "pol"
    & !word == "shit"
    & !word == "jpg"
    & !word == "view"
    & !word == "vp"
    & !word == "ad"
    & !word == "tv"
    & !word == "fit"
    & !word == "post"
    & !word == "thread"
    & !word == "hr"
    & !word == "gif"
    & !word == "webm"
    & !word == "incorrect"
    & !word == "tg"
    & !word == "comments"
    & !word == "search"
    & !word == "top"
    & !word == "site"
    & !word == "home"
    & !word == "reply"
    & !word == "board"
    & !word == "politically"
    & !word == "return"
    & !word == "time"
    & !word == "owned"
    & !word == "added"
    & !word == "vip"
    & !word == "users"
    & !word == "rules"
    & !word == "legal"
```

```
& !word == "lgbt"
& !word == "lit"
& !word == "file"
& !word == "mu"
& !word == "hide"
& !word == "fa"
& !word == "responsibility"
& !word == "style"
& !word == "options"
& !word == "table"
& !word == "page"
& !word == "serve"
& !word == "contact"
& !word == "images"
& !word == "international"
& !word == "poster"
& !word == "people"
& !word == "true"
& !word == "bant"
& !word == "vm"
& !word == "vmg"
& !word == "vrpg"
& !word == "vst"
& !word == "read"
& !word == "news"
& !word == "image"
& !word == "posts"
& !word == "jp"
& !word == "sci"
& !word == "vg"
& !word == "po"
& !word == "toy"
& !word == "vt"
& !word == "wg"
& !word == "biz"
& !word == "ck"
& !word == "desktop"
& !word == "enable"
& !word == "feedback"
& !word == "int"
& !word == "verification"
& !word == "respective"
& !word == "vr"
& !word == "wsg"
& !word == "aco"
& !word == "adv"
& !word == "delete"
& !word == "cm"
& !word == "disable"
& !word == "bfutababurichantomorrowphoton"
& !word == "cgl"
& !word == "comlen"
& !word == "cooldowns"
```

```

& !word == "copyrights"
& !word == "cssversion"
& !word == "diy"
& !word == "gd"
& !word == "hc"
& !word == "ic"
& !word == "incorrectreturn"
& !word == "jsversion"
& !word == "maxfilesize"
& !word == "maxlines"
& !word == "mlp"
& !word == "payment"
& !word == "postform"
& !word == "pw"
& !word == "qa"
& !word == "qst"
& !word == "recaptcha"
& !word == "refresh"
& !word == "replyreturn"
& !word == "soc"
& !word == "sp"
& !word == "trademarks"
& !word == "trv"
& !word == "uploaded"
& !word == "hm"
& !word == "xs"
& !word == "yotsubayotsuba"
& !word == "boards"
& !word == "faq"
& !word == "announcementcrypto"
& !word == "bolsheviknatonazihippiepiraterepublicantask"
& !word == "bypass"
& !word == "capitalistanarchistblack"
& !word == "flaggeographic"
& !word == "huggerunited"
& !word == "locationanarcho"
& !word == "login"
& !word == "nationalistconfederatecommunistcataloniademocrateuropeanfascistgadsdengayjihadikek"
& !word == "nationswhite"
& !word == "refreshpost"
& !word == "supremacistfileplease"
& !word == "ztemplartree"
& !word == "posters"
& !word == "wpjizlog"
& !word == "xxfbsv"
& !word == "wsr"
& !word == "thu"
& !grepl("[^A-Za-z]", word))

```

This code block below is sorting all of the words by frequency, and assigning it to a new data frame. Also, it prints the top 10 most frequent words.

```

tidy_pol_fixed2 <- tidy_pol_fixed %>%
  count(word, sort = TRUE) %>%

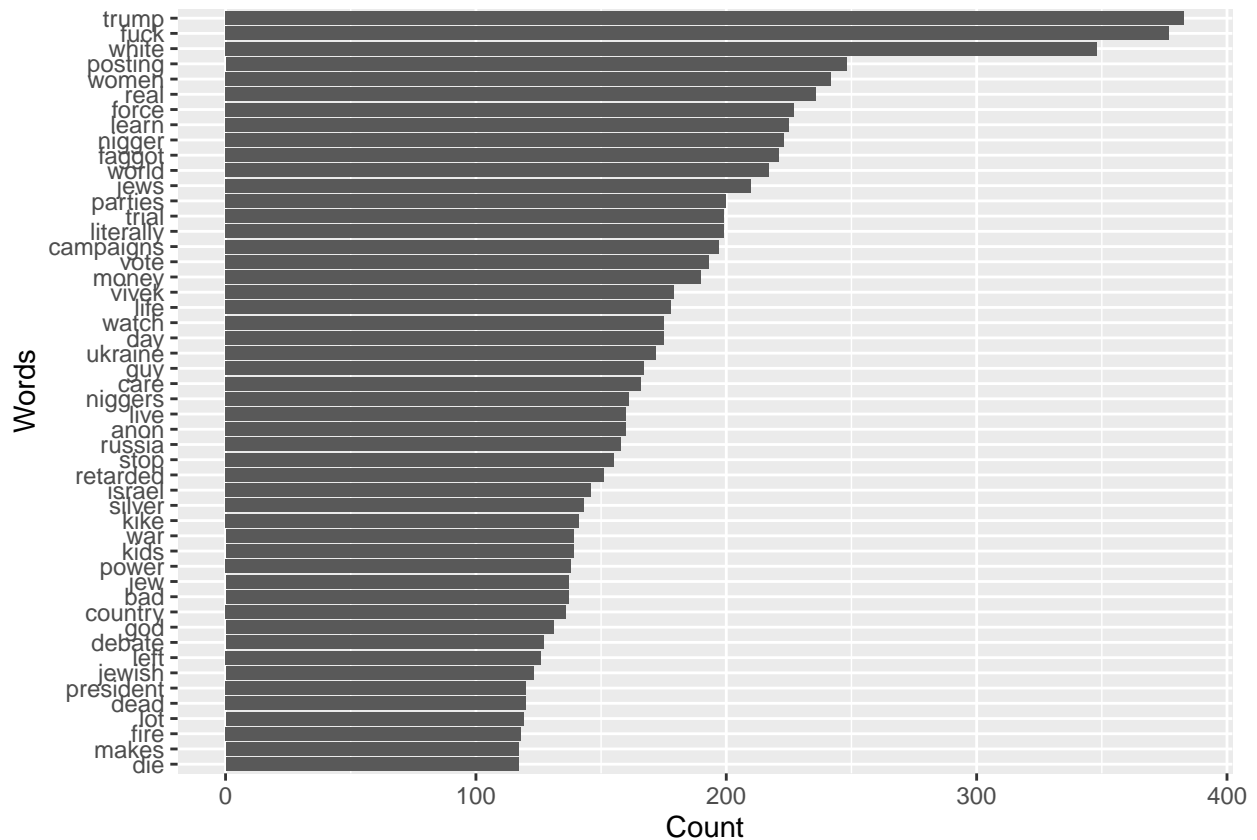
```

```
print(n = 10)
```

```
## # A tibble: 18,661 x 2
##   word      n
##   <chr>   <int>
## 1 trump   383
## 2 fuck    377
## 3 white   348
## 4 posting 248
## 5 women   242
## 6 real    236
## 7 force   227
## 8 learn   225
## 9 nigger   223
## 10 faggot  221
## # i 18,651 more rows
```

Time to Visualize.

```
tidy_pol_fixed2 %>%
  top_n(50) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab("Words") +
  ylab("Count") +
  coord_flip()
```



```
tidy_pol_fixed2 %>%
  with(wordcloud(word, n, max.words = 100, random.order = FALSE, rot.per = 0.0,
    colors = brewer.pal(8, "Dark2")))
```



DON'T FORGET TO SAVE YOUR DATA TO A CSV. CHANGE THE "DATE" IN THE PATH BELOW. MAKE SURE TO ALSO CHANGE THE NAME EACH TIME YOU RUN THE CODE SO YOU DON'T OVERWRITE OLD DATA THAT YOU SAVED. LEAVE THIS PART COMMENTED OUT SO YOU DON'T ACCIDENTIALLY RUN IT.

```
write.csv(tidy_pol_fixed2, "~/Documents/Stats/4Chan Scraper/Aug-22-2023-1116h.csv", row.names=FALSE)
```