Final Project: Sentiment Analysis on a Movie Review Corpus

WAHYU, Zoya Estella 20462503

Load the required library for the project:

```
library(RSelenium)
library(rvest)
library(tidyverse)
library(tidytext)
if (!require(corpus)) install.packages("corpus")
library(corpus) # will be used for word stemming to call text_tokens
if (!require(lemon)) install.packages("lemon")
library(lemon) # will be used to call coord_capped_cart
```

Task 1

Instruction: The starting point could be this webpage (Links to an external site.). * The URLs of user review pages have the same pattern and can be construct programmatically.

Instantiate a new remoteDriver to connect to a running server. Then, use the open method to open the browser and navigate to the specified URL.

```
remDr <- remoteDriver(remoteServerAddr = "localhost", port = 4444, browserName = "chrome")
remDr$open(silent = TRUE)
remDr$navigate("https://www.imdb.com/chart/moviemeter?sort=rk,asc&mode=simple&page=1")
Sys.sleep(2)
```

We can put the list containing the movie titles and their URLs from the html source into results.

```
results <- remDr$getPageSource()[[1]] %>% read_html() %>% html_nodes(".titleColumn")
```

We can obtain the movie title and movie page by using the class “titleColumn” and the attribute “href” on node “a” respectively.

```
# Prepare empty objects of vector type
movie_titles <- character()
review_pages <- character()

# Iterate along "results"
for(i in 1:100){
  # Get the movie title
  title <- results[i] %>%
    # Extract only the text part of the html
    html_text(trim = TRUE) %>%
    # Up to this point, the result should be:
    # "<movie title>\n        (<year released>)\n            <rank>\n(\n\n<rank difference>)"
    # To get the movie title, we only need a part of the string before the first "\n"
    {str_sub(.,1,str_locate(.,"\\n")[,"start"]-1)}
  # Concatenate the currently obtained "title" to the back of "movie_titles" vector
  movie_titles <- c(movie_titles,title)

  # Get the corresponding movie page
  page <- results[i] %>%
    # Select html nodes with <a> tag
    html_nodes("a") %>%
    # Select html that has "href" attribute
    html_attr("href") %>%
    # Add "https://www.imdb.com" to the front of the obtained string to match the correct movie page URL
    str_c("https://www.imdb.com",.)

  # The "page" substring before "?pf_rd_m=" will bring us to the same movie page URL
  # To obtain the movie review URL, we can simply add "reviews" to the back of the substring
  review_page <- page %>% str_extract(".*/?pf_rd_m=") %>% str_sub(1,-10) %>% str_c("reviews")
  # Concatenate the currently obtained "review_page" to the back of "review_pages" vector
  review_pages <- c(review_pages,review_page)
}

head(movie_titles)
```

```
## [1] "Knives Out"     "365 Days"       "The Old Guard"  "Dil Bechara"
## [5] "Jojo Rabbit"    "Raat Akeli Hai"
```

```
head(review_pages)
```

```
## [1] "https://www.imdb.com/title/tt8946378/reviews"
## [2] "https://www.imdb.com/title/tt10886166/reviews"
## [3] "https://www.imdb.com/title/tt7556122/reviews"
## [4] "https://www.imdb.com/title/tt8110330/reviews"
## [5] "https://www.imdb.com/title/tt2584384/reviews"
## [6] "https://www.imdb.com/title/tt12567088/reviews"
```

```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(results, i, title, page, review_page) # run this code for environment data clarity
```

Task 2

Instruction: Collect user reviews and ratings (note that some reviews may have no ratings) for 100 movies, 100 reviews per movie (with spoilers excluded; note that on each review page, there is a checkbox labeled “Hide Spoilers”). * Use R Selenium (covered in the last lab) to render full content before scraping a user review page: + tick the checkbox to hide spoilers + load all reviews for each movie by repeatedly clicking the “load more” button at the bottom of a review page. The number of clicks can be determined by the number of reviews displayed above the “Hide Spoilers” checkbox. + load the text content for each long review by clicking the down arrow button.

Use iteration to render all review pages and get all movie reviews and ratings.

```
# Since we will acquire 100 reviews for each movie, we should prepare vectors of length 100*length(movie_tit
les)
reviews <- vector(length = 100*length(movie_titles))
ratings <- vector(length = 100*length(movie_titles))

# Iterate along "results"
for(i in seq_along(review_pages)){
#for(i in seq(1,100,5)) { # DEBUG
```

```r
  # Navigate the browser to review_pages[i]
  remDr$navigate(review_pages[i])
  Sys.sleep(2)

  # Tick the checkbox to hide spoilers by clicking the element which has "faceter-facets-text" class name
  hide_spoiler <- remDr$findElements(using = "css", ".faceter-facets-text")
  hide_spoiler[[1]]$clickElement()
  Sys.sleep(2)

  review_count <- remDr$getPageSource()[[1]] %>% read_html %>%
    # The number of reviews can be obtained inside the "span" node inside the "header" class
    html_nodes(".header") %>% html_nodes("span") %>% html_text(trim = TRUE) %>%
    # Up to this point, the result should be:
    # "<number of reviews> Reviews" ""               "Hide Spoilers" ""
    # To get the number of reviews, we only need the first element of the vector
    .[1] %>%
    # Replace " Reviews" or " Review" to an empty string,
    # for example, "2,283 Reviews" will be replaced to "2,283"
    str_replace(pattern = " Reviews| Review", replace = "") %>%
    # Replace "," to an empty string,
    # for example, "2,283" will be replaced to "2283"
    str_replace(pattern = ",", replace = "") %>%
    # Change the data type from character to numeric
    as.numeric()

  if(review_count == 0){
    # If there is no review, we can skip scraping the current review page and just go to the next review pag
e
    next
  } else if(review_count > 100){
    # We only need to scrap 100 reviews per movie
    review_count = 100
  }

  # If review_count > 25, we need to press the "Load More" button as many as specified from load_count
  load_count <- (review_count-1) %/% 25
  if(load_count != 0){
    # Load all reviews for each movie by repeatedly clicking the “Load More” button
    for(j in seq_len(load_count)){
      # Click the "Load More" button by finding the element which has "load-more-trigger" id
      load_more <- remDr$findElements(using = "id", "load-more-trigger")
      load_more[[1]]$clickElement()
      Sys.sleep(2)
    }
  }

  # Load the text content for each long review by clicking the down arrow button which has "expander-icon-wr
apper.show-more__control" class name
  expander_list <- remDr$findElements(using = "css", ".expander-icon-wrapper.show-more__control")
  # Click all the down arrow button by clicking as many as the page has
  for(k in seq_along(expander_list)){
    expander_list[[k]]$clickElement()
  }

  # Return list of all reviews on the page which can be obtained by finding "lister-item-content" class name
  contents <- remDr$getPageSource()[[1]] %>% read_html %>% html_nodes(".lister-item-content")

  # Iterate along the review
  for(k in seq_along(contents)){

    # Save the k-th review rating of the i-th movie in rating[100*(i-1)+k]
    ratings[100*(i-1)+k] <- contents[[k]] %>%
      html_nodes(".rating-other-user-rating") %>%
      html_text(trim = TRUE) %>%
      # Up to this point, the result should be: "<number>/10"
      # Replace "/10" to an empty string, for example "10/10" will be replaced "10"
      str_replace(pattern = "/10", replacement = "") %>%
      # If the is no rating, i.e. length of the string == 0, we can fill the vector by NA
      # Else, we change the string into a numeric data type
      {ifelse(length(.)==0, NA, as.numeric(.))}

    # Save the k-th review of the i-th movie in review[100*(i-1)+k]
    reviews[100*(i-1)+k] <- contents[[k]] %>%
      html_nodes(".text.show-more__control") %>%
      html_text(trim = TRUE)
  }
}

# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(i, j, k, load_count, load_more, review_count, hide_spoiler, expander_list, contents) # run this code for
environment data clarity
```

We can close the remDr as we will not need the remote driver for the rest of the tasks.

```
# Close the browser that was openned by remDr
# Note that the server will still be running on the background
remDr$close()
rm(remDr)
```

For the analysis of the forthcoming tasks, we will use the data of tb from “reviews.csv” file which was scrapped from the web on 14 August 2020 7:31 PM HKT.

```
#write_csv(tibble(titles=rep(movie_titles, rep(100,100)), reviews, ratings), "reviews.csv")

# Read the csv and put it on a tibble named tb.
tb <- read_csv("reviews.csv", col_names = TRUE)
head(tb)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 6 x 3
##   titles    reviews                                                 ratings
##   <chr>     <chr>                                                     <dbl>
## 1 Knives O~ "Nothing was typical about this. Everything was beautifully~      9
## 2 Knives O~ "What an excellent film by Rian Johnson; definitely feels l~      8
## 3 Knives O~ "With the exception of my wife and daughter, and possibly m~      8
## 4 Knives O~ "As a fan of mysteries (Sherlock Holmes, Poirot, Columbo, M~      8
## 5 Knives O~ "Best movie of the year, love everything about it, what I'm~     10
## 6 Knives O~ "I enjoyed this a lot. Clever, interesting and plausible in~      9
```

To overwrite and save the newly scrapped data into tb, **enable** the chunk below.

Task 3

Instruction: Sample 2500 positive reviews and 2500 negative reviews (for example, a movie receiving a rating higher than 6 is considered positive or getting thumbs-up, while one with a rating less than 5 is negative or getting thumbs-down) to form a movie review corpus. * Note that the most noteworthy fact for review corpora collected from the Web is that the majority of reviews are highly positive, with 6-star to 10-star reviews. If you cannot find a sufficient number of negative reviews from your 10,000 reviews, report the imbalance of the distribution.

We first separate the reviews according to their ratings, wherein a review of rating greater than 6 will be considered as a positive review and if the rating is less than 5 will be considered as a negative review.

```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(movie_titles, ratings, review_pages, reviews) # run this code for environment data clarity


pos_reviews <- tb %>% filter(ratings>=6)

# We exclude ratings==0 from neg_reviews as it represents reviews which has no ratings.
neg_reviews <- tb %>% filter(ratings<=5 & ratings!=0)
```

From all the reviews available, we randomly sample 2,500 reviews from each group.

```
writeLines(paste("Total positive reviews:", nrow(pos_reviews)))
```

```
## Total positive reviews: 4374
```

```
writeLines(paste("Total negative reviews:", nrow(neg_reviews)))
```

```
## Total negative reviews: 2800
```

```
set.seed(1)
pos_sample <- nrow(pos_reviews) %>%
  # Get 2500 random numbers from 1 to nrow(pos_reviews)
  # In the case where nrow(pos_reviews) <= 2500, all numbers from 1 to nrow(pos_reviews) will be obtained
  {sample(seq_len(.), min(.,2500))} %>%
  # The random numbers will be used as the index of the reviews to be selected
  pos_reviews[.,] %>%
  # Only select the "index", "reviews", and "ratings" column
  transmute(index = row_number(), reviews, ratings)

# Similarly, for the negative sample:
set.seed(1)
neg_sample <- nrow(neg_reviews) %>%
  {sample(seq_len(.), min(.,2500))} %>%
  neg_reviews[.,] %>%
  transmute(index = row_number(), reviews, ratings)

writeLines(paste("Total positive review sample:", nrow(pos_sample)))
```

```
## Total positive review sample: 2500
```

```
writeLines(paste("Total negative review sample:", nrow(neg_sample)))
```

```
## Total negative review sample: 2500
```

```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(tb, pos_reviews, neg_reviews) # run this code for environment data clarity
```

Task 4

Instruction: Use the **BING** lexicon to find sentiment-associated words and predict sentiment for each review. Evaluate outcomes by contrasting them with actual ratings.

Get the sentiment value for each index/review from the positive and negative review samples:

```r
pos_sentiments <- pos_sample %>%
  # Tokenize all reviews by word
  unnest_tokens(word, reviews) %>%
  # Remove the not informative words from the tibble
  anti_join(stop_words, by = "word") %>%
  # Add a "sentiment" column that will predict the sentiment of the corresponding word
  inner_join(get_sentiments("bing"), by = "word") %>%
  # Count the number of positive and negative sentiment words on each review/index
  count(index, ratings, sentiment) %>%
  # Convert the tibble into a wide format by creating "negative" and "positive" sentiment column
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  # Add a "sentiment" column which is used as
  #  the point difference between the total positive and negative sentiment words on each index
  # If the "sentiment" has a positive value,
  #  it means that there are more positive words than negative words on that index, and vice versa
  mutate(ratings, sentiment = positive - negative)


# Similarly, for the negative sentiments:
neg_sentiments <- neg_sample %>%
  unnest_tokens(word, reviews) %>%
  anti_join(stop_words, by = "word") %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(index, ratings, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(ratings, sentiment = positive - negative)
```

Get the number of sentiments grouped by the ratings and the sentiment value, and also the statistics of the sentiments.

```r
# For each rating, count how many indices/reviews have a certain sentiment value
count_sentiments <- bind_rows(pos_sentiments, neg_sentiments) %>%
  count(ratings, sentiment, name="count")

# For each review, get the mean, sd, and number of sentiments
summarise_sentiments <- bind_rows(pos_sentiments, neg_sentiments) %>%
  group_by(ratings) %>%
  # Here, total = number of reviews on each rating
  summarise(mean=mean(sentiment), median=median(sentiment), min=min(sentiment), max=max(sentiment), sd=sd(se
ntiment), total=n())

# Add the summary of the sentiments to count_sentiment, and save it on sentiments_tb
sentiments_tb <- left_join(count_sentiments, summarise_sentiments, by="ratings")

summarise_sentiments
```

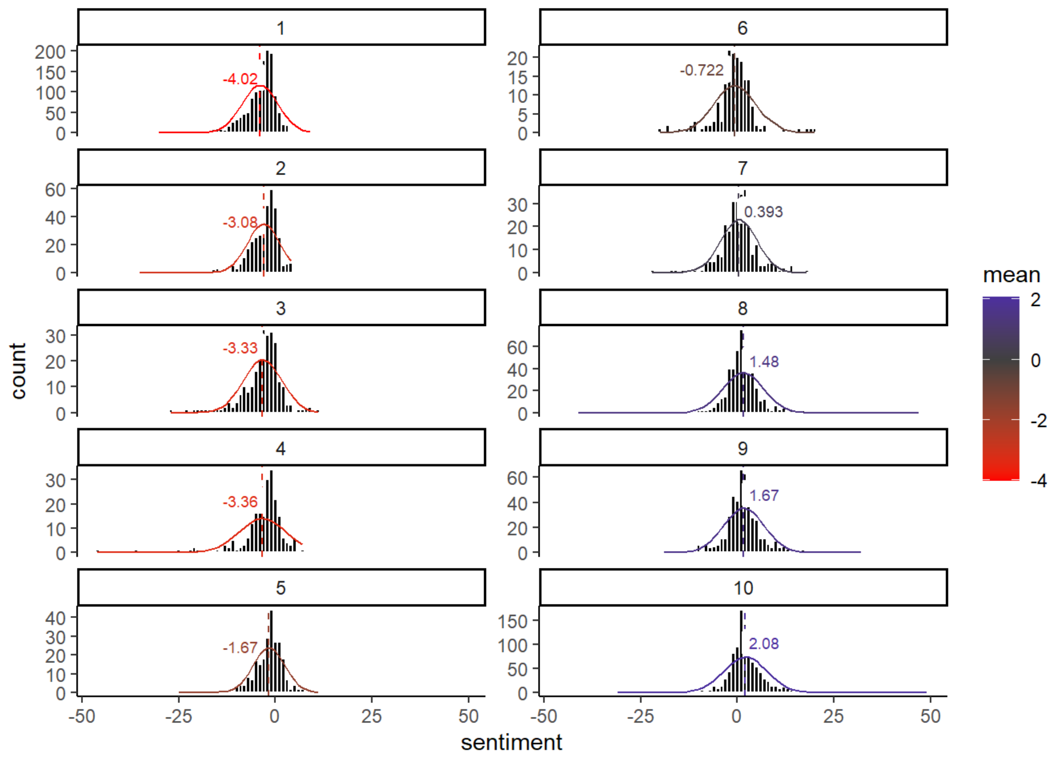
```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 10 x 7
##    ratings   mean median   min   max    sd total
##      <dbl>  <dbl>  <dbl> <int> <int> <dbl> <int>
##  1       1 -4.02      -3   -30     9  4.62  1351
##  2       2 -3.08      -2   -35     4  4.20   365
##  3       3 -3.33      -3   -27    11  4.99   255
##  4       4 -3.36      -2   -46     7  6.09   214
##  5       5 -1.67      -1   -25    11  4.09   246
##  6       6 -0.722     -1   -20    20  5.39   169
##  7       7  0.393      0   -22    18  4.69   270
##  8       8  1.48       1   -41    47  5.20   474
##  9       9  1.67       1   -19    32  5.26   468
## 10      10  2.08       2   -31    49  5.59  1036
```

From the “summarise_sentiments” tibble displayed above, we can see from the “sd” column that the sentiments on each group are similarly dispersed. We can also clearly spot from the other columns that the sentiment value has some correlation to the ratings seeing that the mean, median, min, and max sentiment values generally increase as the rating increases.

To further visualize the tibble, we can see the plot produced by the chunk below.

```
ggplot(sentiments_tb, aes(sentiment)) +
  # Create multipanel plots grouped by "ratings" with 2 plots per column and filled in vertical manner
  # The y axis should be scaled free since the number of words on each ratings are not generally the same du
e to the imbalance of rating distribution
  facet_wrap(~ ratings, ncol=2, dir="v", scales="free_y") +
  # Add main title label
  labs(main="Sentiment Value Distribution by Ratings") +
  # Add a histogram to each plot with coord(sentiment, count(sentiment))
  geom_histogram(data=bind_rows(pos_sentiments, neg_sentiments),
                 mapping=aes(sentiment), binwidth=1, colour = "white", fill="black") +
  # Add a vertical line for the mean(sentiment) of the grouped data
  geom_vline(data=summarise_sentiments,
             aes(xintercept=mean, colour=mean), linetype="dashed") +
  # Add a label that contains the x coordinate of the vertical line, i.e. the mean of the data
  geom_label(data=summarise_sentiments,
             aes(x=ifelse(mean>0, 7, ifelse(mean<0, -9, 0)), y=total/10,
                 label=signif(mean,3), colour=mean),
             label.size=NA, size=2.5) +
  # The sufficiently large independent random samples should be approximately normally distributed
  # We can add a line of a normal distribution with the data's mean and sd to help visualizing the data's di
stribution
  geom_line(aes(y=dnorm(sentiment, mean, sd)*total, colour=mean)) +
  scale_colour_gradient2(low="red", high="blue", mid="grey25") +
  theme_classic()
```



```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(pos_sentiments, neg_sentiments, count_sentiments, summarise_sentiments, sentiments_tb) # run this code f
or environment data clarity
```

From the above multipanel plots, we can see that the sentiments and ratings are correlated to each other. The higher the number of the ratings, the higher the value of the sentiments where the sentiment value is defined as the difference between the positive and negative sentiment words. The higher the value of the sentiment means there are much more positively interpreted words than the negative counterpart.

Task 5

Instruction: Find 5 most commonly-used positive and negative words in the corpus. Create a multipanel plot. In each panel, plot a graphic similar to the following one, which describes the extent to which a word affects the rating of the review where it appears. Specifically, w represents the word, while c represents categories defined in terms of ratings. For instance, the point at rating 3 means that a review containing “disappoint” has probability 0.14 of receiving rating 3.

Get all words in both positive and negative review sample, and find the word sentiment and the word stem.

```
word_sentiment_rating <- bind_rows(pos_sample, neg_sample) %>%
  # Tokenize all reviews by word
  unnest_tokens(word, reviews) %>%
  # Remove the not informative words from the tibble
  anti_join(stop_words, by = "word") %>%
  # Add a "sentiment" column that will predict the sentiment of the corresponding word
  inner_join(get_sentiments("bing"), by = "word") %>%
  # Remove the "index" column and add the "stem_word" column that stands for the word stem of the words in t
he "word" column
  transmute(ratings, word, stem_word=(text_tokens(word, stemmer="en") %>% unlist), sentiment)

word_sentiment_rating
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 49,972 x 4
##    ratings word       stem_word sentiment
##      <dbl> <chr>      <chr>     <chr>
##  1       8 clever     clever    positive
##  2      10 violently  violent   negative
##  3      10 perfect    perfect   positive
##  4      10 perfection perfect   positive
##  5      10 proper     proper    positive
##  6      10 regret     regret    negative
##  7       9 touted     tout      negative
##  8       9 freedom    freedom   positive
##  9       9 died       die       negative
## 10       9 hot        hot       positive
## # ... with 49,962 more rows
```

Get top 5 frequently used positive and negative stemmed words by counting the words and their sentiments from the data in “word_sentiment_rating”.

```r
# We first focus only on getting top 5 stem words with positive sentiments
pos_5 <- word_sentiment_rating %>%
  # Filter only rows which have positive sentiments
  filter(sentiment=="positive") %>%
  # Count the number of occurrence of a certain "stem_word" appear in "word_sentiment_rating"
  count(stem_word, sentiment, sort=TRUE) %>%
  # Add another row which will later be used to save the derived/original word suffixes
  bind_cols("word_suffixes"=NA) %>%
  # Get the top 5 frequently used stem words
  head(5)

# This iteration will be used to fill in the "word_suffixes" column
for(i in seq_len(nrow(pos_5))){
  pos_5$word_suffixes[i] <- word_sentiment_rating %>%
    # Filter only rows which contain the top 5 stem words
    filter(stem_word==pos_5$stem_word[i]) %>%
    # Obtain the distinct derived word from the "word" column in "word_sentiment_rating"
    {unique(.$word)} %>%
    # Get only the suffixes of the derived word by removing the stem word from the word itself
    str_remove_all(pos_5$stem_word[i]) %>%
    # Remove the original word that is identical to the word stem from the word suffixes vector
    .[.!=""] %>%
    # Combine all the suffixes into a single string
    str_c(collapse="/")
  # When the string is empty, it means there is no additional word suffix for the word stem
  # So, we can put NA to the corresponding cell
  if(pos_5$word_suffixes[i]=="") pos_5$word_suffixes[i]=NA
}

# Report the findings
writeLines(str_c("Top 5 most commonly-used positive words:",
                 str_c(pos_5$stem_word,
                       ifelse(!is.na(pos_5$word_suffixes), str_c("(", pos_5$word_suffixes, ") "), " "), pos
_5$n, " times", collapse="\n"),
                 sep="\n"))
```

```
## Top 5 most commonly-used positive words:
## love(d/loving/s/ly) 1351 times
## enjoy(able/ed/ment/ing/s/ably) 751 times
## perfect(ion/ly) 561 times
## fun 504 times
## amaz(ing/ingly/ed/ement/es) 489 times
```

```r
writeLines("\n")
```

```r
# Similarly, for top 5 most frequently used negative stemmed words:

neg_5 <- word_sentiment_rating %>%
  filter(sentiment=="negative") %>%
  count(stem_word, sentiment, sort=TRUE) %>%
  bind_cols("word_suffixes"=NA) %>%
  head(5)

for(i in seq_len(nrow(neg_5))){
  neg_5$word_suffixes[i] <- word_sentiment_rating %>%
    filter(stem_word==neg_5$stem_word[i]) %>%
    {unique(.$word)} %>%
    str_remove_all(neg_5$stem_word[i]) %>%
    .[.!=""] %>%
    str_c(collapse="/")
  if(neg_5$word_suffixes[i]=="") neg_5$word_suffixes[i]=NA
}

writeLines(str_c("Top 5 most commonly-used negative words:",
                 str_c(neg_5$stem_word,
                       ifelse(!is.na(neg_5$word_suffixes), str_c("(", neg_5$word_suffixes, ") "), " "), neg
_5$n, " times", collapse="\n"),
                 sep="\n"))
```

```
## Top 5 most commonly-used negative words:
## bad(ly) 1082 times
## plot 1000 times
## bore(boring/d/s) 520 times
## wast(e/ed/ing) 423 times
## funni(funny) 410 times
```

```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(pos_sample, neg_sample, i) # run this code for environment data clarity
```

After securing the top 5 positive and negative sentiments' stem words, we should get the rating distribution for each of the 10 words by filtering and counting the appropriate rows from the "word_sentiment_rating" corpus.

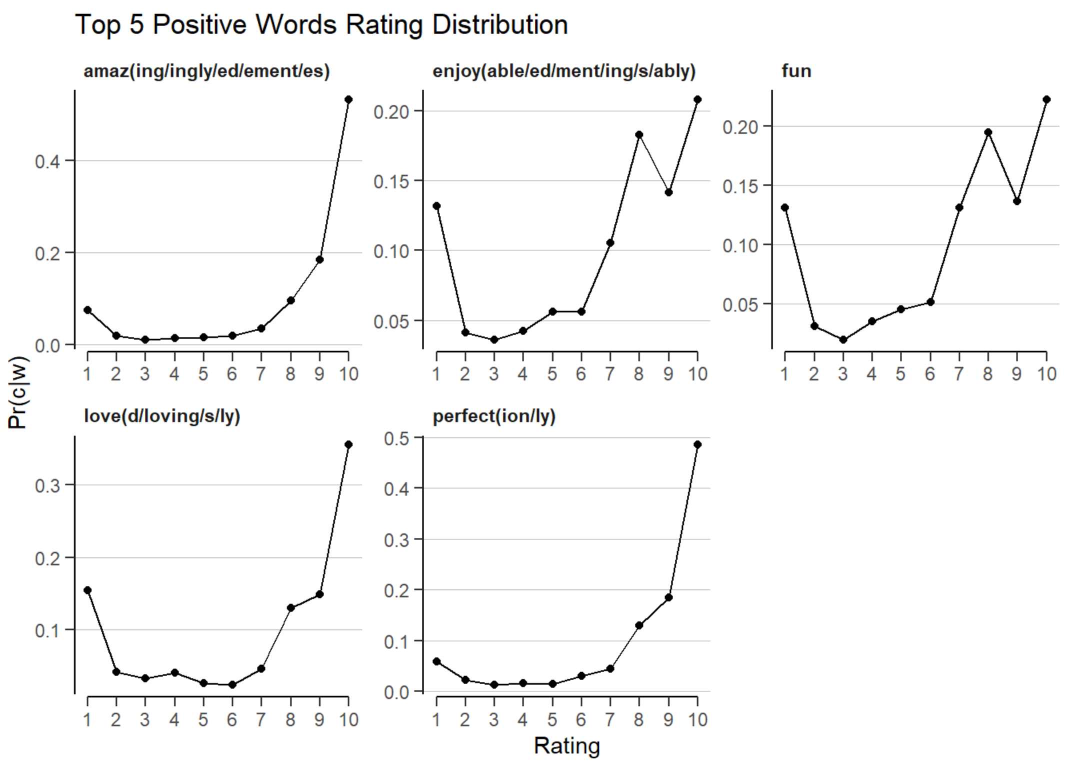
```
pos_5_ratings <- word_sentiment_rating %>%
  # Filter only rows that contain the top 5 stem words
  filter(stem_word %in% pos_5$stem_word) %>%
  # Count the number of occurences that a certain stem word has a certain rating
  count(stem_word, ratings) %>%
  # Arrange the tibble according to the total frequency of each stem word in decreasing manner
  arrange(factor(stem_word, levels = pos_5$stem_word)) %>%
  group_by(stem_word) %>%
  # Add "percentage" column which represents the probability that a review containing the denoted stem word
receives a certain rating
  mutate(percentage=n/sum(n)) %>%
  ungroup %>%
  # Add "word_suffixes" column by left joining the tibble with "pos_5$word_suffixes" column
  left_join(bind_cols("stem_word"=pos_5$stem_word, "word_suffixes"=pos_5$word_suffixes), by="stem_word") %>%
  # Add "value" column whose values come from the concatenation of "stem_word" column and "word_suffixes" co
lumn
  bind_cols(as_tibble(ifelse(!is.na(.$word_suffixes), str_c(.$stem_word, "(", .$word_suffixes, ") "), .$stem
_word))) %>%
  # Select only the following columns
  select(word=value, ratings, n, percentage)

neg_5_ratings <- word_sentiment_rating %>%
  select(stem_word, ratings) %>%
  filter(stem_word %in% neg_5$stem_word) %>%
  count(stem_word, ratings) %>%
  arrange(factor(stem_word, levels = neg_5$stem_word)) %>%
  group_by(stem_word) %>%
  mutate(percentage=n/sum(n)) %>%
  ungroup %>%
  left_join(bind_cols("stem_word"=neg_5$stem_word, "word_suffixes"=neg_5$word_suffixes), by="stem_word") %>%
  bind_cols(as_tibble(ifelse(!is.na(.$word_suffixes), str_c(.$stem_word, "(", .$word_suffixes, ") "), .$stem
_word))) %>%
  select(word=value, ratings, n, percentage)

# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(word_sentiment_rating, pos_5, neg_5) # run this code for environment data clarity
```

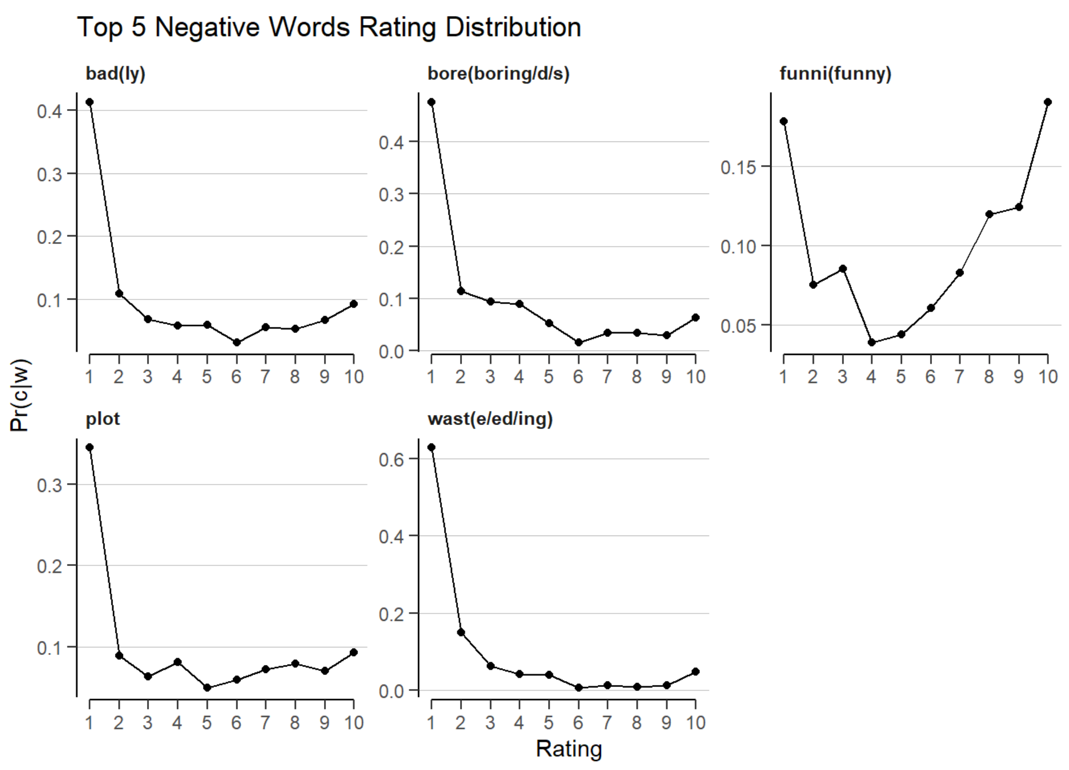
Plot the rating distribution for each top 5 stem word of positive sentiment using ggplot.

```
ggplot(pos_5_ratings, aes(ratings, percentage)) +
  # Create multipanel plots grouped by "word"
  # and free the scales on x and y axis such that the axis lines can appear on each plot
  facet_wrap(~ word, scales="free") +
  # Add points and lines with coord(ratings, percentage)
  geom_point() + geom_line() +
  # Add a capped axis for the x axis
  coord_capped_cart(bottom="both", left="none") +
  # Specify the x-axis tick marks to be 1, 2, ..., 10
  scale_x_continuous(breaks = seq_len(10)) +
  # Add main title, x axis, and y axis labels
  labs(title="Top 5 Positive Words Rating Distribution", x="Rating", y="Pr(c|w)") +
  theme(# Strip the background of the facet labellers
        strip.background = element_rect(fill = NA),
        # Remove the panel grid for the x axis
        panel.grid.major.x = element_blank(),
        # Add the panel grid for the x axis with the specified size and color
        panel.grid.major.y = element_line(size=.1, color="grey75" ),
        # Strip the background of the coordinate panels
        panel.background = element_rect(fill = NA),
        # Remove the panel border
        panel.border = element_blank(),
        # Make the axis line to be visible
        axis.line = element_line(),
        # Elongate the axis ticks to the following length
        axis.ticks.length = unit(.15, "cm"),
        # Make the facet labellers to be bold and left aligned
        strip.text = element_text(face="bold", hjust=0))
```



Similarly, for the rating distribution of top 5 stem words with negative sentiments, we can plot it this way.

```
ggplot(neg_5_ratings, aes(ratings, percentage)) +
  facet_wrap(~ word, scales="free") +
  geom_point() + geom_line() +
  coord_capped_cart(bottom="both", left="none") +
  scale_x_continuous(breaks = seq_len(10)) +
  labs(title="Top 5 Negative Words Rating Distribution", x="Rating", y="Pr(c|w)") +
  theme(strip.background = element_rect(fill = NA),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(size=.1, color="grey75" ),
        panel.background = element_rect(fill = NA),
        panel.border = element_blank(),
        axis.line = element_line(),
        axis.ticks.length = unit(.15, "cm"),
        strip.text = element_text(face="bold", hjust=0))
```



```
# These data will not be used anymore for future analysis, so we can safely remove them.
#rm(pos_5_ratings, neg_5_ratings) # run this code for environment data clarity
```

The rating distribution shown on the two multipanel plots above strengthen the premise of correlation between sentiments of word choice and ratings left by the reviewers.

We can see that a review containing any of the top 5 positive words receiving the highest rating (i.e. 10) has the highest probability compared to other ratings. Likewise, the highest probability that a review containing any of the top 5 positive words receiving the lowest rating has the highest probability compared to other ratings.

However, for the positive words, the probability of getting a certain ratings are not decreasing as the rating gets lower. In fact, the lowest probability of the review gets a certain rating goes to rating 3. After that, the probability of the positive word gets a rating of 2 is higher than 3, and similarly, the probability is higher for rating 1 than 2. We can also see the same thing happen to the negative words rating distribution where the lowest probability goes to a rating of around 4 to 6 instead of the expected 10.

Assuming that the sentiment from the lexicon is correct, this event perhaps happens thanks to the fact that the sentiments are only obtained per word. Sometimes, sentiment-associated words can be preceded by a negating word such as “never”, “no”, “not”, or “without”- which completely change the context of the word.

Also, since the movie ratings reflect the reviewers’ experience while watching the movie, they are often more compelled to leave a review when they have a strong opinion about the movie. So, the high probability on both ends of the ratings might also come from the imbalance of the rating distribution itself.