Untitled

Group 2

2022-11-14

Environment

```
##
## 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

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

Before making plots

```
#Find word counts
book_words <- IMDB %>%
  unnest_tokens(word, review) %>%
  anti_join(stop_words)%>%
  count(docs, word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
total_words <- book_words %>%
  group_by(docs) %>%
  summarize(total = sum(n))
book_words_new <- left_join(book_words, total_words)
```

```
## Joining, by = "docs"
```

```
#Take a brief look at the words
brief_word <- book_words_new%>%count(word)
arrange(brief_word,desc(n))
```

```
## # A tibble: 118,943 x 2
##    word       n
##    <chr>  <int>
##  1 movie  30424
##  2 br     29202
##  3 film   27522
##  4 time   17291
##  5 story  14816
##  6 people 12150
##  7 bad    11778
##  8 watch  10889
##  9 movies 10722
## 10 acting 10680
## # ... with 118,933 more rows
```

This part shows the word that appears more frequently in statistics. The number of words"movie", "film","bar" are the largest in the chart, but they are fuzzy words that are meaningless for classification, so we need to delete them to make the data clean.

```
#Add more "stop_words" and clean data
stop_words <- rbind(stop_words,c("movie","SMART"),c("br","SMART"),c("film","SMART"),c("time","SMART"),c
#Do it again and tidy the data
book_words <- IMDB %>%
  unnest_tokens(word, review) %>%
  anti_join(stop_words)%>%
  count(docs, word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
total_words <- book_words %>%
  group_by(docs) %>%
  summarize(total = sum(n))
book_words_new <- left_join(book_words, total_words)
```

```
## Joining, by = "docs"
```

```
book_words$docs <- as.character(book_words$docs)
book_words <- tibble(book_words)
#Cast a one-token-per-row table
IMDB_dtm <- book_words %>%
  cast_dtm(docs, word, n)
IMDB_dtm
```

```
## <<DocumentTermMatrix (documents: 50000, terms: 118926)>>
## Non-/sparse entries: 3471499/5942828501
## Sparsity           : 100%
## Maximal term length: 72
## Weighting          : term frequency (tf)
```

LDA

```
#Use LDA function to create a 10 topic model
IMDB_lda <- LDA(IMDB_dtm, k = 10, control = list(seed = 1234))
IMDB_lda
```

```
## A LDA_VEM topic model with 10 topics.
```

```
#Examine per-topic-per-word probabilities
IMDB_topics <- tidy(IMDB_lda, matrix = "beta")
IMDB_topics
```

```
## # A tibble: 1,189,260 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 marty 0.0000100
##  2     2 marty 0.00000404
##  3     3 marty 0.00000156
##  4     4 marty 0.000000399
##  5     5 marty 0.0000312
##  6     6 marty 0.00000403
##  7     7 marty 0.00000466
##  8     8 marty 0.000000770
##  9     9 marty 0.00000534
## 10    10 marty 0.000334
## # ... with 1,189,250 more rows
```

The column beta represents the probability of that term being generated from that topic for that document. It is the probability of that term belonging to that topic. And the value of beta are generally low in our data.

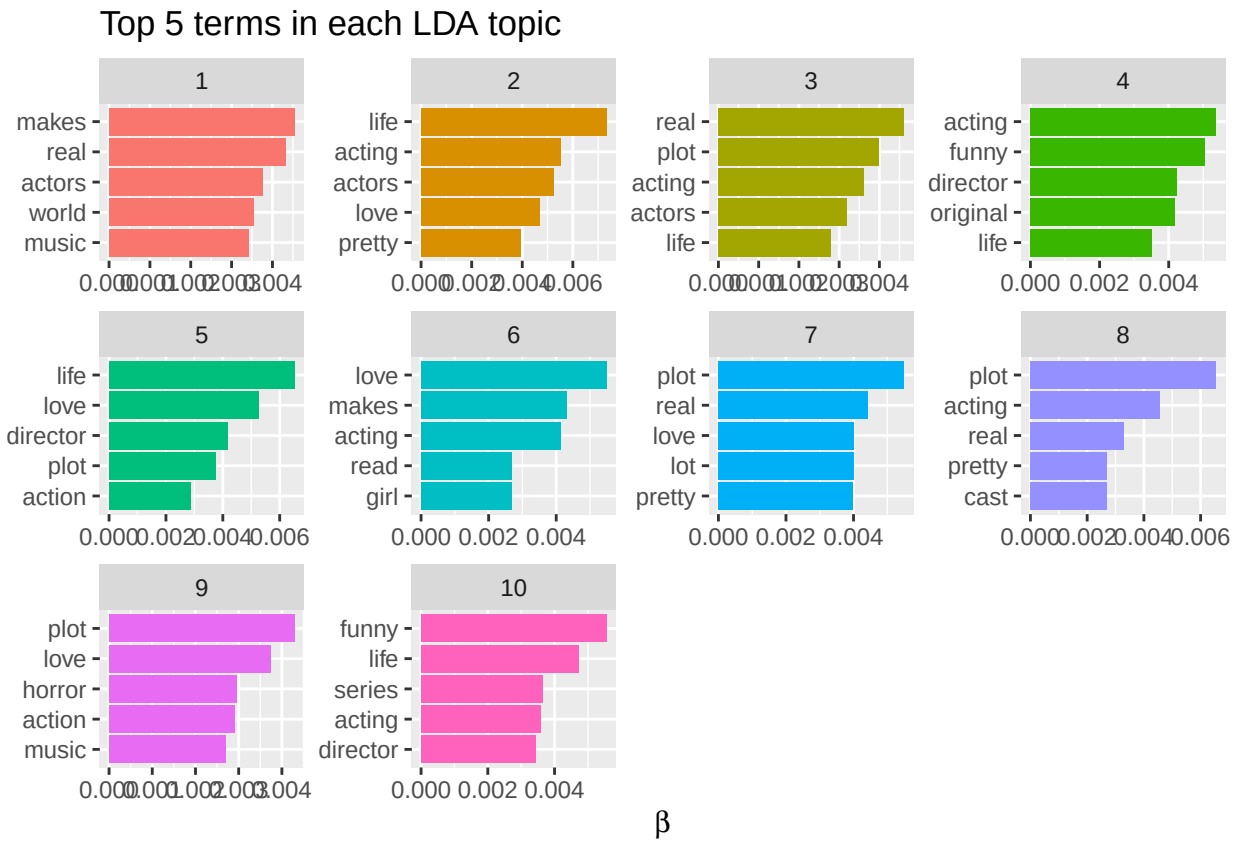
```
#Find the top 5 terms within each docs
IMDB_top_terms <- IMDB_topics %>%
group_by(topic) %>%
slice_max(beta, n = 5) %>%
ungroup() %>%
arrange(topic, -beta)
IMDB_top_terms
```

```
## # A tibble: 50 x 3
##    topic term      beta
##    <int> <chr>    <dbl>
##  1     1 makes  0.00455
##  2     1 real   0.00432
```

```
##  3     1 actors 0.00375
##  4     1 world  0.00354
##  5     1 music  0.00341
##  6     2 life   0.00734
##  7     2 acting 0.00550
##  8     2 actors 0.00523
##  9     2 love   0.00470
## 10     2 pretty 0.00393
## # ... with 40 more rows
```

This table shows the five most occurring terms for each topic and their beta values accordingly.

```
#Make a ggplot2 visualization
IMDB_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()+
  labs(title = "Top 5 terms in each LDA topic",
       x = expression(beta), y = NULL)
```



This part examine which topics are associated with which description fields and we use gamma to represent the probability that each document belongs in each topic. Probabilities vary a lot in the data frame, and our model has assigned a probability to each description belonging to each topic we constructed from the phrases.

```
#Examine the probability each document belongs in each topic
IMDB_lda_gamma <- tidy(IMDB_lda,matrix="gamma")
IMDB_lda_gamma
```

```
## # A tibble: 500,000 x 3
##    document topic  gamma
##    <chr>    <int>  <dbl>
##  1 5816         1 0.0881
##  2 39875        1 0.0857
##  3 5709         1 0.119
##  4 12648        1 0.0950
##  5 40379        1 0.0899
##  6 49130        1 0.0893
##  7 32593        1 0.0944
##  8 41762        1 0.0934
##  9 17610        1 0.161
## 10 10115        1 0.0982
## # ... with 499,990 more rows
```

This part examine which topics are associated with which description fields and we use gamma to represent the probability that each document belongs in each topic. Probabilities vary a lot in the data frame, and our model has assigned a probability to each description belonging to each topic we constructed from the phrases.

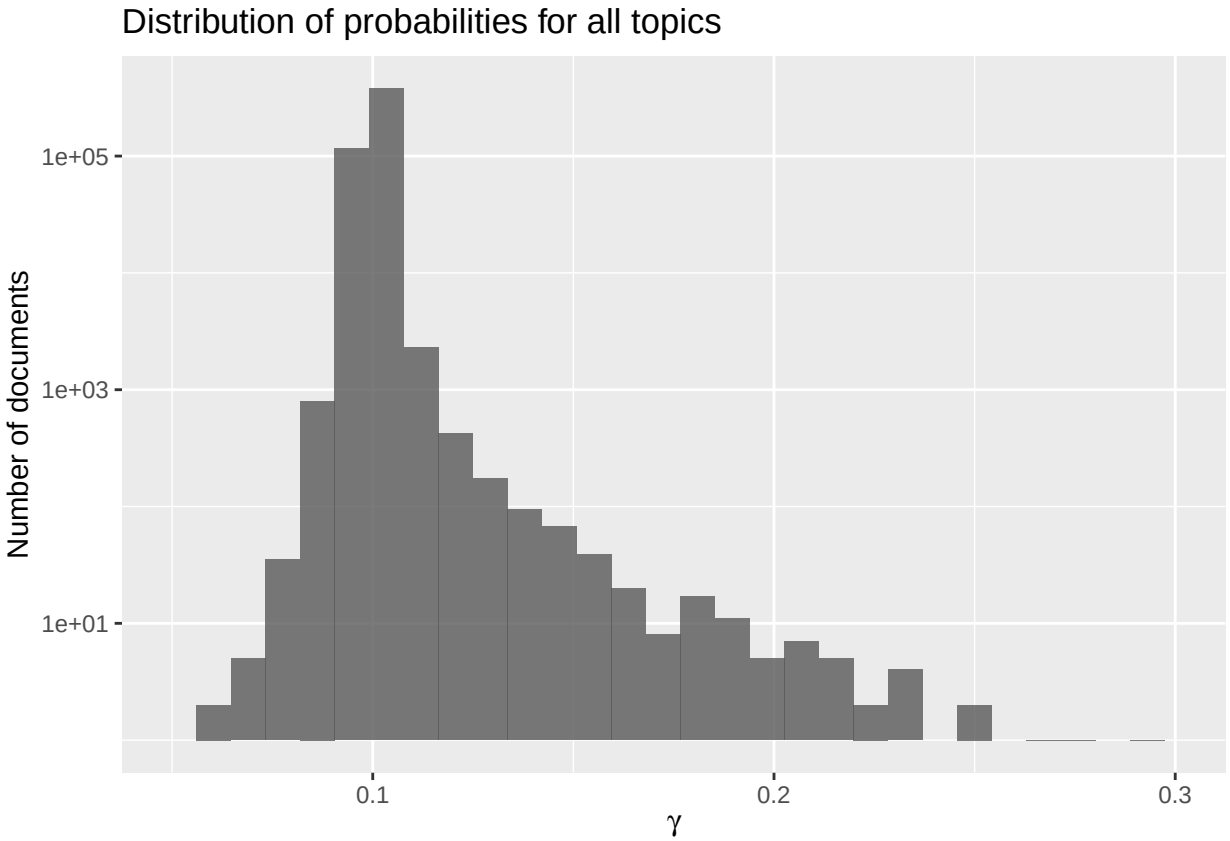
```
#Distribution of probablities for all topics
ggplot(IMDB_lda_gamma, aes(gamma)) +
  geom_histogram(alpha = 0.8) +
  scale_y_log10() +
  labs(title = "Distribution of probabilities for all topics",
       y = "Number of documents", x = expression(gamma))+
  xlim(0.05,0.3)
```

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

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 5 rows containing missing values (geom_bar).
```



This distribution shows that documents are being well discriminated as belonging to a topic or not. We can also look at how the probabilities are distributed within each topic. Notice that gamma runs near zero, which means there are many documents that do not belong in each topic and we need to change the number of topic we chose.

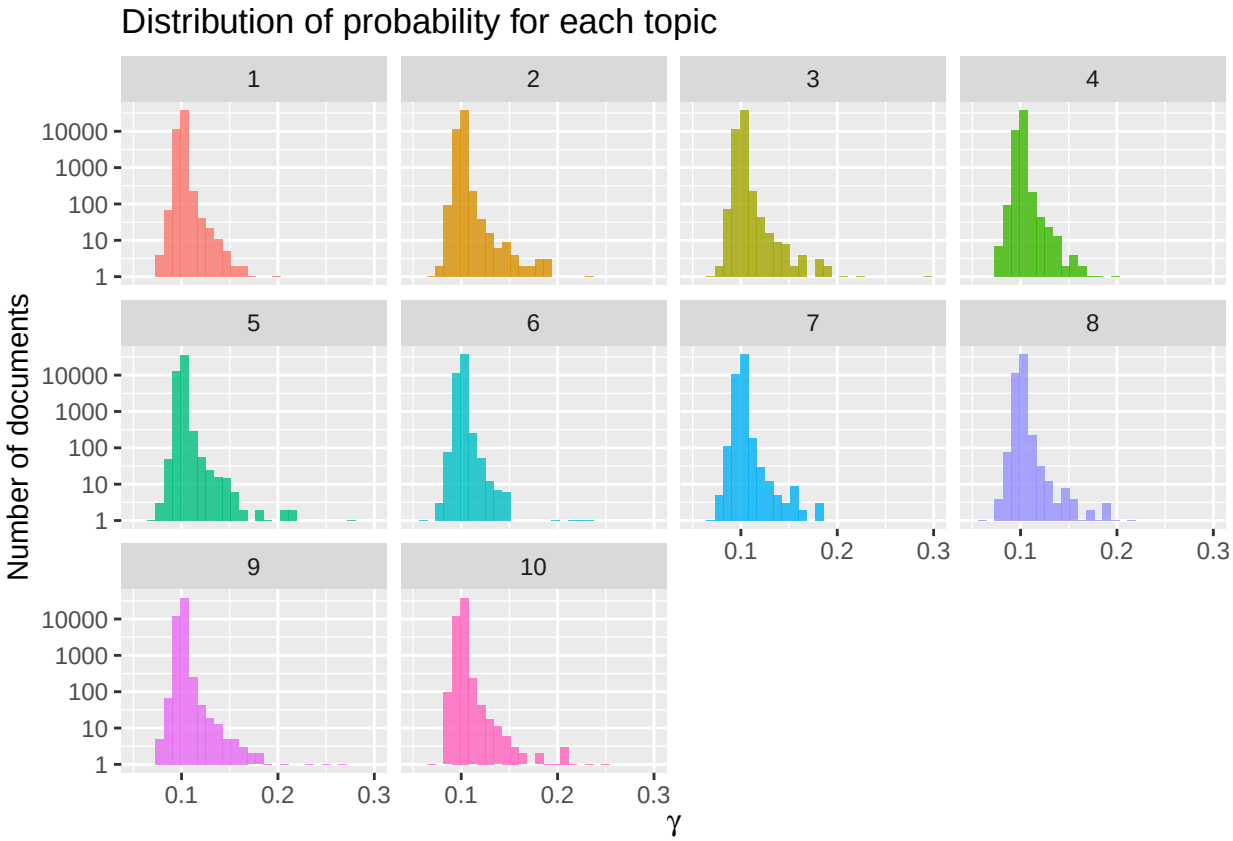
```
#Probability for each topic
ggplot(IMDB_lda_gamma, aes(gamma, fill = as.factor(topic))) +
  geom_histogram(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 4) +
  scale_y_log10() +
  labs(title = "Distribution of probability for each topic",
       y = "Number of documents", x = expression(gamma))+
  xlim(0.05,0.3)
```

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

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 143 rows containing missing values (geom_bar).
```



From these charts, we can see that the magnitude of the values in each topic is far from 1 and mostly concentrated around 0. This is because that in the processing of cleaning up the data, we removing the terms "movie", "actor",which appear most frequently in the ten topics, in order to make the data cleaner. ## Pair words

```
#Count how many times each pair od words occur together
IMDB_title <- book_words[,-3]
IMDB_title <- tibble(IMDB_title)
IMDB_title <- IMDB_title%>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

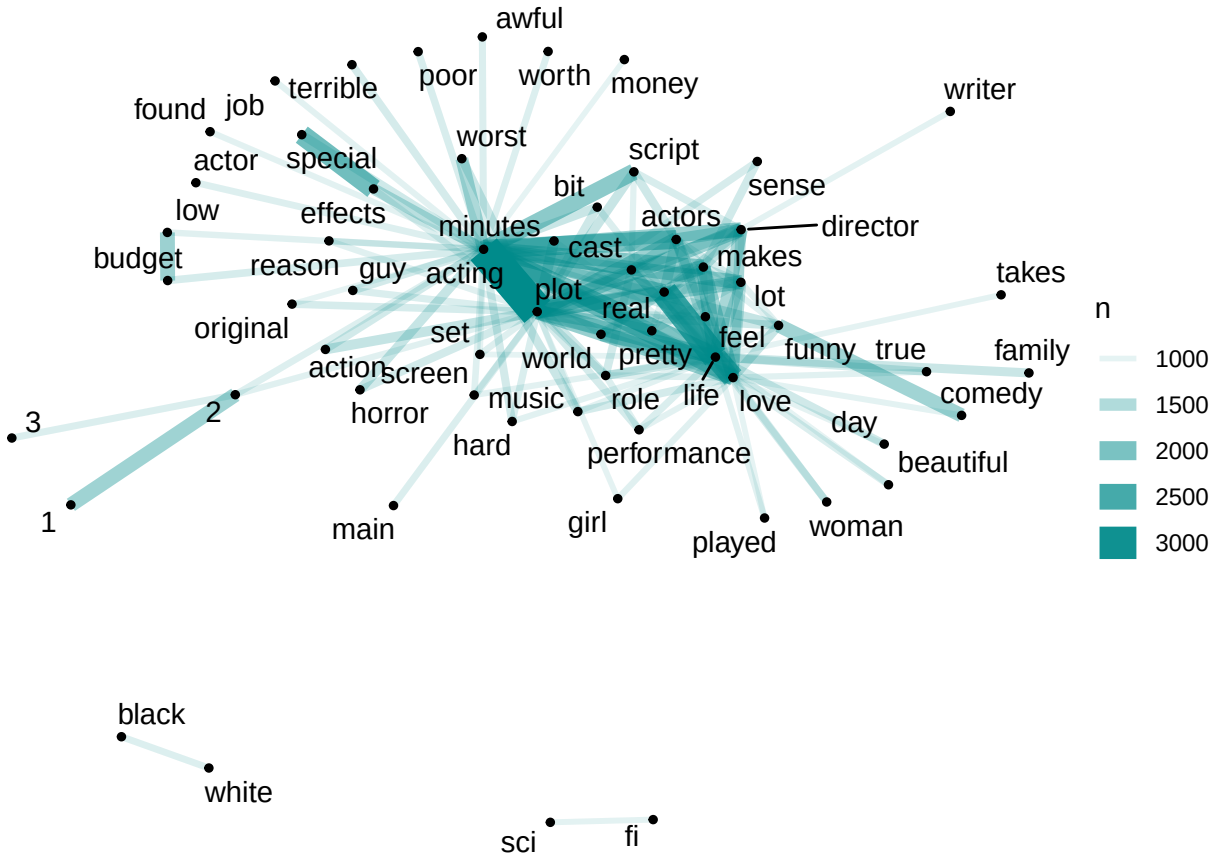
```
title_word_pairs <-IMDB_title %>%
  pairwise_count(word, docs, sort = TRUE, upper = FALSE)
title_word_pairs
```

```
## # A tibble: 73,077,521 x 3
##    item1   item2       n
##    <chr>   <chr>   <dbl>
##  1 plot    acting   3130
##  2 life    real     2525
##  3 love    life     2231
##  4 special effects  2080
##  5 actors  acting   1990
##  6 life    acting   1967
##  7 acting  script   1827
```

```
##  8 love    acting   1786
##  9 real    acting   1743
## 10 love    plot     1725
## # ... with 73,077,511 more rows
```

This chart shows the number of the two most frequently occurring words after deleting the meaningless terms.

```
#Plot networks of these words
set.seed(1234)
title_word_pairs %>%
  filter(n >= 1000) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4") +
  geom_node_point(size = 1) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```



This chart is the visualization of the pairs of words that occur together most often in description fields. This plot displays ”minutes”,”acting”,”plot”,”real” tends to appear more often together. ##Calculating tf-idf

```
#tf_idf
IMDB_tf_idf <- IMDB_title %>%
  count(docs, word, sort = TRUE) %>%
```

```
  bind_tf_idf(word, docs, n)
IMDB_tf_idf %>%
  arrange(-tf_idf)
```

```
## # A tibble: 3,471,499 x 6
##    docs  word          n    tf   idf tf_idf
##    <chr> <chr>     <int> <dbl> <dbl>  <dbl>
##  1 24941 tired         1 1      4.23   4.23
##  2 36845 stop.oz       1 0.333 10.8    3.61
##  3 13316 ernest        1 0.5    6.38   3.19
##  4 17588 book          1 1      2.91   2.91
##  5 30996 written       1 1      2.87   2.87
##  6 9452  sandra        1 0.5    5.69   2.84
##  7 40818 unites        1 0.333  8.42   2.81
##  8 31762 bardwork      1 0.25  10.8    2.70
##  9 43693 outtakes      1 0.333  7.16   2.39
## 10 1418  bad.trust     1 0.2   10.8    2.16
## # ... with 3,471,489 more rows
```

These are the most improtant words in IMDB measured by tf-idf.