Automatic Sampling and Analysis of YouTubeData

Collecting Data With the tuber Package for R

Julian Kohne
Johannes Breuer
M. Rohangis Mohseni

2022-02-22

Required Libraries for This Session

```r
library(tidyverse)
library(lubridate)
library(tuber)
library(quanteda)
library(quanteda.textstats)
library(wordcloud)
```

We also need two libraries that are only available from *GitHub*. You can install them using the `install_github()` function from the `remotes` package.

```r
library(remotes)
install_github("dill/emoGG")
install_github("hadley/emo")
library(emoGG)
library(emo)
```

Note: Emil Hvitfeldt has created the `emoji` package which is based on the `emo` package and also available via *CRAN*.

Get the Data

As in the last session, we will be working with the - now processed and cleaned - comments for the Emoji Movie Trailer. In case you have collected and saved the comments before, you can just load them at this point.

```
FormattedComments <- readRDS("./data/ParsedEmojiComments.rds")
```

Note: Depending on where you saved the data, how you named the file, and what your current working directory is, you might have to adjust the file path.

Repetition: Collecting Data

If you have not collected and parsed the comments before, you, of course, need to do that before you can analyse any data.

NB: To save time and your *YouTube* API quota limit you might not want to do this now.

Step 1: Collecting the comments

```
Comments <- get_all_comments(video_id="r8pJt4dK_s4") # takes a while
```

Repetition: Parsing the Comments

To run the following code the script `yt_parse.R` as well as the ones containing the helper functions (`CamelCase.R`, `ExtractEmoji.R`, and `ReplaceEmoji.R`) need to be in the working directory (you can find those files in the `scripts` folder in the workshop materials).

```r
source("yt_parse.R")
FormattedComments <- yt_parse(Comments) # this will take a while
```

Note: As an alternative to sourcing the `yt_parse.R` file you could also "manually" run the code from the slides for the session on *Processing and Cleaning User Comments* on the collected comments.

Comments Over Time: Data Wrangling 🤠

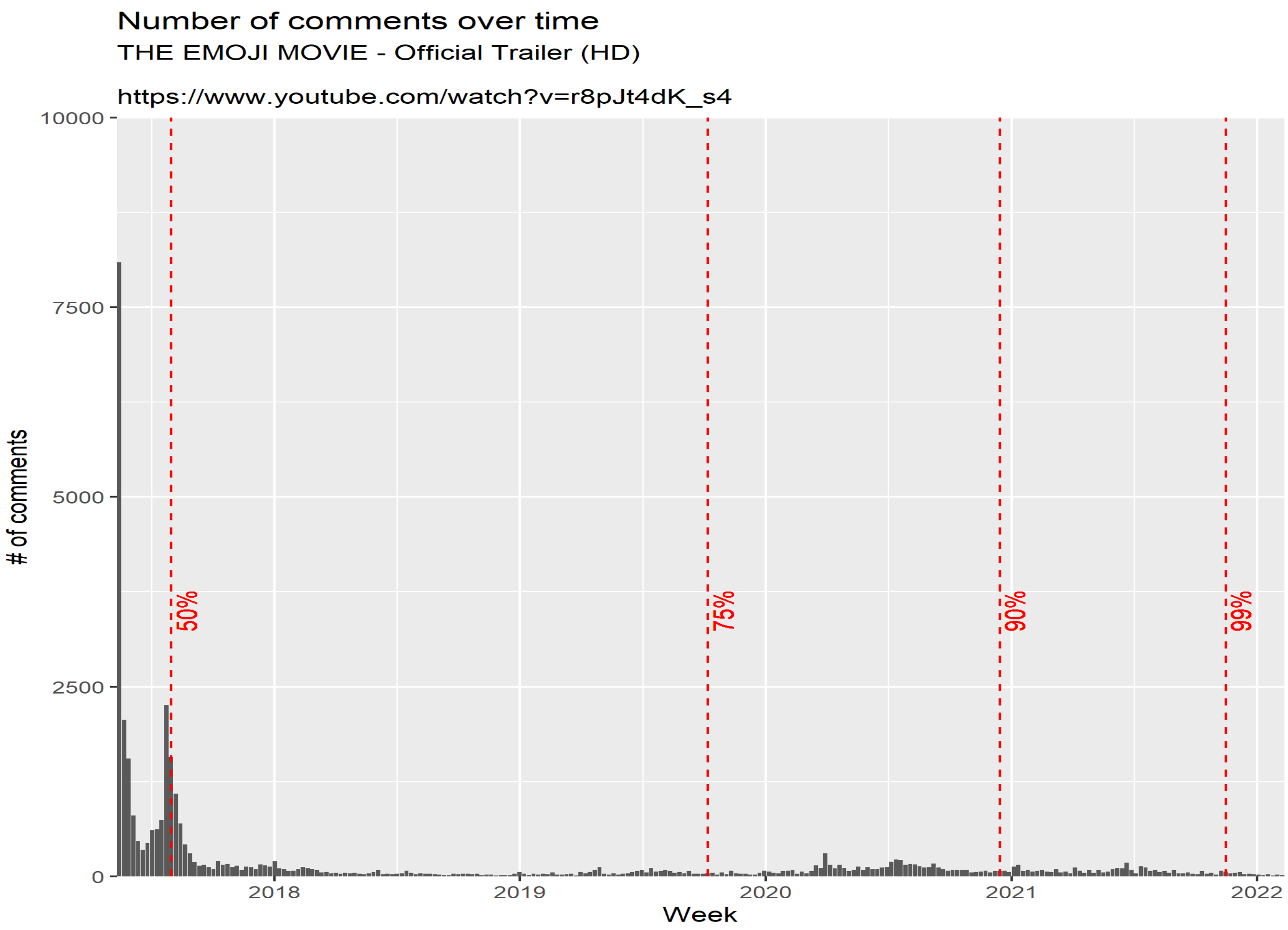
For a first exploratory plot, we want to plot the development of the number of comments per week over time and show until when 50%, 75%, 90%, and 99% of the comments had been posted. This requires some data wrangling.

```
FormattedComments <- FormattedComments %>%
  arrange(Published) %>%
  mutate(date = date(Published),
         week = floor_date(date,
                           unit = "week",
                           week_start = getOption("lubridate.week.sta
         counter = 1)

weekly_comments <- FormattedComments %>%
  count(week) %>%
  mutate(cumulative_count = cumsum(n))

PercTimes <- round(quantile(cumsum(FormattedComments$counter),
                            probs = c(0.5, 0.75, 0.9, 0.99)))
```

Comments Over Time: Plot

```
weekly_comments %>%
  ggplot(aes(x = week, y = n)) +
  geom_bar(stat = "identity") +
  scale_x_date(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,10000)) +
  labs(title = "Number of comments over time",
       subtitle = "THE EMOJI MOVIE - Official Trailer (HD)
       \nhttps://www.youtube.com/watch?v=r8pJt4dK_s4",
       x = "Week",
       y = "# of comments") +
  geom_vline(xintercept = FormattedComments$week[PercTimes],linetype
  geom_text(aes(x = FormattedComments$week[PercTimes][1], label = "50
            colour="red", angle=90, vjust = 1.2) +
  geom_text(aes(x = FormattedComments$week[PercTimes][2], label = "75
            colour="red", angle=90, vjust = 1.2) +
  geom_text(aes(x = FormattedComments$week[PercTimes][3], label = "90
            colour="red", angle=90, vjust = 1.2) +
  geom_text(aes(x = FormattedComments$week[PercTimes][4], label = "99
            colour="red", angle=90, vjust = 1.2)
```

Number of Comments Over Time: Plot



Most Popular Comments

Which comments received the highest number of likes?

```
FormattedComments %>%
  arrange(-LikeCount) %>%
  head(10) %>%
  select(Text, LikeCount, Published)
```

Most Popular Comments

Which comments received the highest number of likes?

```
##                                                                                                          Text
## 1                                                                 Will they show Snapchat nudes in the movie?
## 2                                                                                 The Meme Movie: Coming 2020
## 3                                             Lmao the egg plant emoji never gets used? Do your research lmao
## 4                                                        The book is so much better because it doesn’t exist.
## 5                                           This movie reeks of board room meetings on what kids find "cool".
## 6                                               I believe everyone intentionally looked this up to dislike it
## 7                                                                   The eggplant emoji never used? Suuuuuree.
## 8  So, this thing is still a thing? Ugh, I can't really still believe that you cancelled that Popeye movie...
## 9                                                                                  This is the best part 2:38
## 10                                                                   They cancelled the popeye movie for this
##    LikeCount           Published
## 1       4344 2017-05-16 15:38:40
## 2       3190 2017-10-16 04:08:12
## 3       2969 2017-05-16 23:55:38
## 4       2024 2020-10-30 15:08:17
## 5       1597 2017-05-16 22:40:13
## 6       1543 2020-12-23 18:32:29
## 7       1413 2017-05-17 03:10:34
## 8       1295 2017-05-16 15:32:41
## 9        990 2020-06-08 18:29:03
## 10       808 2020-09-29 14:18:44
```

Text Mining

An introduction to text mining and analysis (for the social sciences) is beyond the scope of this workshop, but there are many great introductions available (for free) online. For example:

- Text Mining with R - A Tidy Approach by Julia Silge & David Robinson: A tidy(verse) approach
- Tutorials for the package `quanteda`
- Text mining for humanists and social scientists in R by Andreas Niekler & Gregor Wiedemann
- Text Mining in R by Jan Kirenz
- Computational Text Analysis by Theresa Gessler
- Automated Content Analysis by Chung-hong Chan (*note*: currently work in progress)

In the following, we will very briefly introduce some key terms and steps in text mining, and then go through some examples of exploring *YouTube* comments (text + emojis).

Popular Text Mining Packages

- tm: the first comprehensive text mining package for R
- tidytext: tidyverse tools & tidy data principles
- **quanteda**: very powerful text mining package with extensive documentation

Text as Data (in a 🌰)

Document = collection of text strings

Corpus = collection of documents (+ metadata about the documents)

Token = part of a text that is a meaningful unit of analysis (often individual words)

Vocabulary = list of all distinct words form a corpus (i.e., all types)

Document-term matrix (DTM) or **Document-feature matrix (DFM)** = matrix with n = # of documents rows and m = size of vocabulary columns where each cell contains the count of a particular word for a particular document

Preprocessing (in a 🌰)

For our examples in this session, we will go through the following preprocessing steps:

1. **Basic string operations**:

   - Transforming to lower case
   - Detecting and removing certain patterns in strings (e.g., punctuation, numbers or URLs)

2. **Tokenization**: Splitting up strings into words (could also be combinations of multiple words: n-grams)

3. **Stopword removal**: Stopwords are very frequent words that appear in almost all texts (e.g., "a","but","it", "the") but have low informational value for most analyses (at least in the social sciences)

NB: There are many other preprocessing options that we will not use for our examples, such as stemming, lemmatization or natural language processing pipelines (e.g., to detect and select specific word types, such as nouns and adjectives). Keep in mind that the choice and order of these preprocessing steps is important and should be informed by your research question.

Tokenization

Before we tokenize the comments, we want to remove newline commands from the strings.

```
FormattedComments <- FormattedComments %>%
  mutate(TextEmojiDeleted = str_replace_all(TextEmojiDeleted,
                                            pattern = "\\\n",
                                            replacement = " "))
```

Tokenization

Now we can tokenize the comments and remove punctuation, symbols, numbers, and URLs.

```
toks <- FormattedComments %>%
  pull(TextEmojiDeleted) %>%
  char_tolower() %>%
  tokens(remove_numbers = TRUE,
               remove_punct = TRUE,
               remove_separators = TRUE,
               remove_symbols = TRUE,
               split_hyphens = TRUE,
               remove_url = TRUE)
```

Document-Feature Matrix

With the tokens we can create a document-feature matrix (DFM) and remove stopwords.

```
commentsDfm <- dfm(toks,
                   remove = quanteda::stopwords("english"))
```

```
## Warning: 'remove' is deprecated; use dfm_remove() instead
```

Most Frequent Words

```
TermFreq <- textstat_frequency(commentsDfm)
head(TermFreq, n = 20)
```

```
##      feature frequency rank docfreq group
## 1      movie     11701    1    8910   all
## 2      emoji      3159    2    2746   all
## 3       like      2819    3    2447   all
## 4       just      2489    4    2215   all
## 5        nom      2239    5       1   all
## 6     people      1546    6    1332   all
## 7       sony      1530    7    1417   all
## 8        bad      1407    8    1293   all
## 9       good      1327    9    1222   all
## 10       one      1221   10    1105   all
## 11      hate      1127   11    1031   all
## 12    emojis      1103   12     993   all
## 13       see      1050   13     930   all
## 14     watch      1042   14     959   all
## 15      make      1025   15     920   all
## 16     think       990   16     904   all
## 17      know       960   17     880   all
## 18    popeye       959   18     883   all
## 19  dislikes       912   19     891   all
## 20       can       888   20     778   all
```

Removing Tokens

We may want to remove additional words (that are not included in the stopwords list) if we consider them irrelevant for our analyses.

```
custom_stopwords <- c("nom", "just", "one")
commentsDfm <- dfm(toks, remove = c(quanteda::stopwords("english"),
                                    custom_stopwords))
```

```
## Warning: 'remove' is deprecated; use dfm_remove() instead
```

```
TermFreq <- textstat_frequency(commentsDfm)
```

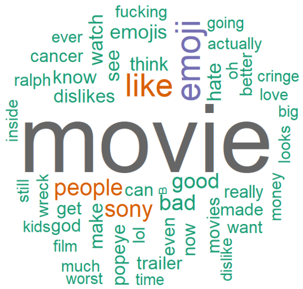
For more options for selecting or removing tokens, see the quanteda documentation.

Wordclouds

```r
wordcloud(words = TermFreq$feature,
          freq = TermFreq$frequency,
          min.freq = 10,
          max.words = 50,
          random.order = FALSE,
          rot.per = 0.35,
          colors = brewer.pal(8, "Dark2"))
```

Note: You can adjust what is plotted by, e.g., changing the minimum frequency (`min.freq`) or the maximum # of words (`max.words`). Check `?wordcloud` for more customization options.

Wordclouds


fucking
ever
watch
emojis
emoji
going
cancer
see
think
actually
ralph
know
like
hate
oh
better
cringe
dislikes
love
big
inside
movie
looks
still
wreck
people
can
B
good
really
money
get
sony
bad
made
kids
god
make
movies
want
film
popeye
lol
even
now
dislike
much
trailer
worst
time

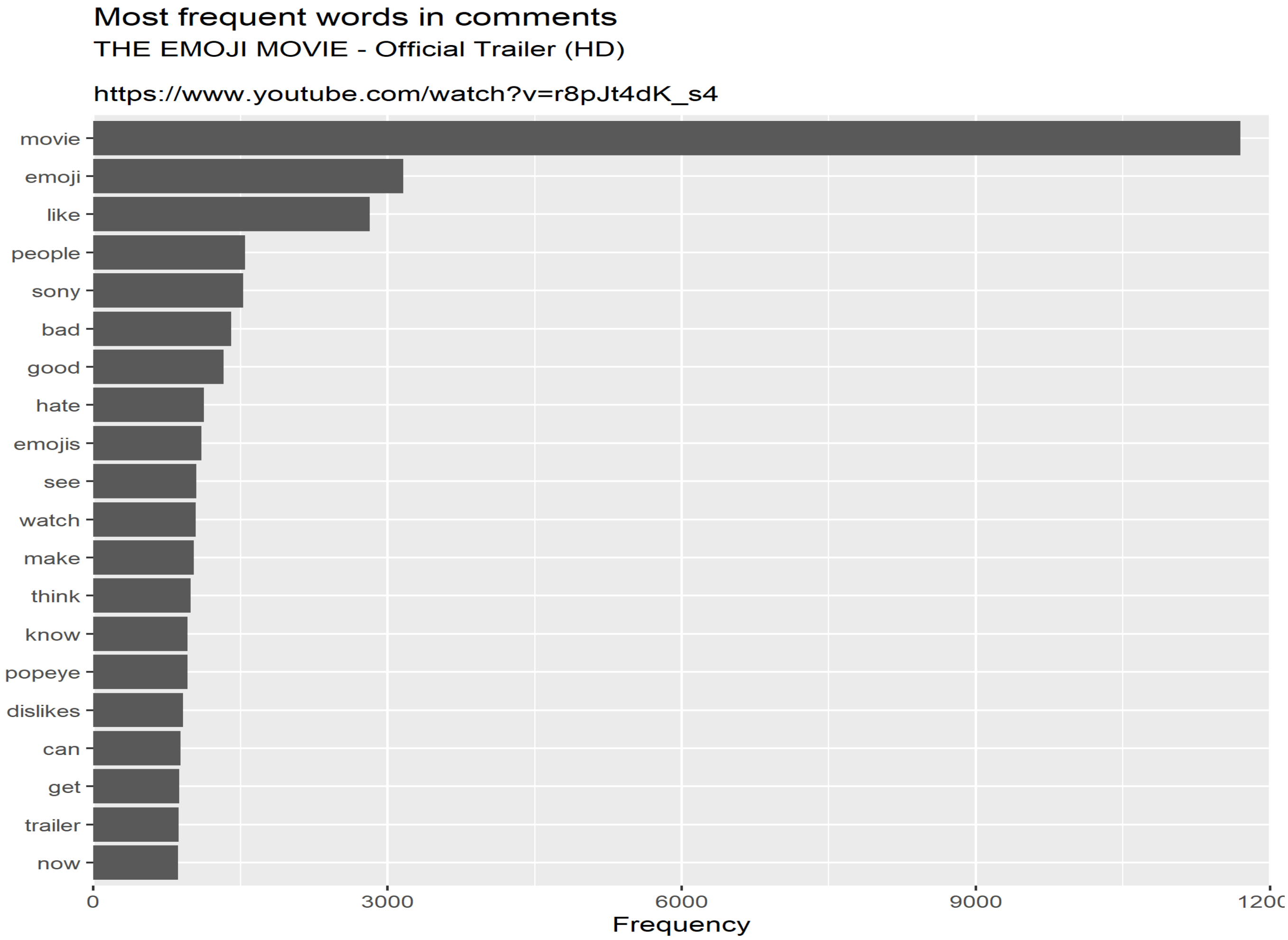
Don't Let Your Words Cloud Your Plots!

Note: To catch up with the state-of-the-art in text analysis, you can check out the blog post "Top 5 most important textual analysis methods papers of the year 2020" by Chung-hong Chan.

Plot Most Frequent Words

```
TermFreq %>%
head(n = 20) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_bar(stat="identity") +
  labs(title = "Most frequent words in comments",
       subtitle = "THE EMOJI MOVIE - Official Trailer (HD)
       \nhttps://www.youtube.com/watch?v=r8pJt4dK_s4",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,12000)) +
  coord_flip()
```

Plot Most Frequent Words

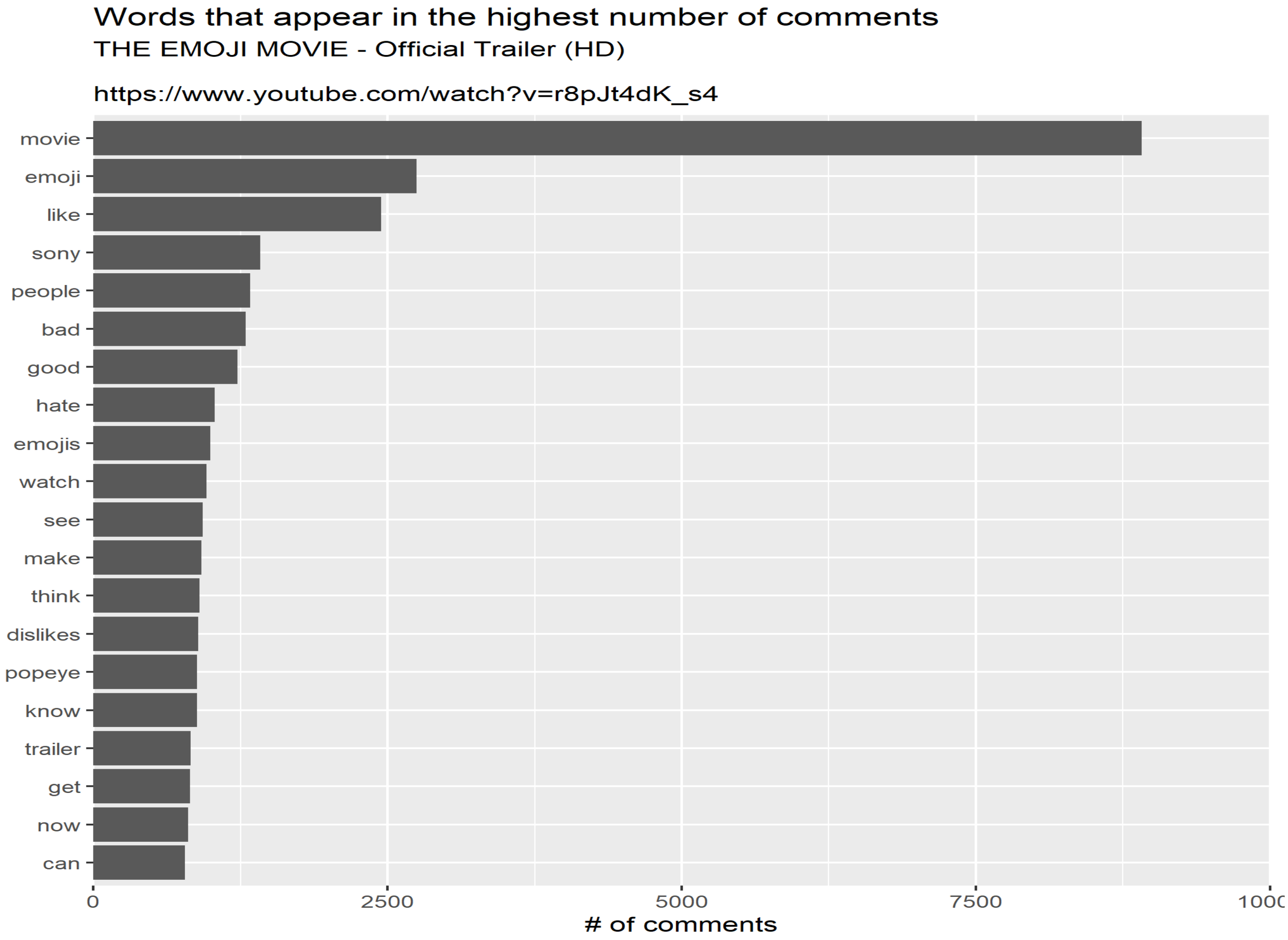


Plot Docfreq

Instead of the raw frequency of words we can also look at the number of comments that a particular word appears in. This metric takes into account that words might be used multiple times in the same comment.

```
TermFreq %>%
head(n = 20) %>%
  ggplot(aes(x = reorder(feature, docfreq), y = docfreq)) +
  geom_bar(stat="identity") +
  labs(title = "Words that appear in the highest number of comments",
       subtitle = "THE EMOJI MOVIE - Official Trailer (HD)
       \nhttps://www.youtube.com/watch?v=r8pJt4dK_s4",
       x = "",
       y = "# of comments") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,10000)) +
  coord_flip()
```

Plot Docfreq



Emojis

In most of the research studying user-generated text from social media, emojis have, so far, been largely ignored. However, emojis convey emotions and meaning, and can, thus, provide additional information or context when working with textual data.

In the following, we will do some exploratory analysis of emoji frequencies in *YouTube* comments. Before we can start, we first need to do some data cleaning again, then tokenize the emojis as some comments include more than one emoji, and create an emoji DFM.

```r
emoji_toks <- FormattedComments %>%
  mutate(Emoji = na_if(Emoji, "NA")) %>% # define missings
  mutate (Emoji = str_trim(Emoji)) %>% # remove spaces
  filter(!is.na(Emoji)) %>% # only keep comments with emojis
  pull(Emoji) %>% # pull out column cotaining emoji labels
  tokens(what = "fastestword") # tokenize emoji labels

EmojiDfm <- dfm(emoji_toks) # create DFM for emojis
```

Most Frequent Emojis

```
EmojiFreq <- textstat_frequency(EmojiDfm)
head(EmojiFreq, n = 10)
```

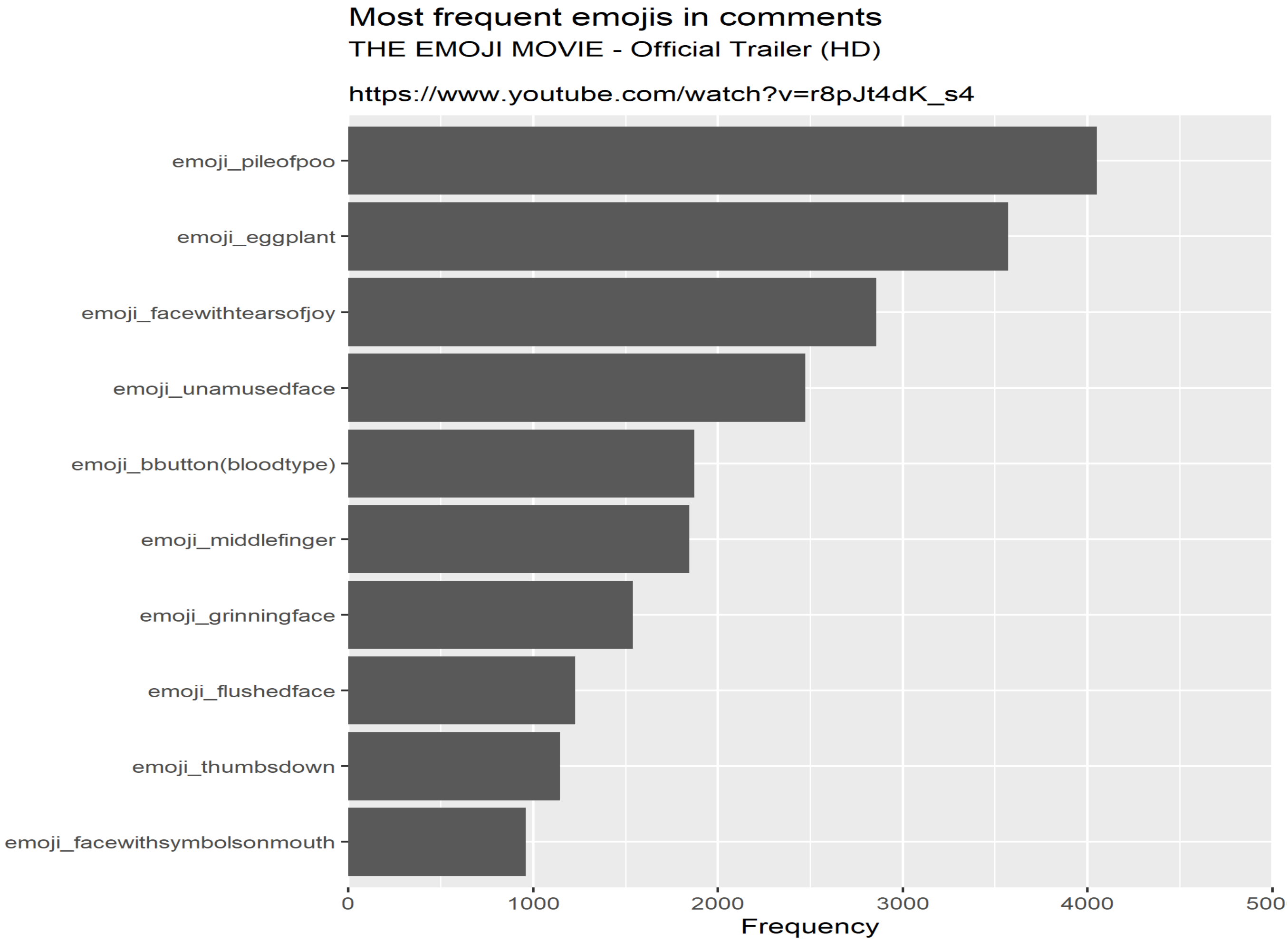
```
##                         feature frequency rank docfreq group
## 1               emoji_pileofpoo      4050    1     531   all
## 2                emoji_eggplant      3571    2     272   all
## 3      emoji_facewithtearsofjoy      2856    3     839   all
## 4            emoji_unamusedface      2473    4     664   all
## 5      emoji_bbutton(bloodtype)      1873    5     129   all
## 6            emoji_middlefinger      1845    6     298   all
## 7            emoji_grinningface      1541    7     362   all
## 8             emoji_flushedface      1226    8     245   all
## 9              emoji_thumbsdown      1145    9     261   all
## 10 emoji_facewithsymbolsonmouth       960   10      89   all
```

Plot Most Frequent Emojis

```
EmojiFreq %>%
head(n = 10) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_bar(stat="identity") +
  labs(title = "Most frequent emojis in comments",
       subtitle = "THE EMOJI MOVIE - Official Trailer (HD)
       \nhttps://www.youtube.com/watch?v=r8pJt4dK_s4",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,5000)) +
 coord_flip()
```

Note: Similar to what we did for the comment text before we could replace `frequency` with `docfreq` in the above code to create a plot with the emojis that appear in the highest number of comments.

Plot Most Frequent Emojis



😎 Emoji Frequency Plot: Preparation (1)

The previous emoji frequency plot was a bit 😪. To make things prettier, we can use the actual emojis instead of the text labels in our plot. Doing this takes a bit of preparation...[1]

As a first step, we need an emoji lookup table in which the values in the name column have the same format as the labels in the feature column of our `EmojiFreq` object.

```
emoji_lookup <- jis %>%
  select(runes, name) %>%
  mutate(runes = str_to_lower(runes),
         name = str_to_lower(name)) %>%
  mutate(name = str_replace_all(name, " ", "")) %>%
  mutate(name = paste0("emoji_", name))
```

[1] For an alternative approach to using emojis in `ggplot2` see this blog post by Emil Hvitfeldt.

😎 Emoji Frequency Plot: Preparation (2)

The second step of preparation for the nicer emoji frequency plot is creating mappings of emojis to data points so that we can use emojis instead of points in a scatter plot.[1]

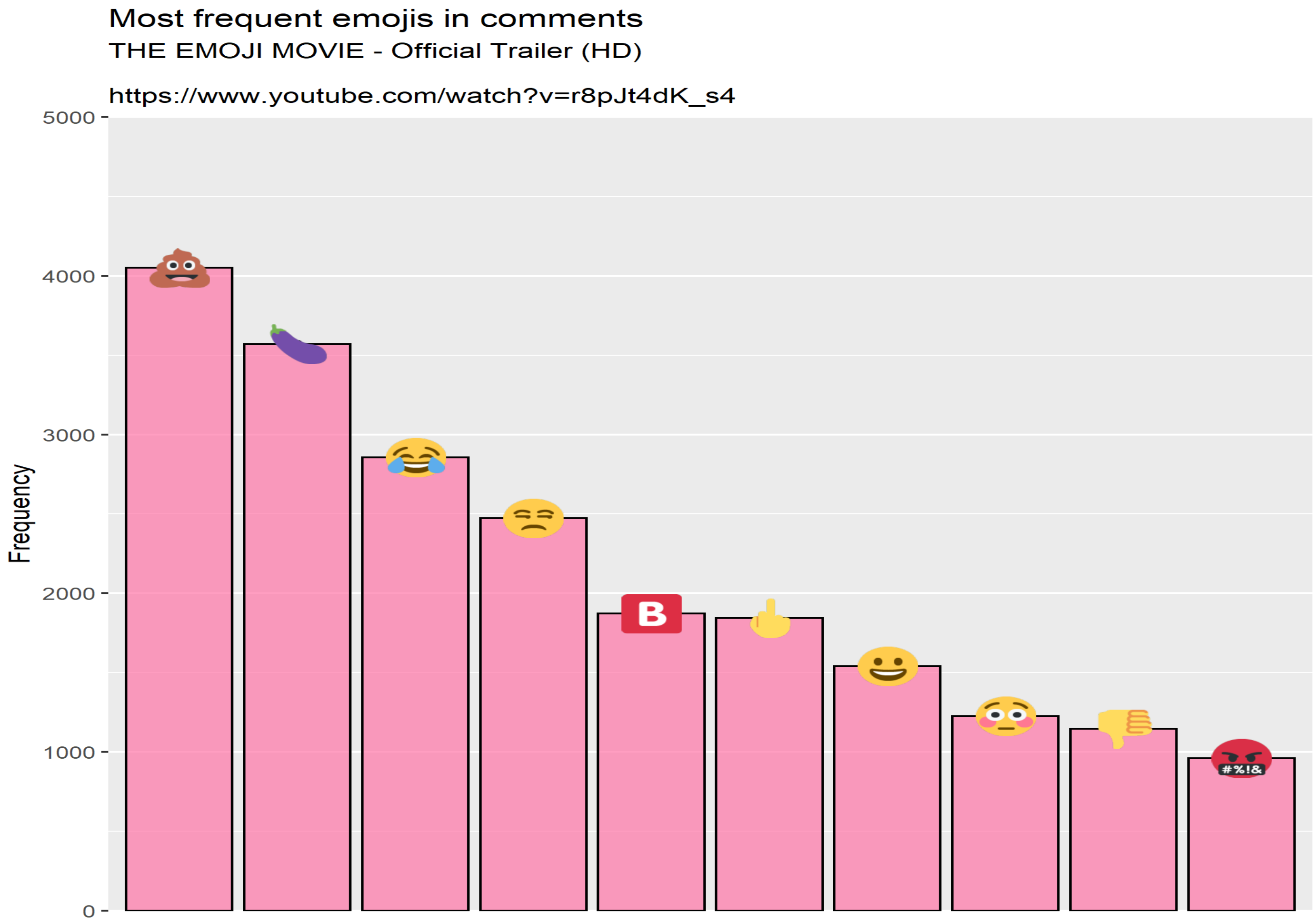
```
top_emojis <- 1:10

for(i in top_emojis){
  name <- paste0("mapping", i)
  assign(name,
         do.call(geom_emoji,list(data = EmojiFreq[i,],
                                 emoji = gsub("^0{2}","", strsplit(tc
}
```

[1] Please note that this code has not been tested systematically. We only used it with a few videos. Depending on which emojis are the most frequent for the video you look at, this might not work because (a) one of the emojis is not included in the emoji lookup table (which uses the `jis` data frame from the emo package) or (b) the content in the `runes` column does not match the format/code that the `emoji` argument in the `geom_emoji` function from the emoGG package expects.

😎 Emoji Frequency Plot

```
EmojiFreq %>%
head(n = 10) %>%
  ggplot(aes(x = reorder(feature, -frequency), y = frequency)) +
  geom_bar(stat="identity",
           color = "black",
           fill = "#FF74A6",
           alpha = 0.7) +
  geom_point() +
  labs(title = "Most frequent emojis in comments",
       subtitle = "THE EMOJI MOVIE - Official Trailer (HD)
       \nhttps://www.youtube.com/watch?v=r8pJt4dK_s4",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,5000)) +
  theme(panel.grid.major.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  mapping1 +
  mapping2 +
  mapping3 +
  mapping4 +
  mapping5 +
  mapping6 +
  mapping7 +
  mapping8 +
  mapping9 +
  mapping10
```

😎 Emoji Frequency Plot



Most frequent emojis in comments

THE EMOJI MOVIE - Official Trailer (HD)

https://www.youtube.com/watch?v=r8pJt4dK_s4

Exercise time 🏋️ 💪 🏃 🚴

Solutions