Automatic sampling and Analysis of YouTube Data

Basic Text Analysis of User Comments

Julian Kohne
Johannes Breuer
M. Rohangis Mohseni

2020-02-11

Required Libraries for This Session

```
library(tidyverse)
library(lubridate)
library(tuber)
library(quanteda)
library(devtools)
install_github("dill/emoGG")
install_github("hadley/emo")
```

Collect & Parse the Data

Note: To save time and your *YouTube* API quota limit we suggest that you don't "code along" in this session

```
Comments <- get_all_comments(c(video_id="DcJFdCmN98s")) # takes a wh
source("yt_parse.R") # yt_parse.R needs to be in the working directo
FormattedComments <- yt_parse(Comments) # will take a while
```

Comments Over Time: Data Wrangling

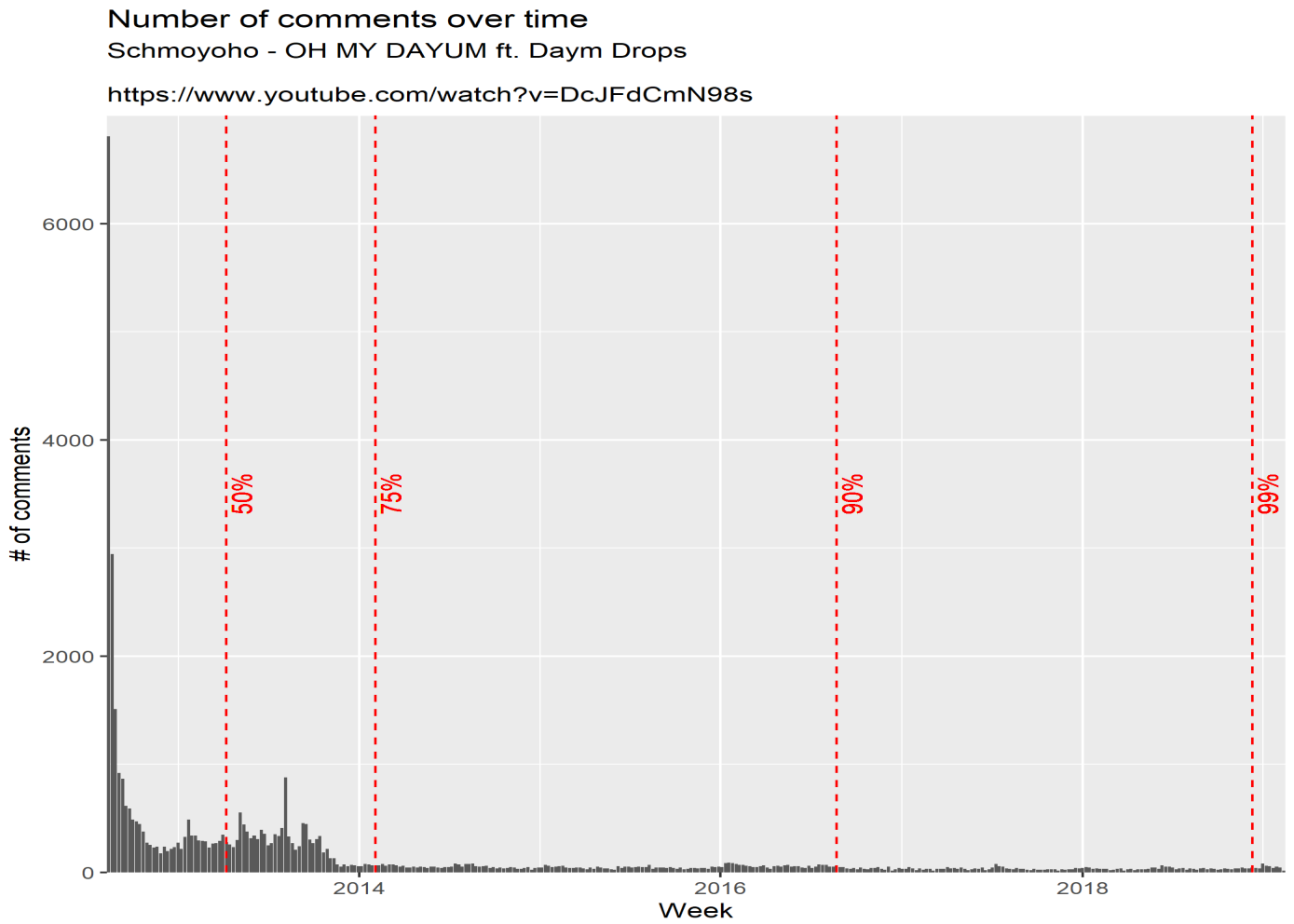
First, we want to plot the development of the number of comments per week over time and show until when 50%, 75%, 90%, and 99% of the comments had been posted. This requires some data wrangling.

```
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 %>%
  group_by(week) %>%
  summarize(count = n()) %>%
  mutate(cumulative_count = cumsum(count))

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 = count)) +
  geom_bar(stat = "identity") +
  scale_x_date(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,7000)) +
  labs(title = "Number of comments over time",
  subtitle = "Schmoyoho - OH MY DAYUM ft. Daym Drops
       \nhttps://www.youtube.com/watch?v=DcJFdCmN98s",
  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



Text Mining

In this session, we will discuss some basic exploratory analyses of *YouTube* user comments. We will explore the use of words as well as the use of emojis.

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

- Text Mining in R 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
- Automatisierte Inhaltsanalyse mit `R` by Cornelius Puschmann

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 strings (+ metadata about the documents)

Corpus = collection of 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

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. **Stop word removal**: Stopwords are very frequent words that appear in almost all texts (e.g., "a","but","it", "the")

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 = " "))
```

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,
                remove_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"))
```

Most Frequent Words

```
TermFreq <- textstat_frequency(commentsDfm)
TermFreq[10:20, ]
```

```
##    feature frequency rank docfreq group
## 10    just      1949   10    1849   all
## 11      oh      1796   11    1333   all
## 12     now      1769   12    1730   all
## 13     get      1756   13    1695   all
## 14    good      1660   14    1609   all
## 15    best      1646   15    1623   all
## 16     one      1390   16    1340   all
## 17       d      1171   17    1123   all
## 18     guy      1166   18    1129   all
## 19      xd      1153   19    1128   all
## 20    damn      1137   20     821   all
```

Removing Tokens

There seem to be ASCII emojis among the most frequent tokens. We might want to remove these and/or other tokens if we consider them irrelevant for our analyses.

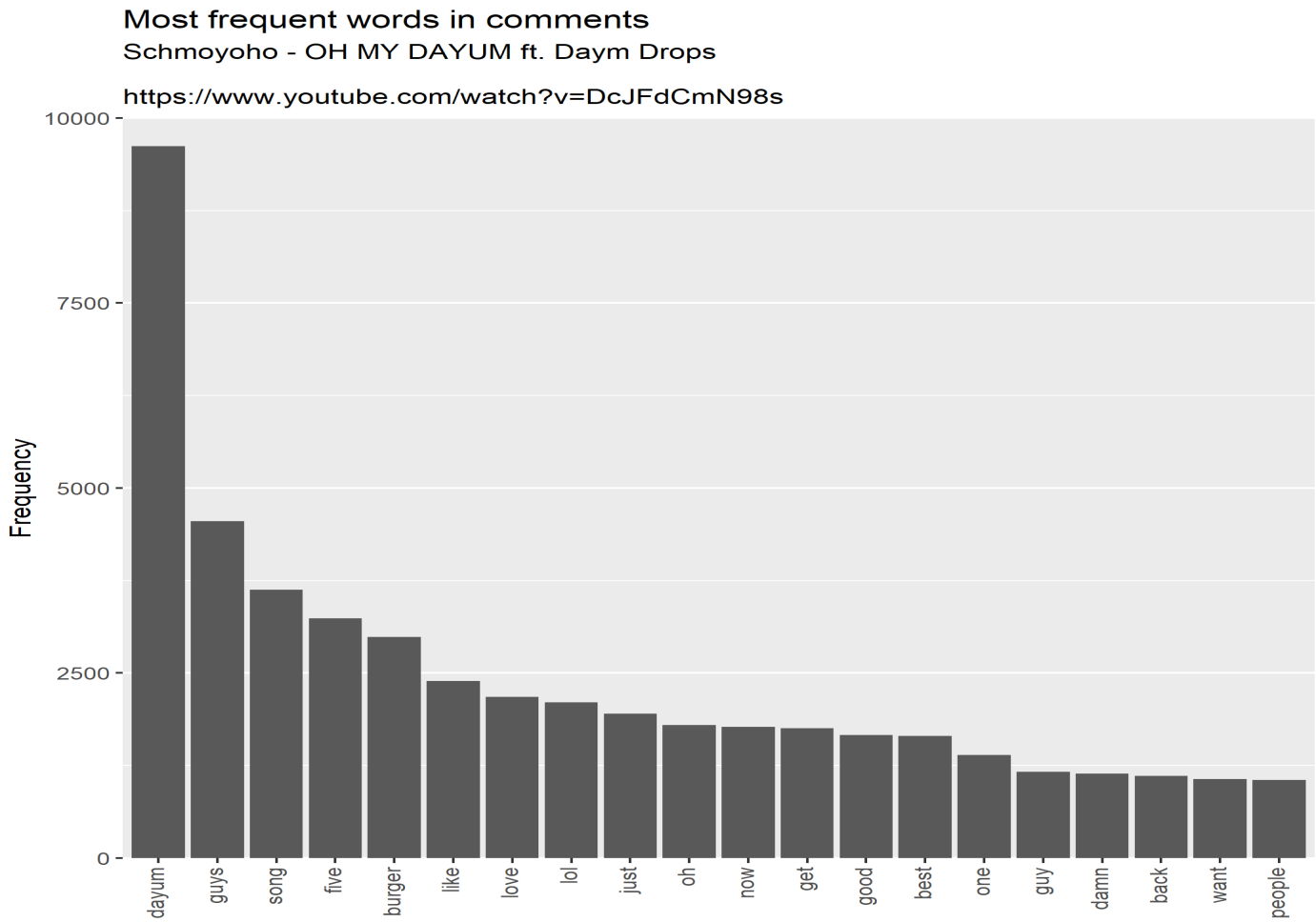
```
custom_stopwords <- c("video","d","xd", "youtube")
commentsDfm <- dfm(toks, remove = c(quanteda::stopwords("english"),
                                    custom_stopwords))
TermFreq <- textstat_frequency(commentsDfm)
```

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

Plot Most Frequent Words (1)

```
head(TermFreq, n = 20) %>%
ggplot(aes(x = reorder(feature, -frequency), y = frequency)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90,
                                   hjust = 1,
                                   vjust = 0.3)) +
  labs(title = "Most frequent words in comments",
       subtitle = "Schmoyoho - OH MY DAYUM ft. Daym Drops
       \nhttps://www.youtube.com/watch?v=DcJFdCmN98s",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,10000)) +
  theme(panel.grid.major.x = element_blank())
```

Plot Most Frequent Words (2)

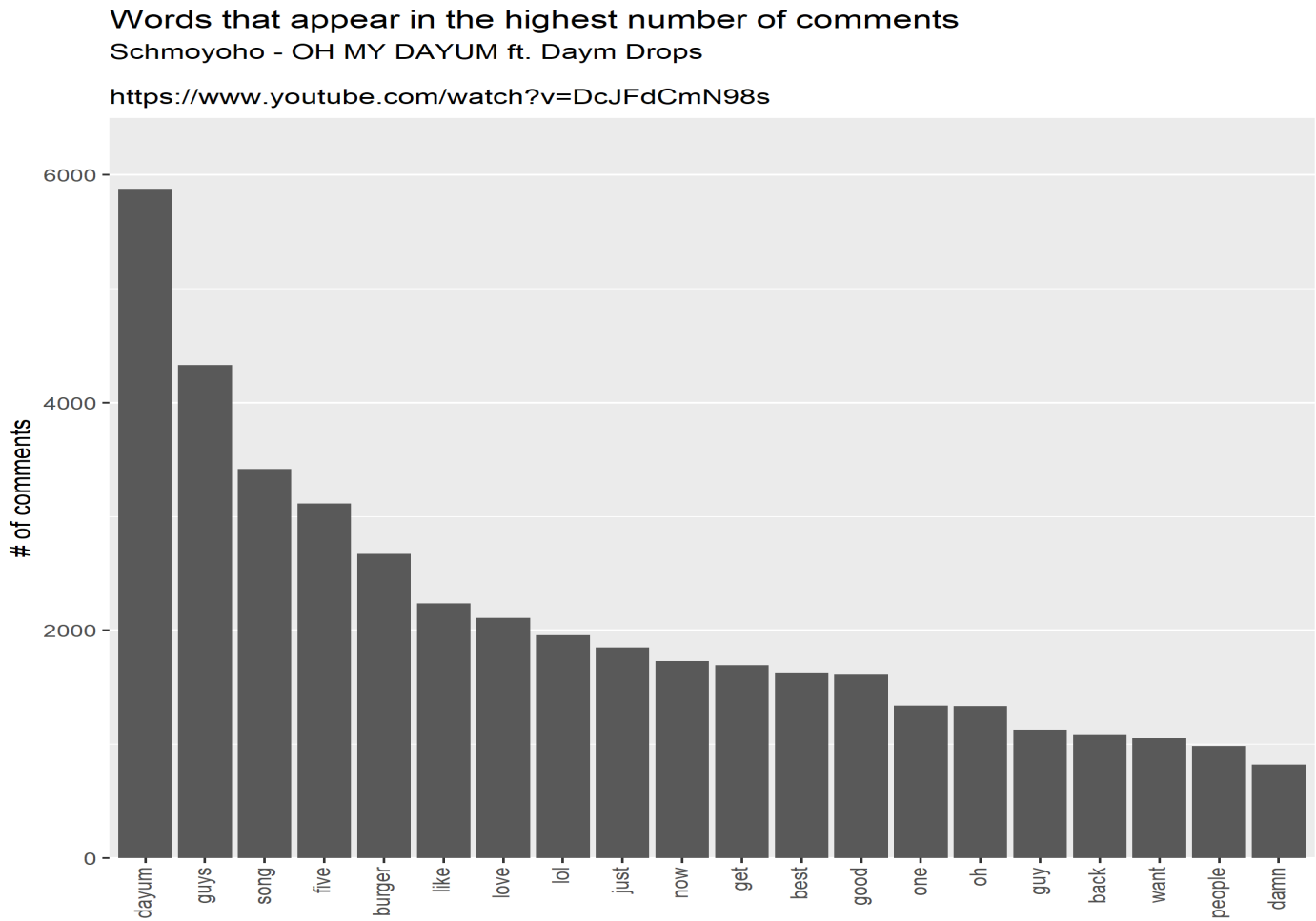


Plot Docfreq (1)

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.

```
head(TermFreq, n = 20) %>%
ggplot(aes(x = reorder(feature, -docfreq), y = docfreq)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90,
                                   hjust = 1,
                                   vjust = 0.3)) +
  labs(title = "Words that appear in the highest number of comments",
       subtitle = "Schmoyoho - OH MY DAYUM ft. Daym Drops
       \nhttps://www.youtube.com/watch?v=DcJFdCmN98s",
       x = "",
       y = "# of comments") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,6500)) +
  theme(panel.grid.major.x = element_blank())
```

Plot Docfreq (2)



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.

```
emoji_toks <- FormattedComments %>%
  mutate_at(c("Emoji"), list(~na_if(., "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() # tokenize emoji labels

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

Most Frequent Emojis

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

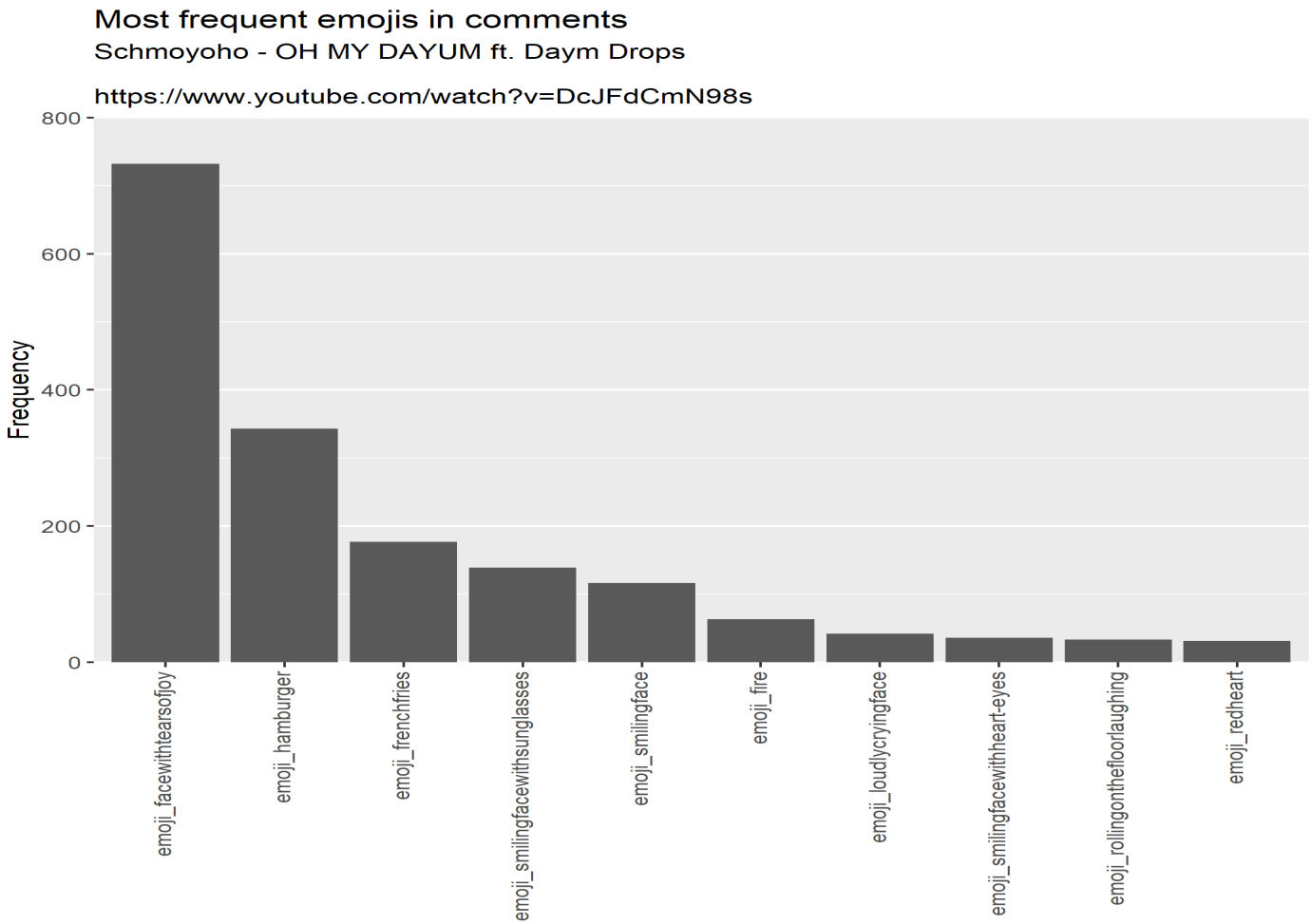
```
##                          feature frequency rank docfreq group
## 1         emoji_facewithtearsofjoy       732    1     261   all
## 2                emoji_hamburger       343    2      44   all
## 3              emoji_frenchfries       177    3      19   all
## 4  emoji_smilingfacewithsunglasses       139    4      11   all
## 5              emoji_smilingface       116    5       7   all
## 6                     emoji_fire        63    6      20   all
## 7          emoji_loudlycryingface        42    7      24   all
## 8  emoji_smilingfacewithheart-eyes        36    8      16   all
## 9  emoji_rollingonthefloorlaughing        33    9      19   all
## 10                emoji_redheart        31   10      22   all
```

Plot Most Frequent Emojis (1)

```
head(EmojiFreq, n = 10) %>%
ggplot(aes(x = reorder(feature, -frequency), y = frequency)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90,
                                   hjust = 1,
                                   vjust = 0.3)) +
  labs(title = "Most frequent emojis in comments",
       subtitle = "Schmoyoho - OH MY DAYUM ft. Daym Drops
       \nhttps://www.youtube.com/watch?v=DcJFdCmN98s",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,800)) +
  theme(panel.grid.major.x = element_blank())
```

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 (2)



😎 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]

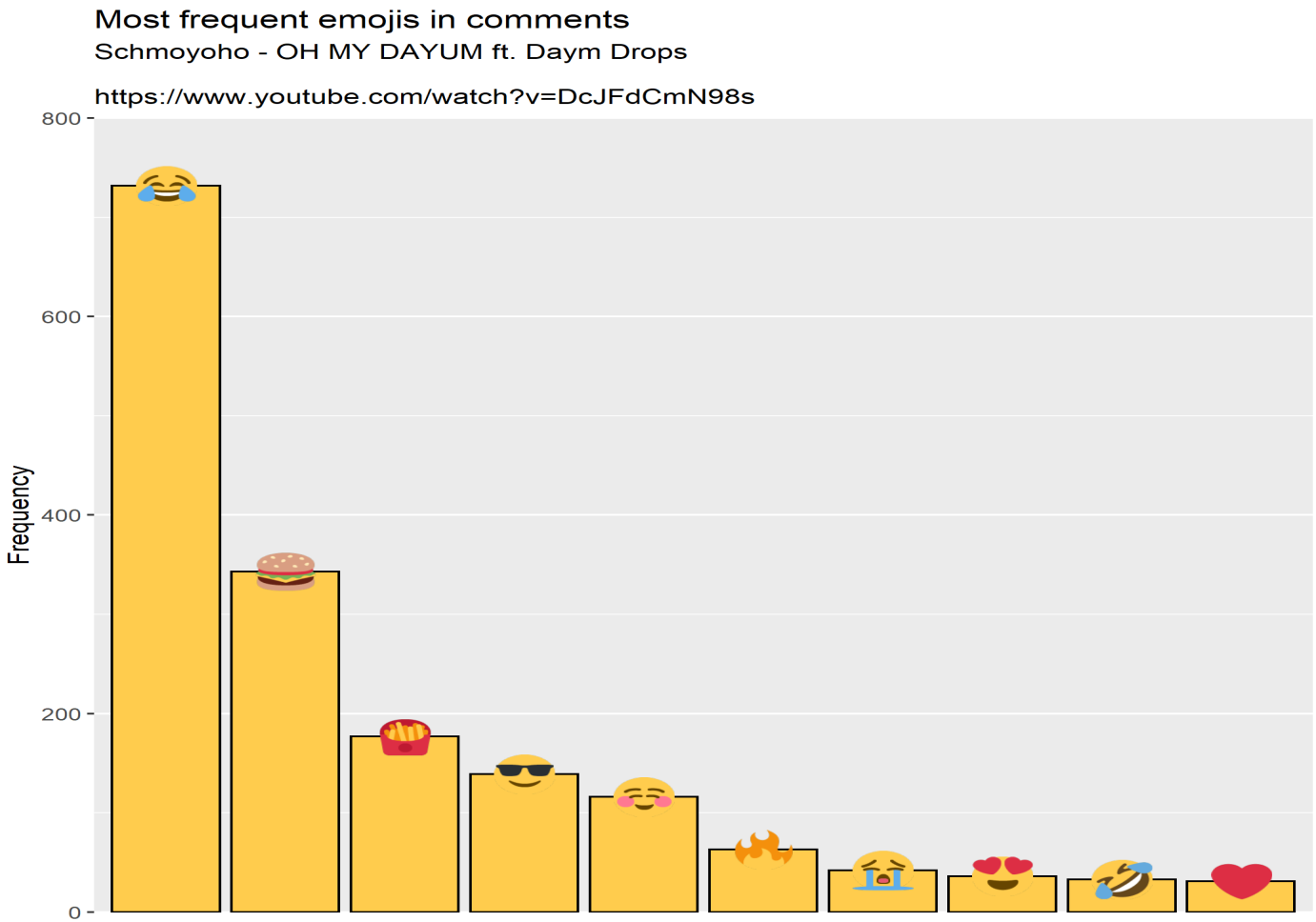
```
top_emojis <- 1:10

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 (1)

```
head(EmojiFreq, n = 10) %>%
ggplot(aes(x = reorder(feature, -frequency), y = frequency)) +
  geom_bar(stat="identity",
           color = "black",
           fill = "#FFCC4D") +
  geom_point() +
  labs(title = "Most frequent emojis in comments",
       subtitle = "Schmoyoho - OH MY DAYUM ft. Daym Drops
       \nhttps://www.youtube.com/watch?v=DcJFdCmN98s",
       x = "",
       y = "Frequency") +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0,800)) +
  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 (2)



Exercise time □♀□□□

Solutions