TAD24

Decoding Review Impact: A Study on Sentiment, Length, and Readability in Movie Reviews - Suryam Gupta

Introduction

Online consumer reviews have become an integral part of modern purchasing decisions, with platforms like Amazon and IMDb hosting millions of reviews for movies and TV shows. These reviews often feature helpfulness voting mechanisms, enabling users to signal the usefulness of a review to others. Despite their prevalence, the factors that determine a review's perceived helpfulness remain under explored. Understanding these factors is crucial, not only for optimizing review visibility and consumer decision-making but also for empowering content creators to craft impactful reviews.

The perceived helpfulness of a review is influenced by various textual, structural, and emotional characteristics. Features such as length, sentiment, and readability have been hypothesized to play significant roles. For instance, while longer reviews may provide more comprehensive information, excessively lengthy reviews could overwhelm readers. Similarly, sentiment polarity can evoke engagement, as readers may resonate more with strongly positive or negative reviews than with neutral ones. Readability, which reflects the clarity and accessibility of a review, is also expected to influence user engagement.

This study builds on existing research to examine how these factors interact to shape the helpfulness ratings of reviews. By focusing on the movie and TV show domain, it seeks to uncover unique patterns in consumer engagement within the entertainment industry.

Background and Literature Review

Previous studies have examined various aspects of online review helpfulness. Mudambi and Schuff (2010) found that review extremity and depth significantly influenced helpfulness votes in product reviews. Their findings suggested that while extreme sentiments were generally preferred, the relationship between review length and helpfulness was more complex, often displaying a curvilinear trend. Yin et al. (2016) explored the role of sentiment analysis and textual features in predicting helpfulness ratings, confirming that emotional intensity often correlated with higher perceived helpfulness.

The role of readability has been highlighted in several studies as well. Ghose and Ipeirotis (2011) demonstrated that reviews written at an optimal readability level tended to receive more engagement, as they struck a balance between accessibility and informativeness. Similarly, Baek et al. (2012) emphasized that content quality, including coherence and clarity, influenced how users perceived review helpfulness.

While these studies offer valuable insights, they primarily focus on general product reviews, leaving a gap in the understanding of review dynamics specific to the entertainment sector. This research aims to bridge that gap by examining reviews for movies and TV shows, where emotional and subjective elements play a more prominent role.

Importance of the Research Topic

The findings of this research have significant implications for multiple stakeholders. For online platforms, understanding what drives helpfulness ratings can inform the development of algorithms that prioritize reviews most likely to benefit users. For content creators, this knowledge can guide the crafting of more effective reviews, enhancing their influence within the community. Moreover, marketers and advertisers can gain insights into consumer sentiment and behavior, enabling them to better tailor their strategies to target audiences.

Furthermore, as reviews play a pivotal role in shaping consumer choices, this study contributes to the broader understanding of digital consumer behavior. By shedding light on the nuanced interactions between textual features and user engagement, it advances the field of computational social science and natural language processing, offering a foundation for future research in online communication and digital media.

Research Question

What factors influence the likelihood of receiving a higher overall voting/perceived helpfulness in movie and TV show reviews?

Hypothesis

1. The readability of a review has a curvilinear relationship with perceived helpfuless, i.e. reviews with less readability score have less vote count. But as this score increases, the vote count increases as well. However, after a certain threshold value, the increase in readability score results in a decrease in vote count.

2. The length of a review has a curvilinear relationship with perceived helpfuless, i.e. shorter length reviews have less vote count, but as the review length increases, the vote count increases as well. However, after a certain threshold value, the increase in review length results in a decrease in vote count.

3. The two extreme sentiments of a review, i.e. strongly positive and strongly negative, correspond to higher vote count, compared to neutral sentiment reviews.

Data Source

The dataset used for this project is the Amazon Review/Product Dataset (https://nijianmo.github.io/amazon/index.html), provided by Julian McAuley and Jianmo Ni, University of California, San Diego (UCSD). It includes data reviews for the range May 1996 - October 2018. It has a total number of 233.1 million reviews of several different categories. In this project, I will be working on the “Movies and TV shows” category.

Reading the data

```
#install.packages("jsonlite")
library(jsonlite)
```

```
# all_data <- list() # empty list to store the data
#
# file_path <- "D:/UMass/1st sem/DACSS 758 TAD/TAD24/Tutorials/Final Project/Movies_a
#
# con <- file(file_path, open = "r")  # this is to open a connection
#
# line_count <- 0
#
# while (length(line <- readLines(con, n = 1, warn = FALSE)) > 0) { # reading file li
#   line_count <- line_count + 1
#   all_data[[line_count]] <- fromJSON(line) # parsing JSON line and appending to the
#   if (line_count >= 50000) {
#     break
#   }
# }
#
# close(con)
#
# all_data_df <- do.call(rbind, all_data)
#
# head(all_data_df)
```

```
# I am having some trouble accessing columns from the above "all_data_df" and I haven
#
# import os
# import json
# import gzip
# import pandas as pd
# import numpy as np
#
# all_data = []
#
# with open(r"D:\UMass\1st sem\DACSS 758 TAD\TAD24\Tutorials\Final Project\Movies_and
#     for i, line in enumerate(f):
#         if i >= 50000:
#             break
#         all_data.append(json.loads(line))
#
# df = pd.DataFrame.from_dict(all_data)
# df = df.dropna(subset=['vote'])
# df.reset_index(inplace=True, drop=True)
# df['vote'] = pd.to_numeric(df['vote'], errors='coerce')
# df.to_csv(r"D:\UMass\1st sem\DACSS 758 TAD\TAD24\Tutorials\Final Project\moviesandt
```

The above python pre-processed data frame is loaded in R then:

```
library(readr)
df <- read_csv("D:\\UMass\\1st sem\\DACSS 758 TAD\\TAD24\\Tutorials\\Final Project\\m
               show_col_types = FALSE)
```

```
New names:
• `` -> `...1`
```

```
sum(is.na(df$vote)) # this returns 1, vote column surprisingly still had a NA value i
```

```
[1] 1
```

```
df <- subset(df, !is.na(vote))
```

```
head(df)
```

```
# A tibble: 6 × 13
   ...1 overall verified reviewTime  reviewerID     asin      style reviewerName
  <dbl>   <dbl> <lgl>    <chr>       <chr>          <chr>     <chr> <chr>
1     0       5 TRUE     04 21, 2005 A3JVF9Y53BEOGC 00050386… {'Fo… Anthony Tho…
2     1       5 TRUE     04 6, 2005  A12VPEOEZS1KTC 00050386… {'Fo… JadeRain
3     2       5 TRUE     12 3, 2010  ATLZNVLYKP9AZ  00050386… {'Fo… T. Fisher
4     3       5 TRUE     11 8, 2005  A2LUL6PRTXE7SE 00050386… {'Fo… PSM/Bokor
5     4       5 TRUE     11 23, 2008 A3TS9EQCNLU0SM 00050926… {'Fo… Amazon Cust…
6     5       2 TRUE     05 2, 2014  A38KRRY00H5TEY 00050926… {'Fo… Trish
# ℹ 5 more variables: reviewText <chr>, summary <chr>, unixReviewTime <dbl>,
#   vote <dbl>, image <chr>
```

```
# removing the first index column
df <- df[,-1]
```

```
dim(df)
```

```
[1] 10654    12
```

The dataset has 10,654 rows and 12 columns (I loaded only the first 50000 rows from the original JSON file, and after removing rows where the “vote” column (this kind of acts as the target variable) has NAs, I am left with around 10,000 rows, which I think is quite a good amount for this project).

```
summary(df)
```

```
    overall       verified        reviewTime         reviewerID
 Min.   :1.000   Mode :logical   Length:10654       Length:10654
 1st Qu.:3.000   FALSE:7315      Class :character   Class :character
 Median :5.000   TRUE :3339      Mode  :character   Mode  :character
 Mean   :4.023
 3rd Qu.:5.000
 Max.   :5.000
     asin              style           reviewerName        reviewText
 Length:10654       Length:10654       Length:10654       Length:10654
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
   summary          unixReviewTime           vote            image          
 Length:10654       Min.   :9.019e+08   Min.   :  2.000   Length:10654      
 Class :character   1st Qu.:1.051e+09   1st Qu.:  2.000   Class :character  
 Mode  :character   Median :1.147e+09   Median :  3.000   Mode  :character  
                    Mean   :1.179e+09   Mean   :  8.018                     
                    3rd Qu.:1.308e+09   3rd Qu.:  6.000                     
                    Max.   :1.524e+09   Max.   :745.000                     
```

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
glimpse(df)
```

```
Rows: 10,654
Columns: 12
$ overall        <dbl> 5, 5, 5, 5, 5, 2, 4, 4, 5, 5, 5, 3, 4, 4, 5, 3, 5, 5, 4…
$ verified       <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T…
$ reviewTime     <chr> "04 21, 2005", "04 6, 2005", "12 3, 2010", "11 8, 2005"…
$ reviewerID     <chr> "A3JVF9Y53BEOGC", "A12VPEOEZS1KTC", "ATLZNVLYKP9AZ", "A…
$ asin           <chr> "000503860X", "000503860X", "000503860X", "000503860X",…
$ style          <chr> "{'Format:': ' DVD'}", "{'Format:': ' DVD'}", "{'Format…
$ reviewerName   <chr> "Anthony Thompson", "JadeRain", "T. Fisher", "PSM/Bokor…
$ reviewText     <chr> "I have seen X live many times, both in the early days …
$ summary        <chr> "A great document of a great band", "YES!!  X LIVE!!", …
$ unixReviewTime <dbl> 1114041600, 1112745600, 1291334400, 1131408000, 1227398…
$ vote           <dbl> 11, 5, 5, 17, 7, 2, 2, 2, 6, 4, 3, 31, 62, 2, 2, 2, 2, …
$ image          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
```

```
names(df)
```

```
 [1] "overall"        "verified"       "reviewTime"     "reviewerID"    
 [5] "asin"           "style"          "reviewerName"   "reviewText"    
 [9] "summary"        "unixReviewTime" "vote"           "image"         
```

Removing escape characters (), which are pretty common when working with textual data and reviews and do not contribute to any relevant information :

```
reviews_with_esc_chr <- df[grepl("\n", df$reviewText), ]
nrow(reviews_with_esc_chr)
```

```
[1] 6056
```

We can see how prevalent these characters are.

```
df$reviewText <- gsub("\n", "", df$reviewText)
```

Calculating length of reviews:

```
df$review_length <- sapply(strsplit(df$reviewText, "\\s+"), length)
```

```
summary(df$review_length)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    1.0    86.0   171.0   240.1   315.0  5069.0 
```

Reviews with very short review_length do not give much information for us (or NLP models) to analyze and interpret results properly, and create discrepancies later in the project when dealing with sentiment analysis or word clouds. For example,

```
df$reviewText[df$review_length==1]
```

```
 [1] "Excellent"   "WOW!"        "good"        "good"        "good"       
 [6] "excellent"   "depressing"  "brutal"      "Excellent"   "Good!!"     
[11] "great"       "Fantastic"   "terrible."   "Great"       "Excellent"  
[16] "OK"          "Masterpiece"
```

```
df$reviewText[df$review_length==2]
```

```
 [1] "great movie."         "good product!"        "As advertised."      
 [4] "Very satisfied"       "enough said"          "thanks you"          
 [7] "not biblical"         "luv it"               "Great movie"         
[10] "Always entertaining." "Didn't like"          "Great gift"          
[13] "GREAT MOVIE!"         "Its ok"
```

Hence, I will be removing such rows.

Additionally, we have a few reviews which are very long compared to most of the reviews in the dataset, essentially acting as an outliers. I will set the threshold of 1024 and remove all reviews with length longer than 1024 as later in the project I plan to use pre-trained models, which usually have the capacity to only handle 1024 tokens. If I do not do this, other longer reviews can get truncated and lose contextual information, leading to inaccurate and unreliable results.

```
nrow(df[df$review_length < 10 | df$review_length > 1024,])
```

```
[1] 235
```

```
nrow(df)
```

```
[1] 10654
```

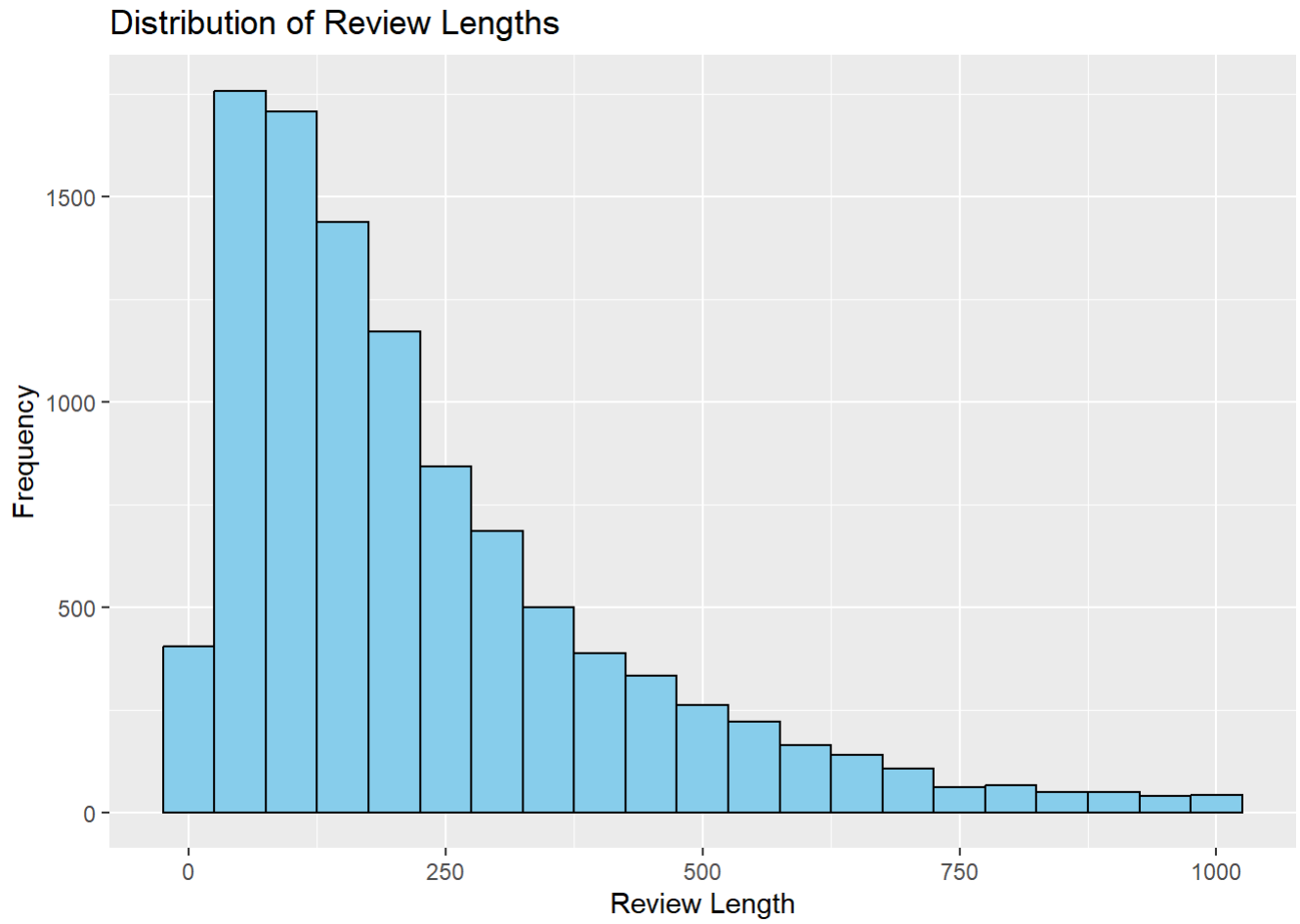
Finally, we see that only 235 rows in total, out 10,654, have length less than 10 or more than 1024, so we are anyway not losing on a lot of data and only keeping quality data which will give us more relevant information.

```
df <- df[!(df$review_length < 10 | df$review_length > 1024), ]

# Reset row index
rownames(df) <- NULL
```

```
summary(df$review_length)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   10.0    88.0   171.0   228.9   310.0  1021.0 
```

```
library(ggplot2)
ggplot(df, aes(x = review_length)) +
  geom_histogram(binwidth = 50, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Review Lengths", x = "Review Length", y = "Frequency"
```




```
dim(df)
```

```
[1] 10419    13
```

Creating corpus and summary

```
library(devtools)
```

```
Loading required package: usethis
```

```
library(tidytext)
library(plyr)
```

```
------------------------------------------------------------------------------

You have loaded plyr after dplyr - this is likely to cause problems.
If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
library(plyr); library(dplyr)

------------------------------------------------------------------------------


Attaching package: 'plyr'

The following objects are masked from 'package:dplyr':

    arrange, count, desc, failwith, id, mutate, rename, summarise,
    summarize
```

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ lubridate 1.9.3     ✓ tibble    3.2.1
✓ purrr     1.0.2     ✓ tidyr     1.3.1

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ plyr::arrange()   masks dplyr::arrange()
✖ purrr::compact()  masks plyr::compact()
✖ plyr::count()     masks dplyr::count()
✖ plyr::desc()      masks dplyr::desc()
✖ plyr::failwith()  masks dplyr::failwith()
✖ dplyr::filter()   masks stats::filter()
✖ purrr::flatten()  masks jsonlite::flatten()
✖ plyr::id()        masks dplyr::id()
✖ dplyr::lag()      masks stats::lag()
✖ plyr::mutate()    masks dplyr::mutate()
✖ plyr::rename()    masks dplyr::rename()
✖ plyr::summarise() masks dplyr::summarise()
✖ plyr::summarize() masks dplyr::summarize()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
library(quanteda)
```

```
Package version: 4.1.0
Unicode version: 15.1
ICU version: 74.1
Parallel computing: 8 of 8 threads used.
See https://quanteda.io for tutorials and examples.
```

```
library(quanteda.textplots)
movie_corpus <- corpus(df$reviewText)
movie_corpus
```

```
Corpus consisting of 10,419 documents.
text1 :
"I have seen X live many times, both in the early days and th..."

text2 :
"I was so excited for this!  Finally, a live concert video fr..."

text3 :
"X is one of the best punk bands ever. I don't even like call..."

text4 :
"I must admit I was hesitant to purchase this DVD. A classic ..."

text5 :
"Watch this and learn how we ended up in the state we are in ..."

text6 :
"While there was a lot of information in this piece (some of ..."

[ reached max_ndoc ... 10,413 more documents ]
```

```
movie_summary <- summary(movie_corpus)
head(movie_summary)
```

```
   Text Types Tokens Sentences
1 text1    48     56         4
2 text2   123    201        13
3 text3   246    501        16
4 text4    90    129         9
5 text5    17     19         1
6 text6    78    108         3
```

Tokenization and pre-processing

```
# Tokenize with removal of punctuation and numbers
movie_tokens <- tokens(movie_corpus, remove_punct = TRUE, remove_numbers = TRUE)

# removing more symbols (as remove_punct didn't remove them and they appear in the wo
movie_tokens <- tokens_remove(movie_tokens, pattern = c(">", "<", "="))

# Remove English stopwords
```

```
movie_tokens <- tokens_remove(movie_tokens, pattern = stopwords("english"))

# Convert to lowercase
movie_tokens <- tokens_tolower(movie_tokens)

movie_tokens[1]
```

```
Tokens consisting of 1 document.
text1 :
 [1] "seen"    "x"       "live"    "many"    "times"   "early"   "days"
 [8] "recent"  "reunion" "shows"   "trust"   "say"
[ ... and 14 more ]
```

```
#tokens_select(movie_tokens, pattern = ">", valuetype = "fixed") # to check presence
#any(sapply(as.list(movie_tokens), function(doc) any(grepl("<", doc))))
```

Lemmatization

I apply lemmatization instead of stemming primarily because the other techniques I am going to apply in the future would only work if the words make sense. For example, for sentiment analysis, I will probably use a dictionary based method and if I have the word “happi” instead of “happy,” sentiment analysis simply wouldn’t work as expected.

```
lemm_tokens <- tokens_replace(movie_tokens,
                              pattern = lexicon:: hash_lemmas$token,
                              replacement = lexicon:: hash_lemmas$lemma)
lemm_tokens
```

```
Tokens consisting of 10,419 documents.
text1 :
 [1] "see"     "x"       "live"    "many"    "time"    "early"   "day"
 [8] "recent"  "reunion" "show"    "trust"   "say"
[ ... and 14 more ]

text2 :
 [1] "excite"  "finally" "live"    "concert" "video"   "x"       "see"
 [8] "twice"   "time"    "tony"    "g"       "guitar"
[ ... and 71 more ]

text3 :
 [1] "x"    "one"  "good" "punk" "band" "ever" "even" "like" "call" "punk"
[11] "band" "call"
[ ... and 222 more ]

text4 :
 [1] "must"     "admit"    "hesitant" "purchase" "dvd"      "classic"
 [7] "punk"     "band"     "perform"  "twenty"   "plus"     "year"
[ ... and 58 more ]
```

```
text5 :
[1] "watch"   "learn"   "end"     "state"   "dying"   "country"

text6 :
 [1] "lot"         "information" "piece"       "come"        "fast"
 [6] "freight"     "train"       "another"     "side"        "video"
[11] "doubt"       "produce"
[ ... and 30 more ]

[ reached max_ndoc ... 10,413 more documents ]
```

Creating DFM and Feature Co-occurrence matrix

```
lemm_dfm <- dfm(lemm_tokens)
lemm_dfm
```

```
Document-feature matrix of: 10,419 documents, 62,273 features (99.85% sparse) and 0 docvars.
       features
docs    see x live many time early day recent reunion show
  text1   1 1    2    1    1     1   1      1       1    2
  text2   4 4    1    0    2     0   0      0       0    0
  text3   0 9    2    0    1     0   0      0       0    0
  text4   0 2    1    0    0     0   0      0       0    0
  text5   0 0    0    0    0     0   0      0       0    0
  text6   0 0    0    0    0     0   0      0       0    0
[ reached max_ndoc ... 10,413 more documents, reached max_nfeat ... 62,263 more features ]
```

```
topfeatures(lemm_dfm, 20)
```

```
   movie      film      good       one       see      make      like       get
   20073     19481     15319     11203      8403      8224      7272      6585
    time      just     great character     story     watch        go     scene
    6441      5997      5635      4858      4840      4749      4741      4392
     can      life     think      even
    4311      4225      4160      4138
```

```
smaller_dfm <- dfm_trim(lemm_dfm, min_termfreq = 5)
smaller_dfm <- dfm_trim(smaller_dfm, min_docfreq = .2, docfreq_type = "prop")
smaller_dfm
```

```
Document-feature matrix of: 10,419 documents, 40 features (68.87% sparse) and 0 docvars.
       features
docs    see many time say dvd come way know one make
  text1   1    1    1   1   1    1   0    0   0    0
  text2   4    0    2   0   3    1   1    1   1    1
  text3   0    0    1   1   2    0   2    2   6    0
  text4   0    0    0   0   2    0   0    0   3    0
  text5   0    0    0   0   0    0   0    0   0    0
  text6   0    0    0   0   0    1   0    0   1    0
[ reached max_ndoc ... 10,413 more documents, reached max_nfeat ... 30 more features ]
```

```
# create fcm from dfm
smaller_fcm <- fcm(smaller_dfm)
smaller_fcm
```

```
Feature co-occurrence matrix of: 40 by 40 features.
        features
features  see many time  say  dvd come  way know   one  make
    see  6329 4113 8755 4752 4173 4880 4227 5203 13173 10043
    many    0 1291 3439 1836 1584 1869 1776 2024  5529  4138
    time    0    0 3949 3675 2890 4001 3341 3977 10862  8119
    say     0    0    0 1457 1666 2171 1931 2668  6158  4507
    dvd     0    0    0    0 3168 1959 1505 1623  4896  4138
    come    0    0    0    0    0 1432 2020 2497  6550  4830
    way     0    0    0    0    0    0 1167 2261  5569  4456
    know    0    0    0    0    0    0    0 1979  6985  5251
    one     0    0    0    0    0    0    0    0 10779 14247
    make    0    0    0    0    0    0    0    0     0  6311
[ reached max_feat ... 30 more features, reached max_nfeat ... 30 more features ]
```

```
dim(smaller_fcm)
```

```
[1] 40 40
```

```
myFeatures <- names(sort(colSums(smaller_fcm), decreasing = TRUE)[1:30])
myFeatures
```

```
 [1] "film"      "movie"     "good"      "go"        "character" "scene"
 [7] "life"      "story"     "much"      "look"      "watch"     "people"
[13] "take"      "get"       "end"       "can"       "find"      "love"
[19] "think"     "little"    "great"     "also"      "first"     "one"
[25] "like"      "just"      "make"      "year"      "really"    "even"
```

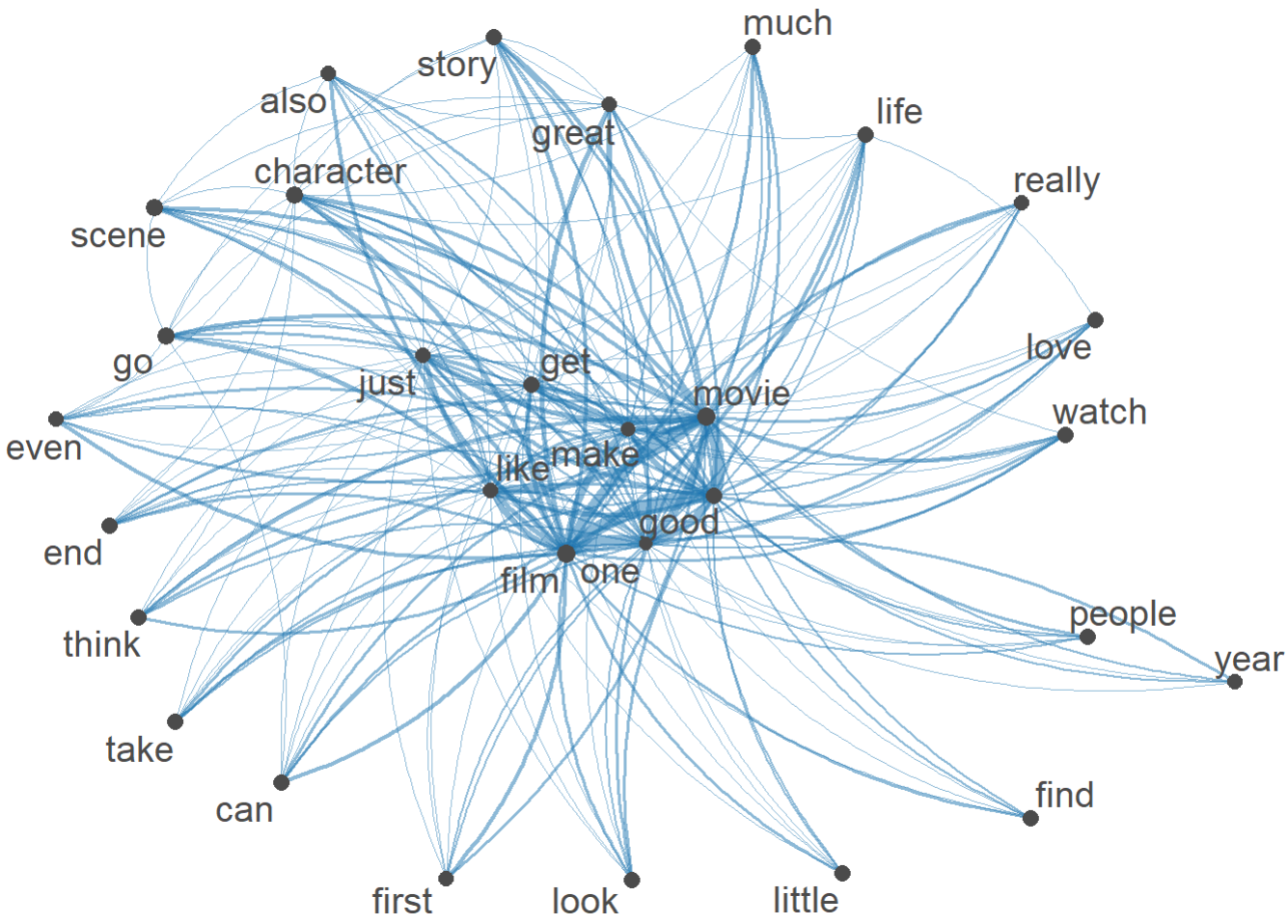
```
# retain only those top features as part of our matrix
even_smaller_fcm <- fcm_select(smaller_fcm, pattern = myFeatures, selection = "keep")
even_smaller_fcm
```

```
Feature co-occurrence matrix of: 30 by 30 features.
        features
features   one  make  like just  good even   get really  also great
  one    10779 14247 11824 9977 24999 7400 11142   6288  7415  8653
  make       0  6311  9550 7877 18771 5620  8235   4974  5657  6168
  like       0     0  5512 7832 16002 4994  7948   4927  4749  5287
  just       0     0     0 3806 13477 4210  6746   4359  3759  4350
  good       0     0     0    0 19379 9390 14931   9137 10319 11554
  even       0     0     0    0     0 1963  4655   2782  2924  3084
  get        0     0     0    0     0    0  4945   4291  4439  5007
  really     0     0     0    0     0    0     0   2062  2612  3062
  also       0     0     0    0     0    0     0      0  2095  3414
  great      0     0     0    0     0    0     0      0     0  3239
[ reached max_feat ... 20 more features, reached max_nfeat ... 20 more features ]
```

```
dim(even_smaller_fcm)
```

```
[1] 30 30
```

```
# compute size weight for vertices in network
size <- log(colSums(even_smaller_fcm))

# create plot
textplot_network(even_smaller_fcm, vertex_size = size/ max(size) * 3)
```



Word Clouds

```
set.seed(0)
textplot_wordcloud(lemm_dfm, min_count = 50, random_order = FALSE)
```

As visible, words like “movie,” “film,” “one,” etc. are prominently featured because they are obvious words in the context of a movie dataset and do not give a lot of information. Hence, making a filtered_dfm just for the purpose of a cleaner word cloud.

```
#install.packages("quanteda.textstats")
library(quanteda.textplots)
library(quanteda.textstats)

word_frequencies <- textstat_frequency(lemm_dfm)
#head(word_frequencies)

# Setting a maximum count threshold
max_count <- 7000

# Filter words based on the maximum count
filtered_dfm <- dfm_select(lemm_dfm,
                           pattern = word_frequencies$feature[word_frequencies$frequ
                           selection = "keep")
#filtered_dfm

# Create the word cloud with the filtered DFM
suppressWarnings({
  set.seed(0)
  textplot_wordcloud(filtered_dfm, min_count = 25, random_order = FALSE)
})
```

Hence, we see that beyond the obvious movie-related terms, words like “watch,” “character,” “story,” “scene,” “dvd,” etc. are very frequent.

Testing first hypothesis - Readability and its association with the vote count

```
movie_corpus
```

```
Corpus consisting of 10,419 documents.
text1 :
"I have seen X live many times, both in the early days and th..."

text2 :
"I was so excited for this!  Finally, a live concert video fr..."

text3 :
"X is one of the best punk bands ever. I don't even like call..."

text4 :
"I must admit I was hesitant to purchase this DVD. A classic ..."

text5 :
"Watch this and learn how we ended up in the state we are in ..."

text6 :
```

```
"While there was a lot of information in this piece (some of ..."

[ reached max_ndoc ... 10,413 more documents ]
```

```
library(quanteda)
library(quanteda.textstats)

# calculating readability scores
readability_scores <- textstat_readability(movie_corpus, measure = c("Flesch.Kincaid"
head(readability_scores)
```

```
  document Flesch.Kincaid
1    text1       3.917000
2    text2       5.170723
3    text3      11.311329
4    text4       7.068690
5    text5       5.196667
6    text6      13.031579
```

```
# combing vote column from df and readability scores in one dataframe
readability_df <- data.frame(vote = df$vote, readability = readability_scores$Flesch)
head(readability_df)
```

```
  vote readability
1   11    3.917000
2    5    5.170723
3    5   11.311329
4   17    7.068690
5    7    5.196667
6    2   13.031579
```

```
str(readability_df)
```

```
'data.frame':   10419 obs. of  2 variables:
 $ vote       : num  11 5 5 17 7 2 2 2 6 4 ...
 $ readability: num  3.92 5.17 11.31 7.07 5.2 ...
```

```
# correlation analysis
correlation <- cor(readability_df$vote, readability_df$readability)
cat("Correlation between readability and vote count:", correlation)
```

```
Correlation between readability and vote count: 0.04187853
```

```
# mean readability in the corpus
round(mean(readability_df$readability), 1)
```

```
[1] 10.9
```

```
median(readability_df$readability)
```

```
[1] 9.733
```

```
c(min(readability_df$readability), max(readability_df$readability))
```

```
[1]  -0.5722727 203.3046304
```

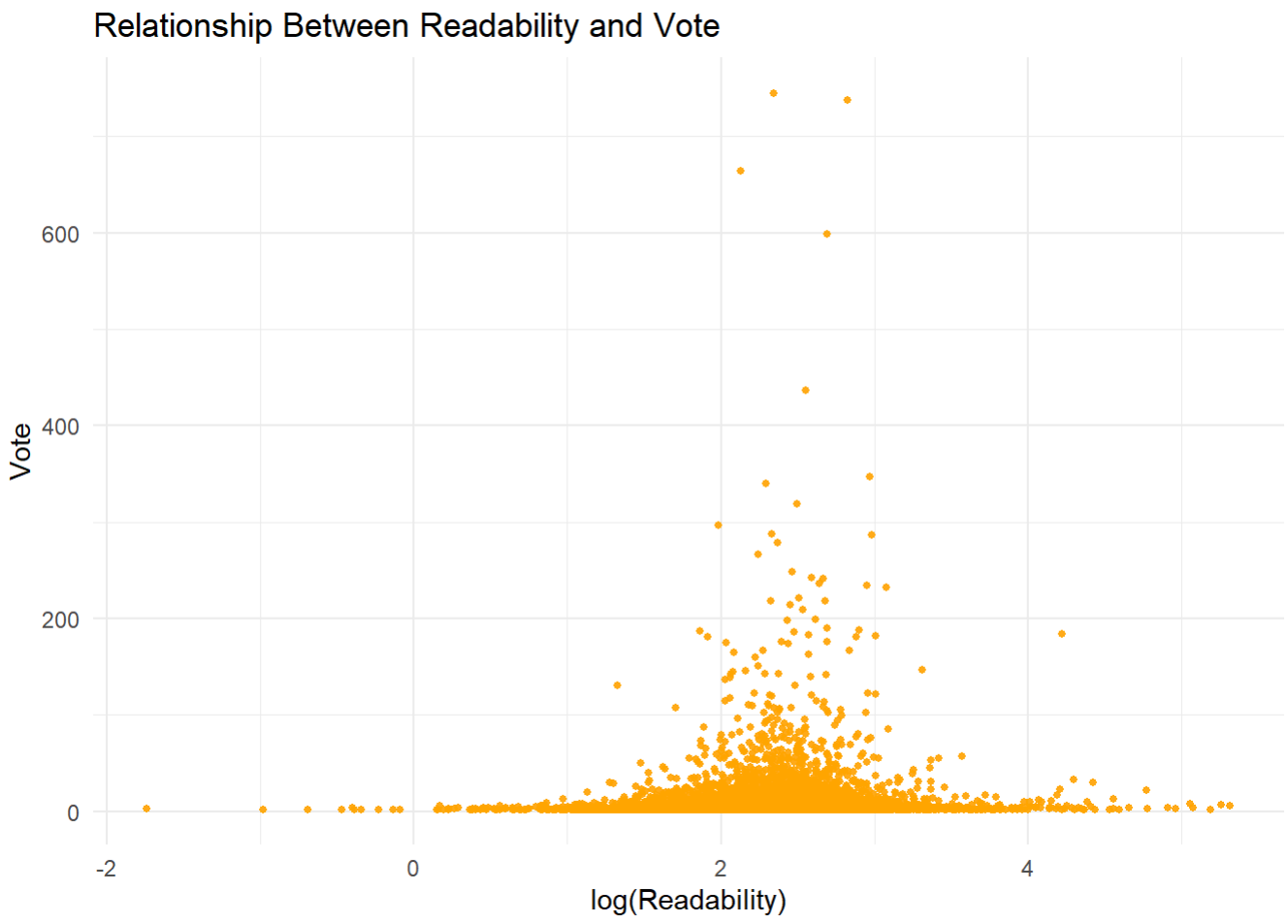
We see that the correlation value between readability scores and vote count is very low, only 0.041. The mean and median values are also at a low of 10.8 and 9.73 respectively, suggesting that these reviews, on an average, are difficult to comprehend. The minimum value is -0.57 and maximum value is 203.30.

From the above graph, we see that very few values are above 100, essentially acting as outliers and disrupting the scale of the plot. There are only 10 such reviews, so we can definitely just temporarily remove them, but even better, we can log the readability column to scale down such big values and then visualize this plot for better interpretability.

```
ggplot(readability_df, aes(x=log(readability), y=vote)) +
  geom_point(color = "orange", size = 1, alpha = 0.9) +
  labs(
    title = "Relationship Between Readability and Vote",
    x = "log(Readability)", y = "Vote") +
  theme_minimal()
```

```
Warning in log(readability): NaNs produced
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_point()`).
```



We notice a **curvilinear** relationship, i.e. it suggests that readability improves the helpfulness of a review to a certain point, but beyond that threshold, further improvements in readability yields diminishing returns or even reduce engagement if the review becomes too simplistic or formulaic.

- **Low readability**: If a review is hard to read due to grammar issues, spelling errors, or poor structure, it is likely to be perceived as less helpful, receiving fewer votes.
- **Moderate readability**: As readability improves, reviews become clearer, more engaging, and easier to understand, leading to more votes.
- **High readability**: After reaching a certain level of clarity, further improvements might not significantly increase helpfulness. If reviews are overly simplified or "too polished," readers may perceive them as less authentic or overly generic, thus reducing engagement.

So, the results support hypothesis and the relationship between readability and helpfulness votes is curvilinear, with the most helpful reviews striking a balance between clarity and depth, but not oversimplifying the content.

```
# saving these scores in original df for later use
df$readability <- readability_df$readability
```

Testing second hypothesis - Review length and its association with the vote count

```
mean(df$review_length)
```

```
[1] 228.9422
```

```
median(df$review_length)
```

```
[1] 171
```

```
c(min(df$review_length), max(df$review_length))
```

```
[1]   10 1021
```

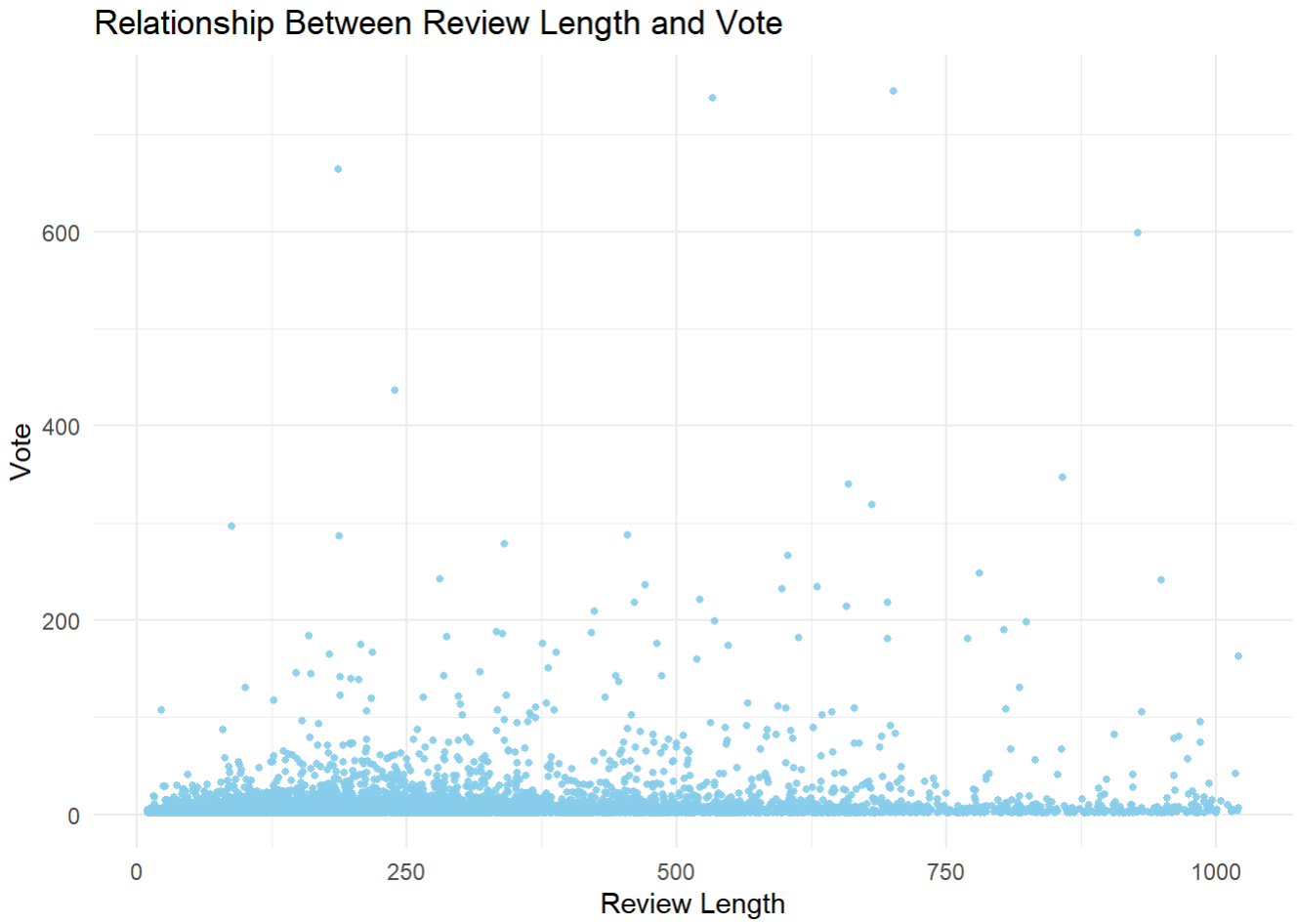
```
# correlation analysis
correlation <- cor(df$vote, df$review_length)
cat("Correlation between review length and vote count:", correlation)
```

```
Correlation between review length and vote count: 0.1721328
```

The mean and median values from overall reviews in the dataset is approximately 229 and 171 words respectively. There is a very weak correlation of 0.17 between review_length and vote count.

```
ggplot(df, aes(x=review_length, y=vote)) +
  geom_point(color = "skyblue", size = 1, alpha = 0.9) +
  labs(
    title = "Relationship Between Review Length and Vote",
```

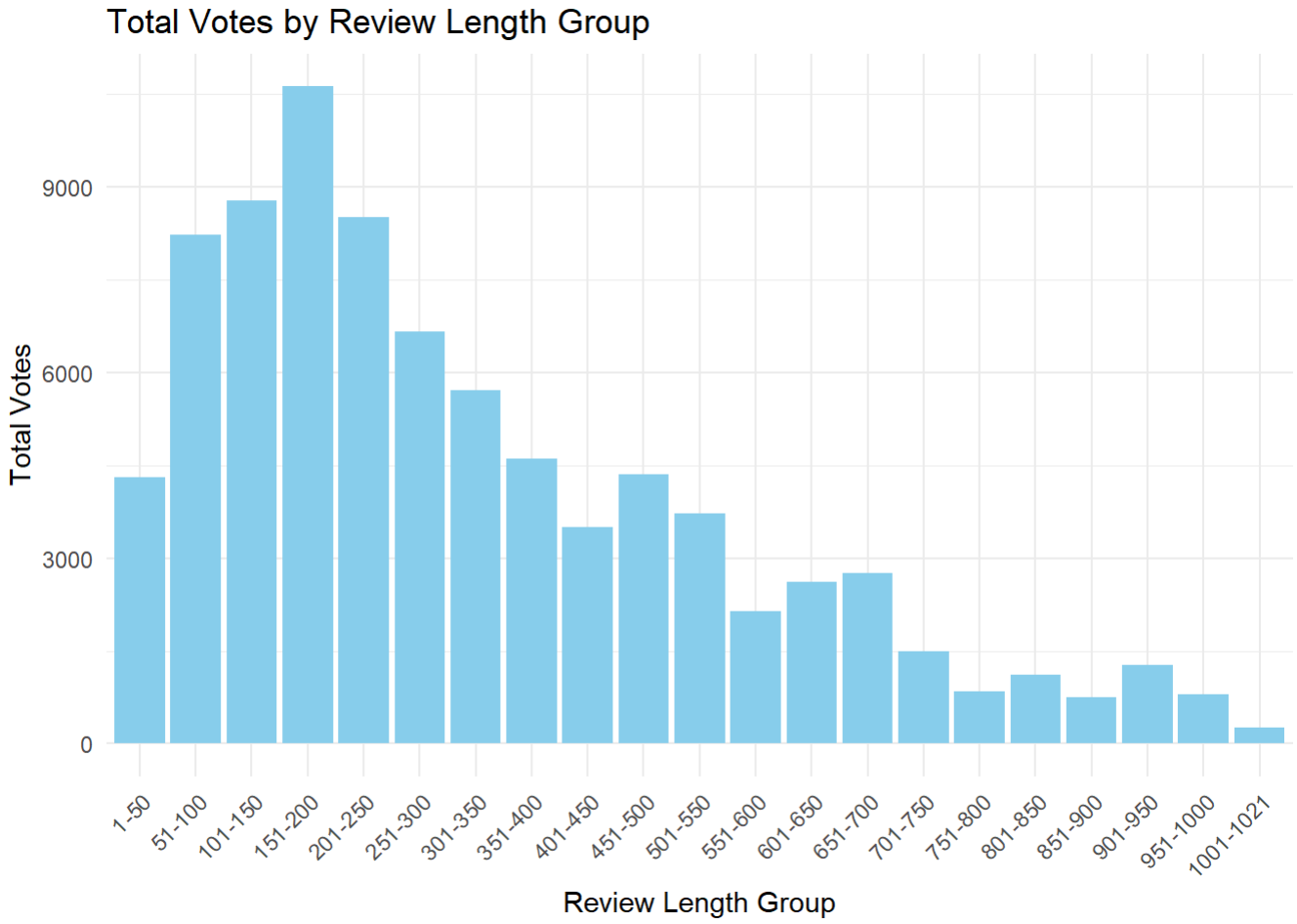
```
       x = "Review Length", y = "Vote") +
  theme_minimal()
```



This does not the capture the relationship well because of presence of outliers.

To visualize the review length and vote relation better and in a broader sense, we can make bins of range, say 50, and compute total votes in each range, followed by making a bar graph of the same.

```
# Creating bins for review_length with the appropriate ranges
bins <- seq(1, max(df$review_length), by = 50)
bins <- c(bins, max(df$review_length) + 1)
labels <- paste(bins[1:(length(bins) - 1)],
                "-", bins[2:length(bins)] - 1, sep = "")
df$length_group <- cut(df$review_length, breaks = bins, right = FALSE,
                       labels = labels, include.lowest = TRUE)

# Computing total votes for each length group
total_votes_by_length <- aggregate(vote ~ length_group, data = df, sum)

ggplot(total_votes_by_length, aes(x = length_group, y = vote)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Votes by Review Length Group",
       x = "Review Length Group",
       y = "Total Votes") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The observed curvilinear relationship between review length and total votes suggests that the initial increase in review length results in a positive impact on the perceived helpfulness of the review, as reflected by higher vote counts. Shorter reviews may fail to provide enough information, leading to fewer votes, but as the review length increases, reviews become more detailed and informative, leading to an increase in helpfulness votes.

However, after a certain threshold, the benefit of increasing review length diminishes. Longer reviews may become repetitive or overly detailed, which could reduce their readability or engagement, resulting in a decrease in helpfulness votes. This suggests that while a moderate level of review length may be optimal for maximizing helpfulness votes, excessively long reviews may not contribute as effectively to the review process, possibly leading to a decrease in perceived usefulness.

This curvilinear relationship highlights the importance of striking a balance between providing sufficient information without overwhelming readers, making it a key factor in optimizing the effectiveness of online reviews. Hence, the results support the hypothesis.

Sentiment Analysis

```
library(quanteda.dictionaries)
library(quanteda.sentiment)
```

```
Attaching package: 'quanteda.sentiment'

The following object is masked from 'package:quanteda':
```

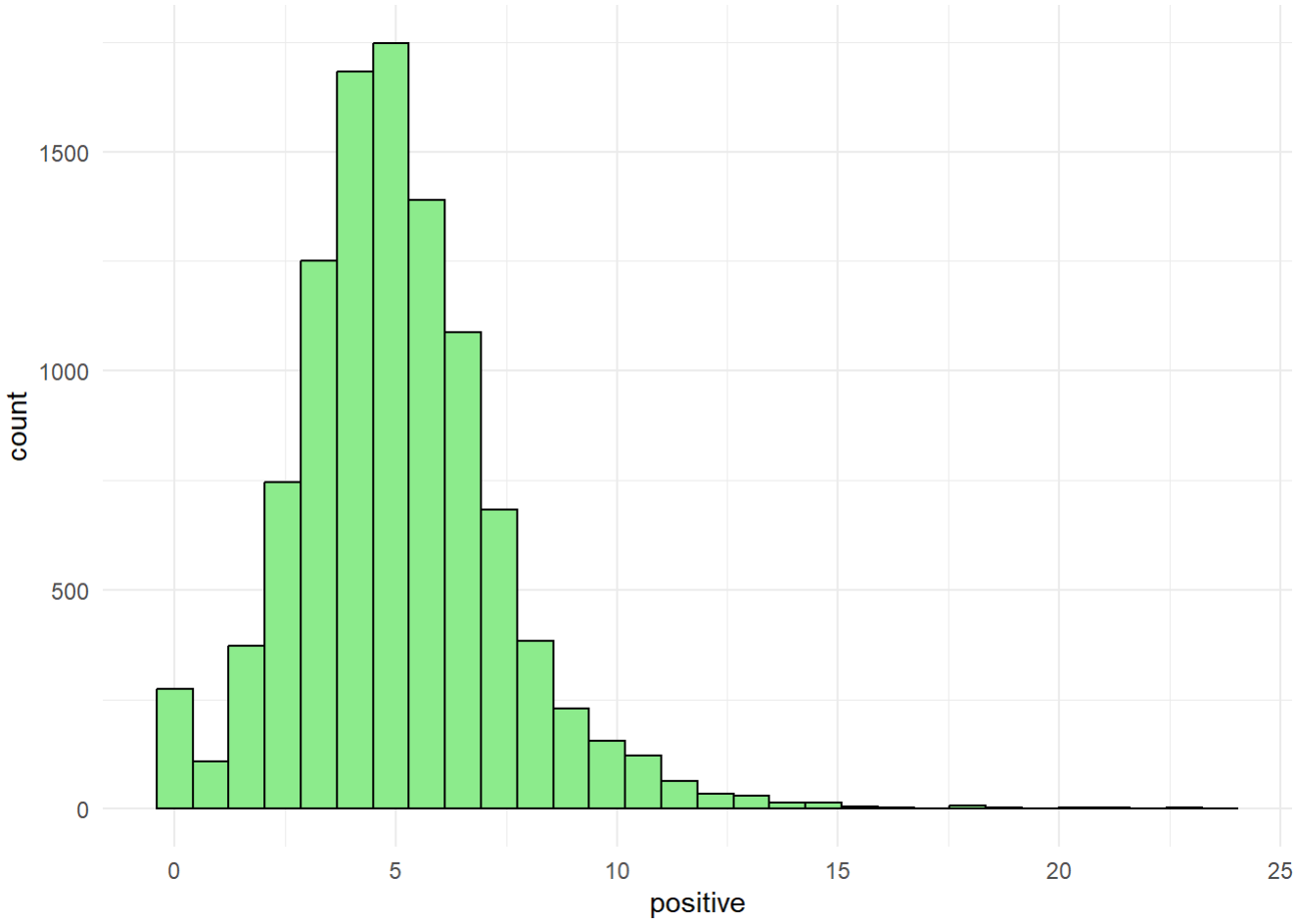
```
data_dictionary_LSD2015
```

```
movieSentiment_nrc <- liwcalike(movie_corpus, data_dictionary_NRC)

#names(movieSentiment_nrc)
#View(movieSentiment_nrc)
```

Positive Reviews

```
ggplot(movieSentiment_nrc) +
  geom_histogram(aes(x = positive), fill = "lightgreen", color = "black") +
  theme_minimal()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Looking at reviews on the right tail (extremely positive):

```
movie_corpus[which(movieSentiment_nrc$positive > 15)]
```

```
Corpus consisting of 32 documents.
text152 :
"Very Clever! Well Acted! A downright good time. Good god I s..."

text174 :
"Perfect educational item for car. Daughter hates learning......"
```

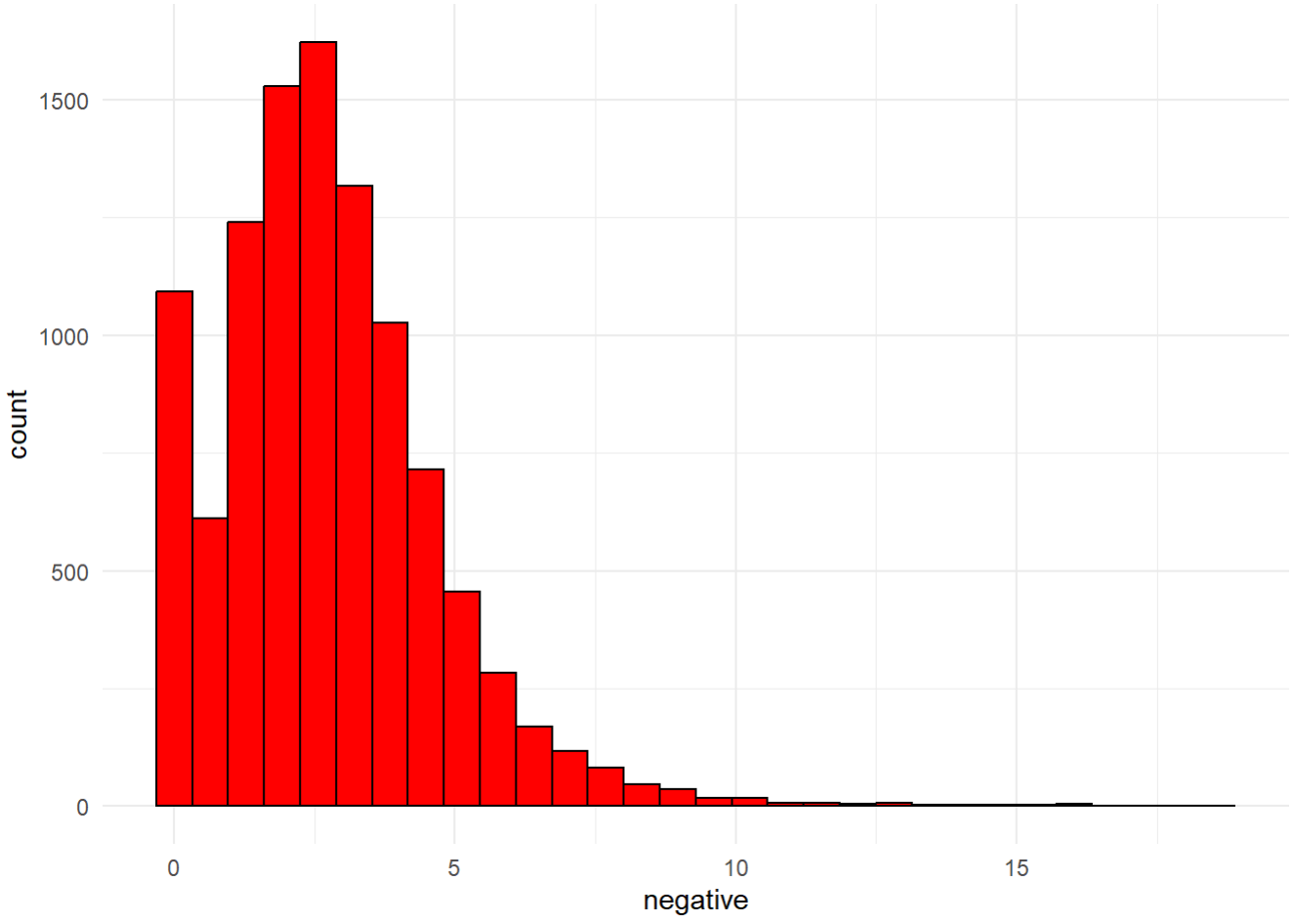
```
text443 :
"Wonderful!!  I haven't seen it all but very good stuff."

text481 :
"A real CLASSIC!  Gave this as a gift to my mother, and she i..."

text682 :
"Both of my kids love these series of books.  They are at a p..."

text700 :
"American dreamsDrugs, cash, girls -- an endless highwayIt's ..."

[ reached max_ndoc ... 26 more documents ]
```

These are indeed very positive reviews!

Negative Reviews

```
        ggplot(movieSentiment_nrc) +
          geom_histogram(aes(x = negative), fill = "red", color = "black") +
          theme_minimal()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
        movie_corpus[which(movieSentiment_nrc$negative > 15)]
```

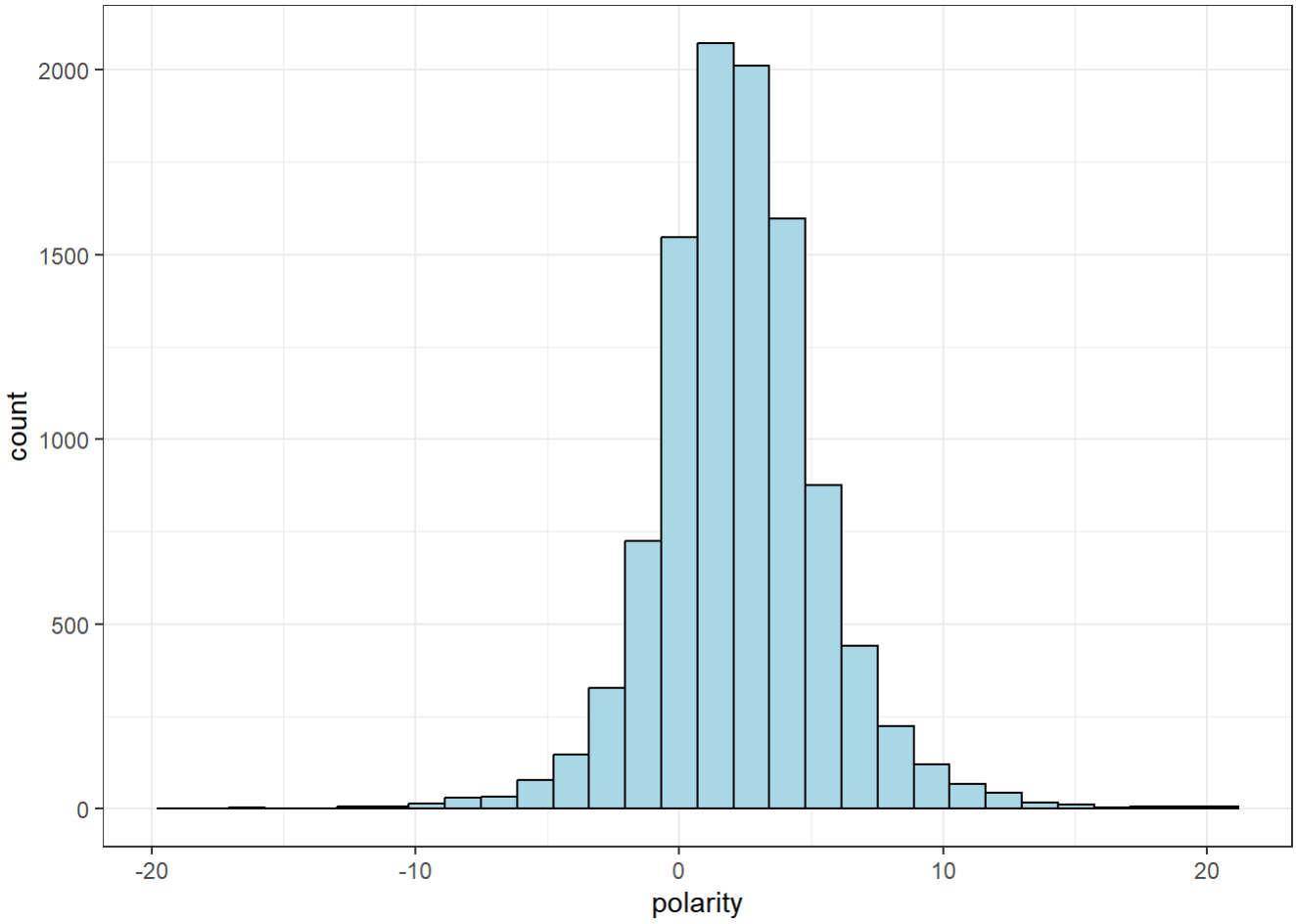
```
Corpus consisting of 10 documents.
text912 :
"There's nothing wrong with this movie, if you like a bunch o..."

text917 :
"I HATE THIS MOVIE! I HATE THIS MOVIE! I HATE THIS MOVIE!I HA..."

text1546 :
""Midnight Express" is the upsetting true story of an America..."

text2730 :
"At first, I thought that this was a horror movie. This nerve..."

text5250 :
"I like 1941. John Belushi I great as Wild Bill Keisto. Rober..."

text5407 :
"Really childish and not Hunter Thompsonesque at all.  One jo..."

[ reached max_ndoc ... 4 more documents ]
```

This also seems to be categorizing negative reviews quite accurately.

Polarity

```
movieSentiment_nrc$polarity <- movieSentiment_nrc$positive - movieSentiment_nrc$negat

ggplot(movieSentiment_nrc) +
  geom_histogram(aes(polarity), fill = "lightblue", color = "black") +
  theme_bw()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
movie_corpus[which(movieSentiment_nrc$polarity < -10)]
```

```
Corpus consisting of 21 documents.
text912 :
"There's nothing wrong with this movie, if you like a bunch o..."

text917 :
"I HATE THIS MOVIE! I HATE THIS MOVIE! I HATE THIS MOVIE!I HA..."

text1462 :
"I love horror movies. I love black Comady, but "Fight Night"..."

text1546 :
""Midnight Express" is the upsetting true story of an America..."

text2888 :
"Yet another overrated movie. I fell asleep watching it, even..."

text3537 :
""Strip Tease" was typical. In it, Burt Reynolds was depicted..."

[ reached max_ndoc ... 15 more documents ]
```

```
movie_corpus[which(movieSentiment_nrc$polarity > 10)]
```

```
Corpus consisting of 165 documents.
text75 :
"This is a good movie to share with your friends and family. ..."

text92 :
"I bought this to watch with family near christmas. I love th..."

text114 :
"no substance, waters down the true meaning of Christmas-Actu..."

text152 :
"Very Clever! Well Acted! A downright good time. Good god I s..."

text165 :
"This dvd does not keep my son's attention as well as other d..."

text174 :
"Perfect educational item for car. Daughter hates learning....."

[ reached max_ndoc ... 159 more documents ]
```

These reviews and their associated sentiments are very accurate!

```
# saving for later use
df$polarity <- movieSentiment_nrc$polarity
```

Testing third hypothesis - Sentiments and their association with the vote count

This is good, but to test my third hypothesis, I need a package which returns positive, negative, and neutral tags for a review. Hence, I use the package SentimentAnalysis.

The SentimentAnalysis package in R calculates sentiment using a lexicon-based approach, relying on predefined sentiment word lists and rules. It classifies text into sentiments (positive, negative, neutral) based on the occurrence of sentiment-laden words. In contrast, the movieSentiment_nrc dataset is a specialized lexicon from the NRC Word-Emotion Association Lexicon that associates words with specific emotions (like joy, sadness) and sentiment. The key difference lies in SentimentAnalysis focusing on broad sentiment polarity, while movieSentiment_nrc provides a more nuanced emotion-based classification.

```
#install.packages("SentimentAnalysis")
library(SentimentAnalysis)
```

```
Warning: package 'SentimentAnalysis' was built under R version 4.4.2


Attaching package: 'SentimentAnalysis'

The following object is masked from 'package:base':

    write
```

How it works:

```
# # Analyze a single string to obtain a binary response (positive / negative)
# sentiment <- analyzeSentiment("Yeah, this was a great soccer game for the German te
# convertToBinaryResponse(sentiment)$SentimentQDAP
#
# # Create a vector of strings
# documents <- c("Wow, I really like the new light sabers!",
#                "That book was excellent.",
#                "R is a fantastic language.",
#                "The service in this restaurant was miserable.",
#                "This is neither positive or negative.",
#                "The waiter forget about my dessert -- what poor service!",
#                "okayish movie.",
#                "this is beautiful and amazing it love it so so much!!!")
#
# # Analyze sentiment
# sentiment <- analyzeSentiment(documents)
# sentiment
#
# # Extract dictionary-based sentiment according to the QDAP dictionary
# sentiment$SentimentQDAP
#
# # View sentiment direction (i.e. positive, neutral and negative)
# convertToDirection(sentiment$SentimentQDAP)
#

documents <- c("This is good",
               "This is bad",
               "This is inbetween")
convertToDirection(analyzeSentiment(documents)$SentimentQDAP)
```

```
[1] positive negative neutral
Levels: negative neutral positive
```

This package cannot handle a lot of data at a time, so splitting into chunks.

```
# First chunk
df1 <- df[1:1000,]
sen1 <- convertToDirection(analyzeSentiment(df1$reviewText)$SentimentQDAP)

# Second chunk
df2 <- df[1001:2000,]
sen2 <- convertToDirection(analyzeSentiment(df2$reviewText)$SentimentQDAP)

# Third chunk
df3 <- df[2001:3000,]
sen3 <- convertToDirection(analyzeSentiment(df3$reviewText)$SentimentQDAP)

# Fourth chunk
df4 <- df[3001:4000,]
sen4 <- convertToDirection(analyzeSentiment(df4$reviewText)$SentimentQDAP)
```

```
# Fifth chunk
df5 <- df[4001:5000,]
sen5 <- convertToDirection(analyzeSentiment(df5$reviewText)$SentimentQDAP)

# Sixth chunk
df6 <- df[5001:6000,]
sen6 <- convertToDirection(analyzeSentiment(df6$reviewText)$SentimentQDAP)

# Seventh chunk
df7 <- df[6001:7000,]
sen7 <- convertToDirection(analyzeSentiment(df7$reviewText)$SentimentQDAP)

# Eighth chunk
df8 <- df[7001:8000,]
sen8 <- convertToDirection(analyzeSentiment(df8$reviewText)$SentimentQDAP)

# Ninth chunk
df9 <- df[8001:9000,]
sen9 <- convertToDirection(analyzeSentiment(df9$reviewText)$SentimentQDAP)

# Tenth chunk
df10 <- df[9001:10000,]
sen10 <- convertToDirection(analyzeSentiment(df10$reviewText)$SentimentQDAP)

# Final chunk (from 10001 to 10419)
df11 <- df[10001:10419,]
sen11 <- convertToDirection(analyzeSentiment(df11$reviewText)$SentimentQDAP)

# Combining all sentiment results into one vector
all_sentiments <- c(sen1, sen2, sen3, sen4, sen5, sen6, sen7, sen8, sen9, sen10, sen1

df$review_sentiment <- all_sentiments
```

```
table(df$review_sentiment)
```

```
negative  neutral positive
    1409      455     8555
```

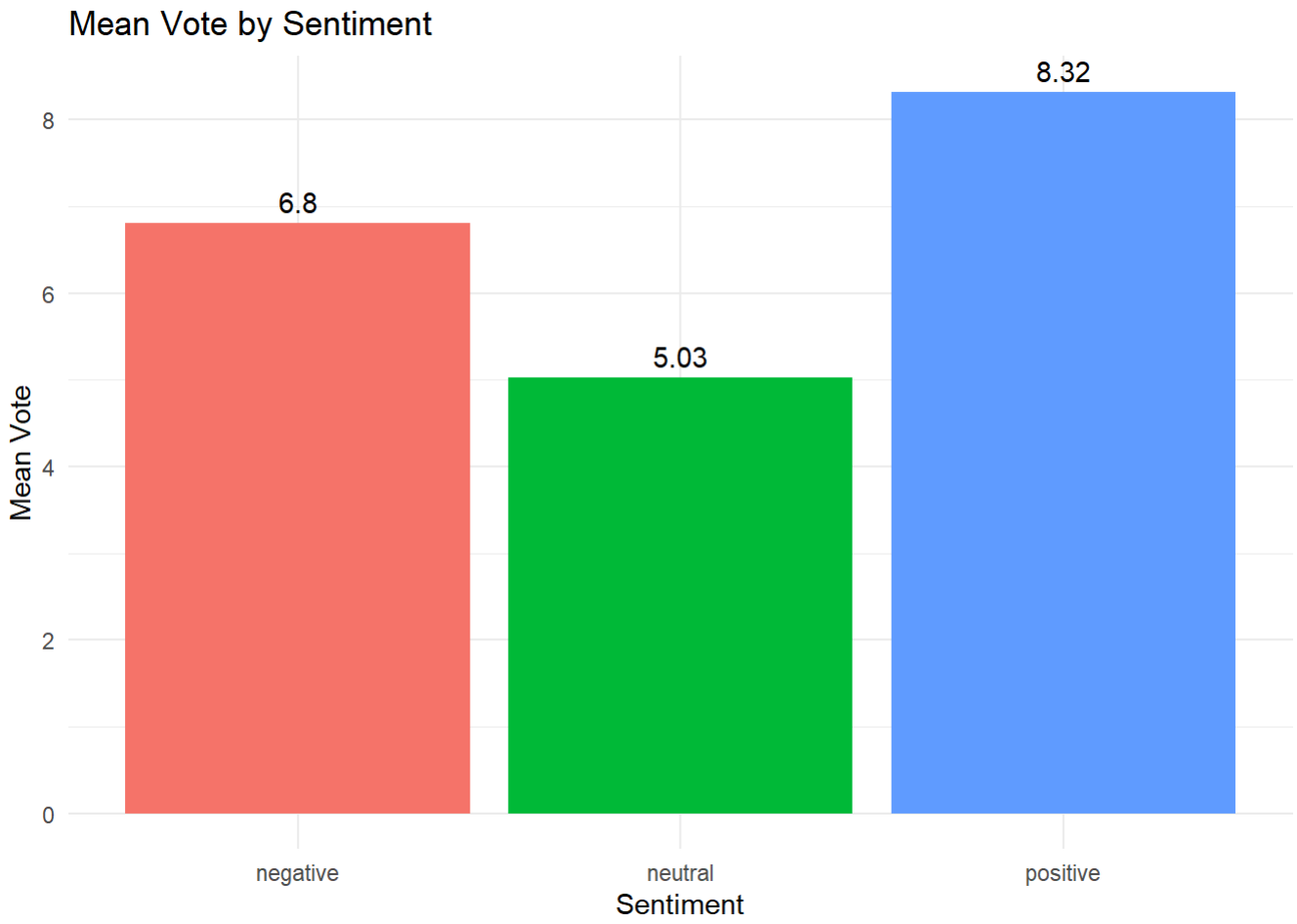
We notice that majority of reviews in our dataset are positive.

Calculating mean vote count for each sentiment:

```
# Calculating mean vote for each sentiment class
mean_votes <- tapply(df$vote, df$review_sentiment, mean)

print(mean_votes)
```

```
negative  neutral positive
6.802697 5.026374 8.323086
```

```r
# Convering mean_votes to a data frame for plotting
mean_votes_df <- data.frame(sentiment = names(mean_votes), mean_vote = mean_votes)

# Plotting the results
ggplot(mean_votes_df, aes(x = sentiment, y = mean_vote, fill = sentiment)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  geom_text(aes(label = round(mean_vote, 2)), vjust = -0.5) +
  labs(title = "Mean Vote by Sentiment", x = "Sentiment", y = "Mean Vote") +
  theme_minimal()
```



The analysis reveals that negative reviews have a mean vote count of 6.8, while positive reviews garner a higher mean vote count of 8.32. Both of these are significantly greater than the mean vote count for neutral reviews, which stands at only 5.03. These findings support the third hypothesis, which posits that strongly positive and strongly negative reviews are more likely to receive higher vote counts compared to neutral sentiment reviews.

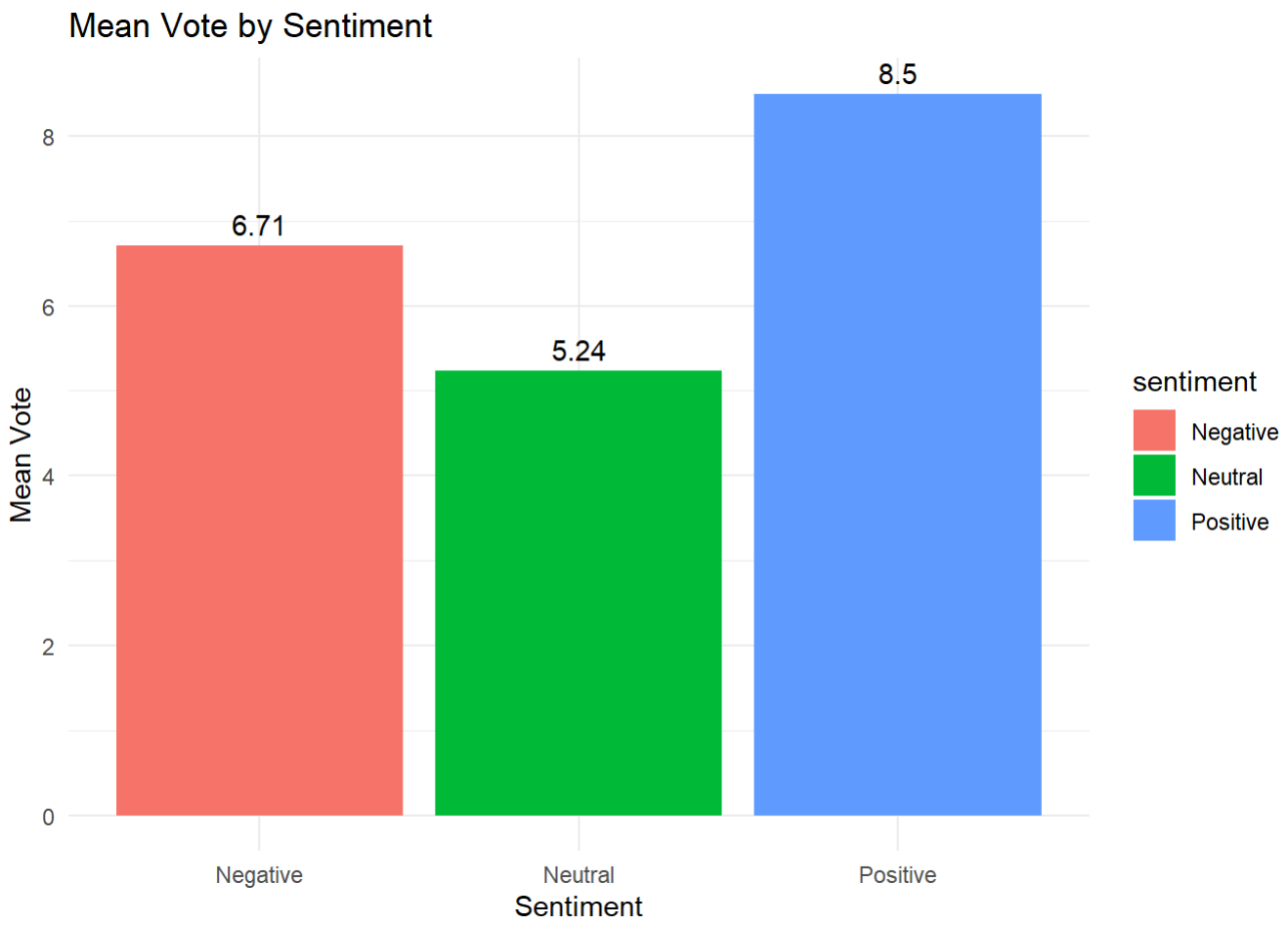
This pattern can be attributed to the fact that extreme sentiments—whether positive or negative—offer more decisive and actionable information to readers. A strongly positive review helps users identify movies they might enjoy, while a strongly negative review serves as a cautionary flag, helping users avoid potentially disappointing choices. In both cases, the emotional clarity of these reviews aids in the decision-making process, prompting readers to express agreement by upvoting them.

In contrast, neutral reviews, which lack strong emotional direction, are less likely to provide readers with clear guidance. These reviews may fail to evoke a strong response from users, leading them to either downvote the review or simply move on to the next one in search of more definitive opinions. Thus, the engagement with reviews appears to be driven by the clarity and decisiveness of sentiment, reinforcing the idea that extreme sentiments have a stronger influence on user interaction and helpfulness voting.

Comparing with movieSentiment_nrc

To test the hypothesis with movieSentiment_nrc as well, we categorize the sentiment such that negative polarity values are termed as “Negative”, polarity values of 0 are termed as “Neutral,” and positive polarity values are termed as “Positive.”

```
# Creating sentiment labels based on polarity
movieSentiment_nrc$sentiment <- ifelse(movieSentiment_nrc$polarity > 0, "Positive", i
movieSentiment_nrc$vote <- df$vote

# Computing mean vote for each sentiment
mean_vote_by_sentiment <- aggregate(vote ~ sentiment, data = movieSentiment_nrc, FUN

library(ggplot2)
ggplot(mean_vote_by_sentiment, aes(x = sentiment, y = vote, fill = sentiment)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(vote, 2)), vjust = -0.5) +
  labs(title = "Mean Vote by Sentiment", x = "Sentiment", y = "Mean Vote") +
  theme_minimal()
```



We get pretty much the same result.

Topic Modelling via Latent Dirichlelt Allocation (LDA)

```
library(text2vec)

# creates string of combined lowercased words
tokens <- tolower(df$reviewText[1:2000])

# performs tokenization
tokens <- word_tokenizer(tokens)

# prints first two tokenized rows
head(tokens, 1)
```

```
[[1]]
 [1] "i"          "have"       "seen"       "x"          "live"
 [6] "many"       "times"      "both"       "in"         "the"
[11] "early"      "days"       "and"        "their"      "more"
[16] "recent"     "reunion"    "shows"      "trust"      "me"
[21] "when"       "i"          "say"        "this"       "they"
[26] "never"      "disappoint" "as"         "a"          "live"
[31] "band"       "this"       "dvd"        "document"   "of"
[36] "a"          "show"       "on"         "their"      "home"
[41] "turf"       "of"         "la"         "is"         "a"
[46] "dream"      "come"       "true"       "can't"      "wait"
```

```
# iterates over each token
it <- itoken(tokens, ids = movie_review$id[1:2000], progressbar = FALSE)

# prints iterator
it
```

```
<itoken>
  Inherits from: <CallbackIterator>
  Public:
    callback: function (x)
    clone: function (deep = FALSE)
    initialize: function (x, callback = identity)
    is_complete: active binding
    length: active binding
    move_cursor: function ()
    nextElem: function ()
    x: GenericIterator, iterator, R6
```

```
# built the vocabulary
v <- create_vocabulary(it)

# print vocabulary
v
```

```
Number of docs: 2000
0 stopwords:  ...
ngram_min = 1; ngram_max = 1
Vocabulary:
             term term_count doc_count
```

```
            <char>      <int>     <int>
    1:    00movie          1         1
    2:         02          1         1
    3: 0345408926          1         1
    4: 0445043024          1         1
    5:         05          1         1
   ---
25113:         to       9907      1726
25114:         of      11817      1773
25115:          a      12319      1826
25116:        and      13258      1841
25117:        the      26750      1915
```

```
dim(v)
```

```
[1] 25117     3
```

```
# prunes vocabulary
v <- prune_vocabulary(v, term_count_min = 10, doc_proportion_max = 0.2)

# check dimensions
dim(v)
```

```
[1] 3750    3
```

```
# creates a closure that helps transform list of tokens into vector space
vectorizer <- vocab_vectorizer(v)
```

```
# creates document term matrix
dtm <- create_dtm(it, vectorizer, type = "dgTMatrix")
```

```
# create new LDA model
lda_model <- LDA$new(n_topics = 7, doc_topic_prior = 0.1,
                     topic_word_prior = 0.01)

# print other methods for LDA
lda_model
```

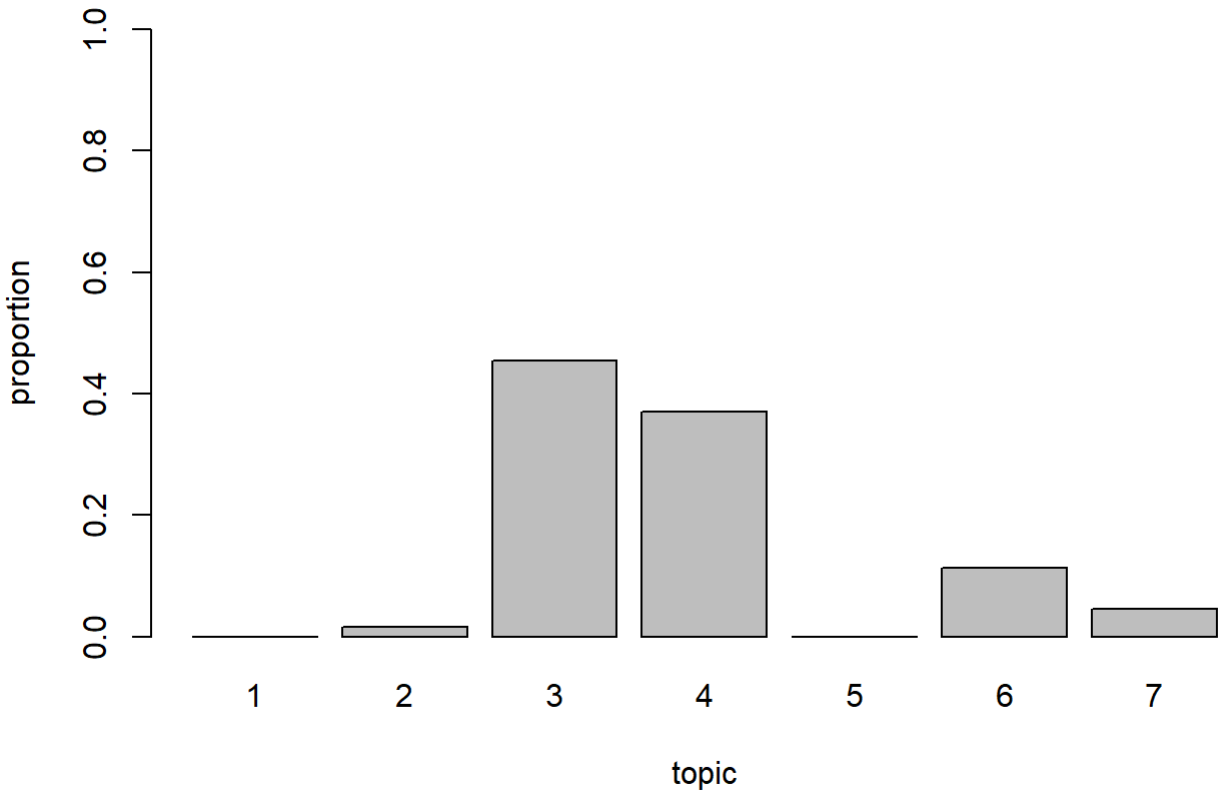
```
<WarpLDA>
  Inherits from: <LDA>
  Public:
    clone: function (deep = FALSE)
    components: active binding
    fit_transform: function (x, n_iter = 1000, convergence_tol = 0.001, n_check_convergence =
10,
    get_top_words: function (n = 10, topic_number = 1L:private$n_topics, lambda = 1)
    initialize: function (n_topics = 10L, doc_topic_prior = 50/n_topics, topic_word_prior =
1/n_topics,
    plot: function (lambda.step = 0.1, reorder.topics = FALSE, doc_len = private$doc_len,
    topic_word_distribution: active binding
    transform: function (x, n_iter = 1000, convergence_tol = 0.001, n_check_convergence = 10,
```

```
  Private:
    calc_pseudo_loglikelihood: function (ptr = private$ptr)
    check_convert_input: function (x)
    components_: NULL
    doc_len: NULL
    doc_topic_distribution: function ()
    doc_topic_distribution_with_prior: function ()
    doc_topic_matrix: NULL
    doc_topic_prior: 0.1
    fit_transform_internal: function (model_ptr, n_iter, convergence_tol, n_check_convergence,
    get_c_all: function ()
    get_c_all_local: function ()
    get_doc_topic_matrix: function (prt, nr)
    get_topic_word_count: function ()
    init_model_dtm: function (x, ptr = private$ptr)
    internal_matrix_formats: list
    is_initialized: FALSE
    n_iter_inference: 10
    n_topics: 7
    ptr: NULL
    reset_c_local: function ()
    run_iter_doc: function (update_topics = TRUE, ptr = private$ptr)
    run_iter_word: function (update_topics = TRUE, ptr = private$ptr)
    seeds: 1296760928.7923 1350757484.24201
    set_c_all: function (x)
    set_internal_matrix_formats: function (sparse = NULL, dense = NULL)
    topic_word_distribution_with_prior: function ()
    topic_word_prior: 0.01
    transform_internal: function (x, n_iter = 1000, convergence_tol = 0.001,
n_check_convergence = 10,
    vocabulary: NULL
```

```r
# fitting model
doc_topic_distr <-
  lda_model$fit_transform(x = dtm, n_iter = 1000,
                          convergence_tol = 0.001, n_check_convergence = 25,
                          progressbar = FALSE)
```

```
INFO  [23:38:09.864] early stopping at 150 iteration
INFO  [23:38:10.600] early stopping at 50 iteration
```

```r
barplot(doc_topic_distr[1, ], xlab = "topic",
        ylab = "proportion", ylim = c(0,1),
        names.arg = 1:ncol(doc_topic_distr))
```




```
# get top n words
lda_model$get_top_words(n = 10, topic_number = c(1L, 2L, 3L, 4L, 5L, 6L, 7L),
                        lambda = 1)
```

```
      [,1]       [,2]         [,3]       [,4]      [,5]       [,6]
 [1,] "ray"      "she"        "series"   "because" "evil"     "action"
 [2,] "blu"      "life"       "version"  "too"     "resident" "over"
 [3,] "special"  "joan"       "years"    "bad"     "game"     "him"
 [4,] "version"  "two"        "book"     "know"    "video"    "fun"
 [5,] "edition"  "war"        "show"     "say"     "zombies"  "last"
 [6,] "scenes"   "through"    "jane"     "your"    "zombie"   "world"
 [7,] "original" "family"     "set"      "think"   "action"   "does"
 [8,] "close"    "history"    "episodes" "why"     "link"     "films"
 [9,] "features" "characters" "did"      "go"      "alice"    "air"
[10,] "release"  "cast"       "made"     "could"   "games"    "made"
      [,7]
 [1,] "travis"
 [2,] "vampire"
 [3,] "him"
 [4,] "driver"
 [5,] "taxi"
 [6,] "night"
 [7,] "horror"
 [8,] "de"
 [9,] "new"
[10,] "bickle"
```

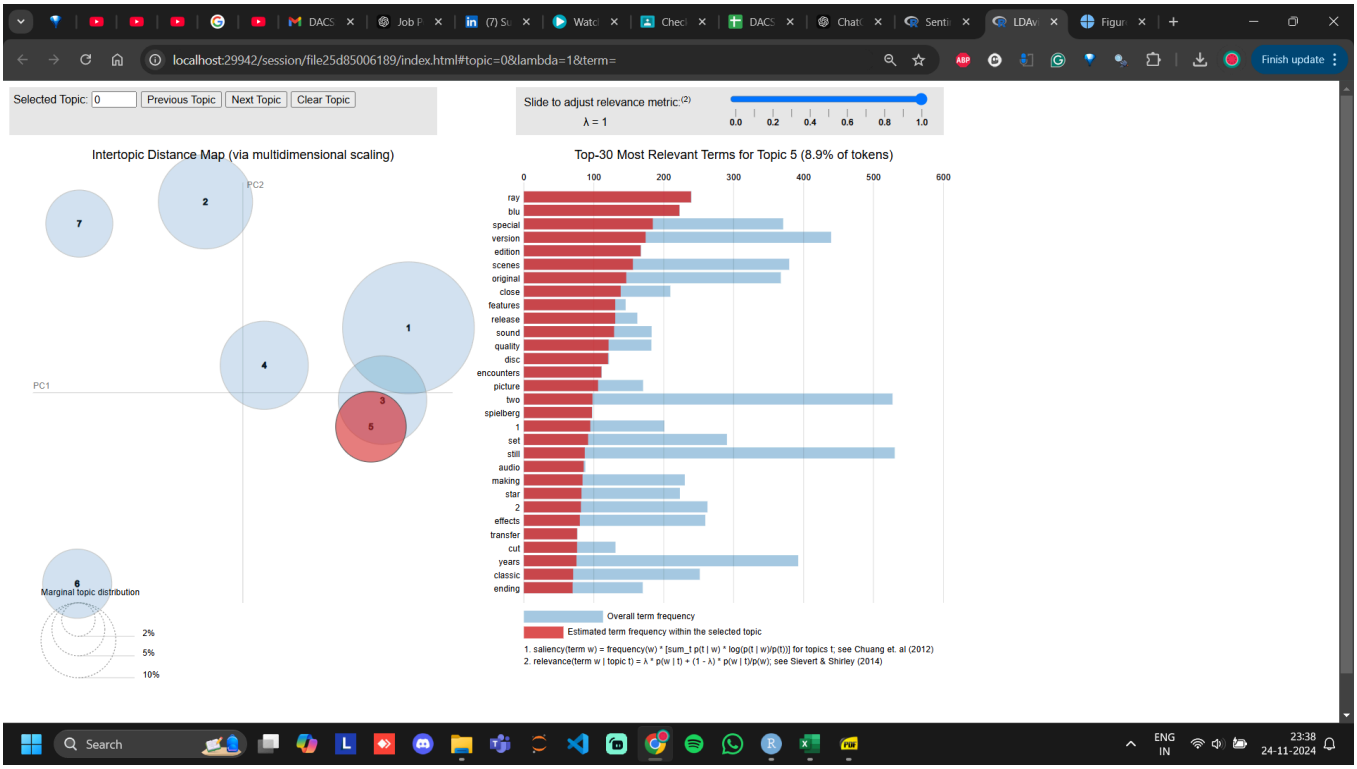
```
library(LDAvis)
```

```
Warning: package 'LDAvis' was built under R version 4.4.2
```

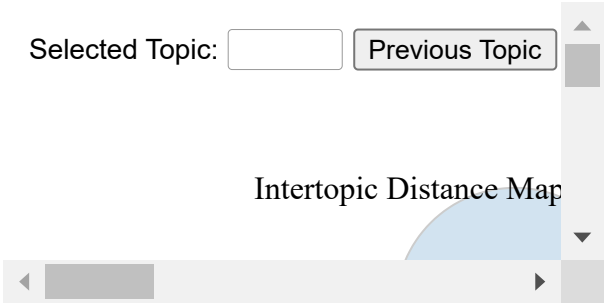
```
lda_model$plot()
```

```
Loading required namespace: servr
```

Attached is the screenshot of the Topic Modelling plot.



Topic Modeling



LDA Plot

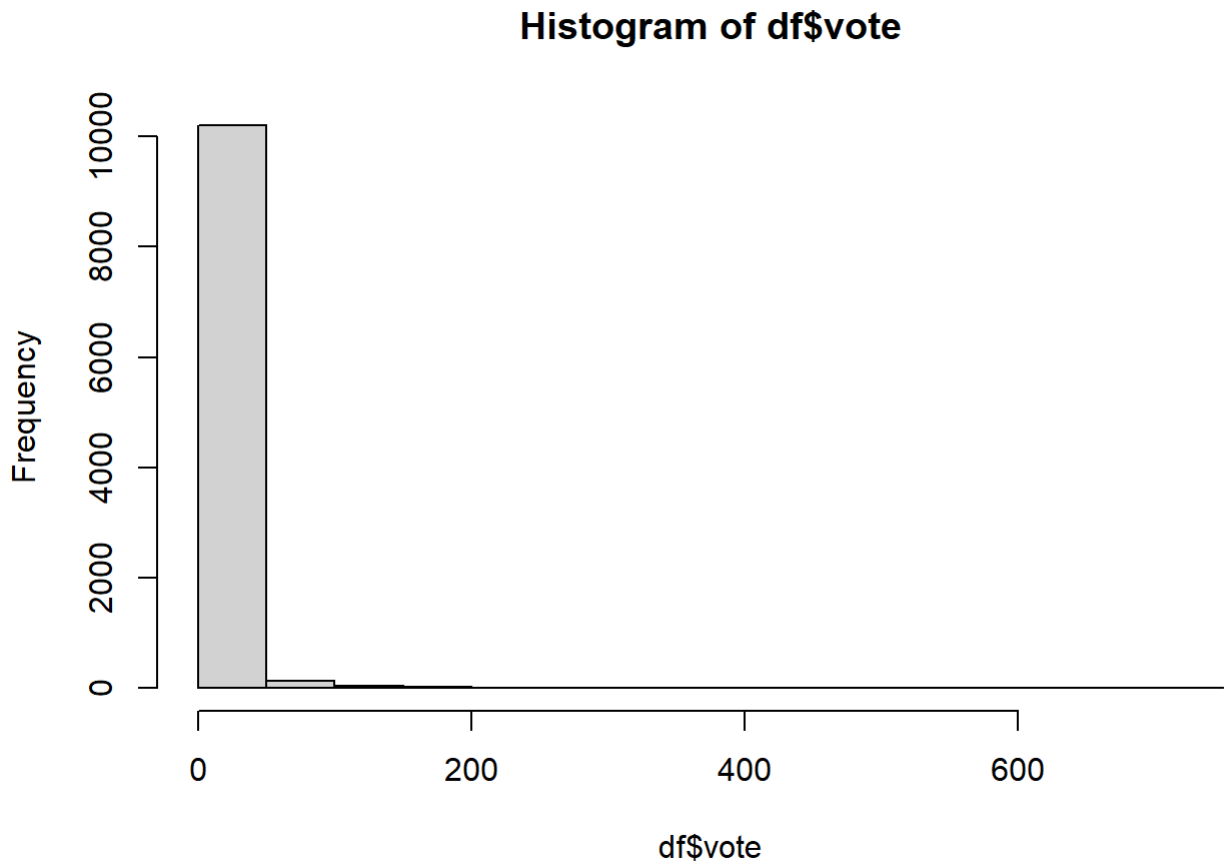
Model Fitting

```
table(df$vote)
```

```
   2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
3382 1901 1221  811  601  400  307  211  171  165  115  121   74   64   66   62
  18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33
  49   42   33   37   34   22   31   16   17   21   19   16   23   16    9   14
```

```
 34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49
 11  12  13   8   5   9   9   9   9  10   6   4   3   7   4   6
 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
  2   6   5   4   8   8   5   6   4   3   4   3   3   4   5   3
 66  67  68  69  71  72  73  74  76  77  78  79  80  81  82  83
  3   3   2   5   3   2   6   5   4   3   2   2   3   1   3   1
 85  86  87  88  89  91  93  94  95  96  97  99 102 104 105 106
  1   2   3   1   2   2   1   2   2   1   1   1   4   1   2   1
107 108 109 110 111 113 114 117 119 120 121 122 130 136 138 139
  3   1   2   1   1   1   2   1   1   2   1   2   2   1   1   1
141 142 143 145 146 147 151 160 163 165 167 174 175 176 181 182
  1   2   1   1   1   1   1   1   1   1   2   1   1   2   2   1
183 184 186 187 188 190 198 199 209 214 218 221 232 234 236 241
  1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1
242 248 266 278 286 287 297 319 340 347 437 599 664 738 745
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
hist(df$vote)
```



The dependent variable “vote” is a count variable and skewed towards 0. A simple linear regression model is unsuitable for such a count data because it assumes a normal distribution of residuals, constant variance (homoscedasticity), and allows negative predictions, which are not meaningful for counts. Additionally, it assumes a linear relationship between predictors and the outcome, whereas count data often exhibits a multiplicative or exponential relationship. Models like Poisson or Negative Binomial regression are better suited, as they address these issues and can handle the skewness and over-dispersion commonly observed in count data.

```
mean(df$vote)
```

```
[1] 7.97351
```

```
var(df$vote)
```

```
[1] 513.1185
```

Hence, our data exhibits overdispersion — where the variance significantly exceeds the mean. This violates the assumption of Poisson Regression that mean must be approximately equal to variance, hence we cannot use this.

Unlike Poisson regression, which assumes mean ≈ variance, the Negative Binomial Regression model introduces an extra dispersion parameter to handle overdispersion. This flexibility improves the model fit and ensures more accurate coefficient estimates and p-values. It's particularly suitable for count data with high variability.

```
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```
glm.nb(vote ~ review_length + polarity + readability, data=df) |> summary()
```

```
Call:
glm.nb(formula = vote ~ review_length + polarity + readability,
    data = df, init.theta = 1.140255236, link = log)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.396220   0.021054  66.316  < 2e-16 ***
review_length 0.002220   0.000051  43.538  < 2e-16 ***
polarity      0.002741   0.003158   0.868    0.385
readability   0.006172   0.001277   4.832 1.35e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(1.1403) family taken to be 1)

    Null deviance: 12663  on 10418  degrees of freedom
Residual deviance: 10598  on 10415  degrees of freedom
AIC: 63439

Number of Fisher Scoring iterations: 1

              Theta:  1.1403
```

```
          Std. Err.:  0.0160

2 x log-likelihood:  -63428.6810
```

Interpretation

- **Review Length**: The positive coefficient ((0.0022)) is highly significant ((p < 0.001)), indicating a strong relationship between review length and vote count.
- **Polarity**: The coefficient ((0.0027)) is not statistically significant ((p = 0.385)), suggesting that sentiment (positive or negative) does not strongly impact vote counts.
- **Readability**: The positive coefficient ((0.0062)) is significant ((p < 0.001)), implying that more readable reviews tend to receive more votes.

This analysis shows that review length and readability influence vote counts, while polarity does not.

Discussion and Conclusion

This study explored the factors influencing the perceived helpfulness of reviews in the movie and TV show domain, specifically focusing on readability, review length, and sentiment. The results indicate several important insights:

Readability: The relationship between readability and helpfulness votes follows a curvilinear trend. While moderately readable reviews tend to garner higher votes, overly simplistic or excessively complex reviews diminish engagement. This finding underscores the importance of crafting reviews that strike a balance between accessibility and informativeness.

Review Length: Similarly, review length showed a curvilinear association with perceived helpfulness. Reviews that are too short fail to provide enough information, while overly long reviews may overwhelm readers. This suggests that concise yet comprehensive reviews are optimal for maximizing helpfulness votes.

Sentiment: Strongly positive and strongly negative reviews received significantly higher votes compared to neutral reviews. This demonstrates that emotional clarity plays a vital role in engaging users and shaping their decision-making processes.

These findings highlight key behavioral patterns in online review consumption and provide actionable insights for reviewers aiming to optimize their content's impact. The results also offer practical implications for platforms, suggesting the need to refine algorithms that prioritize reviews for display based on these patterns.

Future Work

While this research presents valuable insights, several avenues for future exploration remain:

1. Domain Generalization: Extending the analysis to other product categories can help determine if the observed patterns hold universally or if they are specific to the entertainment industry.
2. Cultural and Regional Factors: Investigating how cultural and regional differences influence review engagement and voting behavior can provide a more comprehensive understanding of user dynamics.

3. Incorporating Multimedia Features: Future studies could analyze the role of multimedia elements (e.g., images, videos) within reviews to assess their impact on perceived helpfulness.

4. Temporal Trends: Analyzing how review helpfulness evolves over time, particularly for older content, could shed light on the dynamics of user behavior and relevance.

5. Advanced Sentiment Analysis: Employing advanced models for sentiment detection, such as transformer-based models (e.g., BERT or GPT), may enhance the accuracy and granularity of sentiment classification, providing deeper insights into user engagement.

6. Personalized Recommendations: Exploring how user-specific preferences and behavior influence perceived helpfulness could pave the way for personalized review-ranking algorithms.

By addressing these areas, future research can build upon the findings of this study, advancing our understanding of online review dynamics and their broader implications for platforms, consumers, and content creators.

References

1. Dashtipour, K., Gogate, M., Adeel, A., Larijani, H., & Hussain, A. (2021). Sentiment analysis of persian movie reviews using deep learning. Entropy, 23(5), 596.

2. Qaisar, S. M. (2020, October). Sentiment analysis of IMDb movie reviews using long short-term memory. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-4). IEEE.

3. Kumar, S., De, K., & Roy, P. P. (2020). Movie recommendation system using sentiment analysis from microblogging data. IEEE Transactions on Computational Social Systems, 7(4), 915-923.

4. Daeli, N. O. F., & Adiwijaya, A. (2020). Sentiment analysis on movie reviews using Information gain and K-nearest neighbor. Journal of Data Science and Its Applications, 3(1), 1-7.