HW_MOVIE

Ruihang Han

```r
# Load necessary libraries
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(tm)
```

```
Loading required package: NLP

Attaching package: 'NLP'

The following object is masked from 'package:ggplot2':

    annotate
```

```r
library(topicmodels)
library(ldatuning)
```

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

```r
library(tidytext)
library(Rtsne)
```

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

```r
library(ggplot2)
library(wordcloud)
```

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

```
Loading required package: RColorBrewer
```

```r
library(RColorBrewer)
```

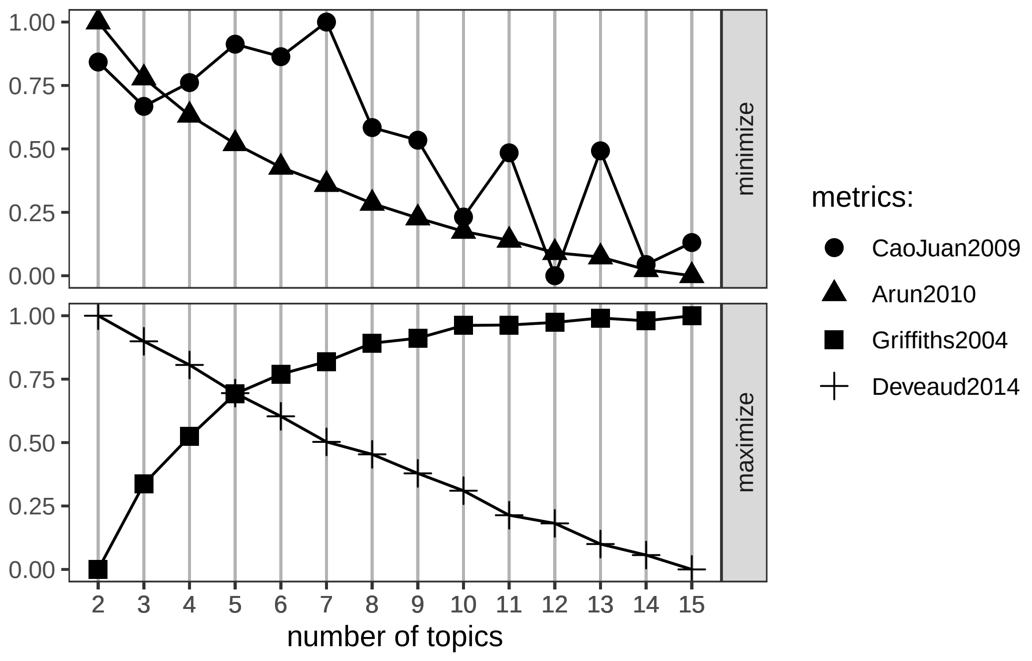
```r
# Load and preprocess the movie plot data
movie_data <- read_csv("movie_plots.csv", col_types = cols())

# Define custom stopwords list with additional words
custom_stopwords <- c(stopwords("en"), "will", "would", "can", "could", "may", "might", "mus


# Create a text corpus from movie plots
corpus <- VCorpus(VectorSource(movie_data$Plot))

# Preprocess the text: lowercase, remove punctuation, remove stopwords, and trim whitespace
corpus <- corpus %>%
    tm_map(content_transformer(tolower)) %>%
    tm_map(removePunctuation) %>%
    tm_map(removeWords, custom_stopwords) %>%
    tm_map(stripWhitespace)

# Generate a Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)
```

```r
# Set up parameters and run the topic modeling evaluation
topic_evaluation <- FindTopicsNumber(
    dtm,
    topics = seq(2, 15, by = 1),
    metrics = c("CaoJuan2009", "Arun2010", "Griffiths2004", "Deveaud2014"),
    method = "Gibbs",
    control = list(seed = 1234),
    mc.cores = 1L,
    verbose = TRUE
)
```

```
fit models... done.
calculate metrics:
  CaoJuan2009... done.
  Arun2010... done.
  Griffiths2004... done.
  Deveaud2014... done.
```

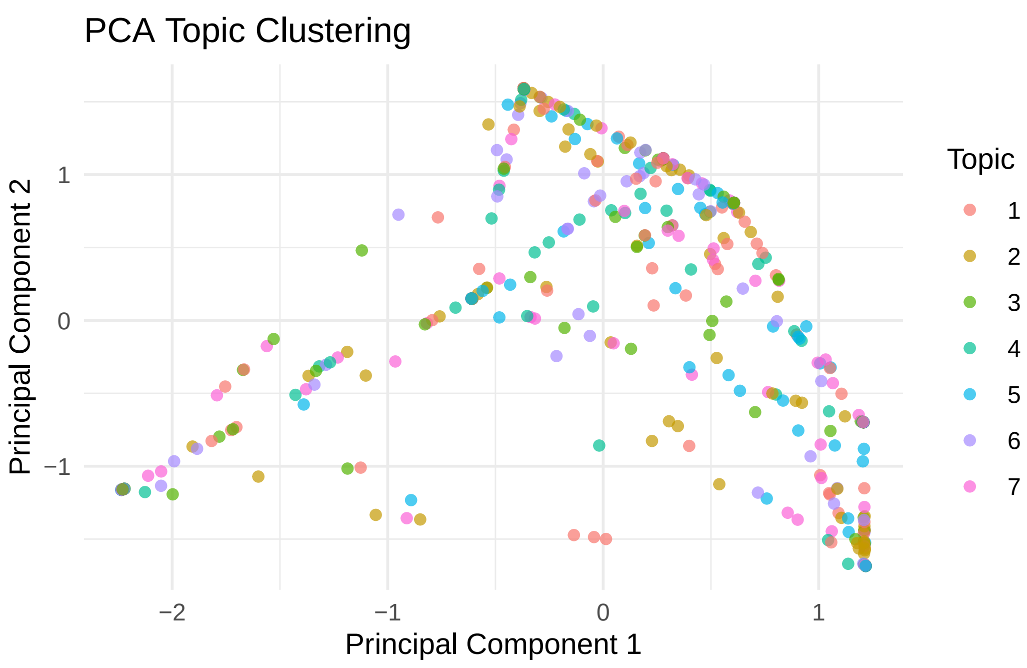
```
# Plot the results of the topic number evaluation
FindTopicsNumber_plot(topic_evaluation)
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
i The deprecated feature was likely used in the ldatuning package.
  Please report the issue at <https://github.com/nikita-moor/ldatuning/issues>.
```



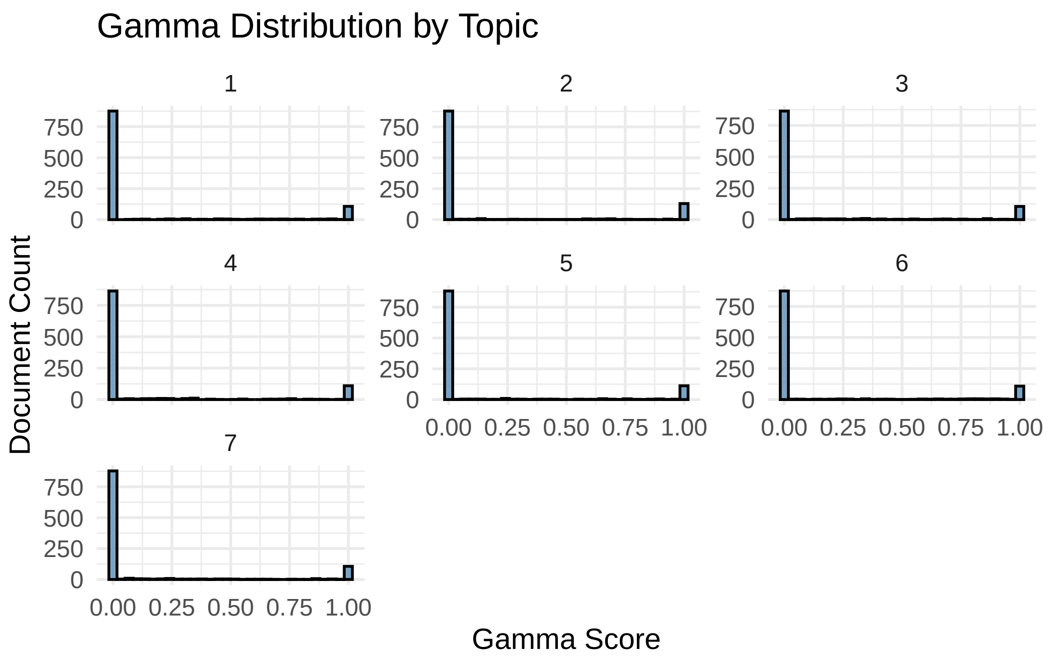
This plot presents an analysis of the optimal number of topics for a topic model, using four evaluation metrics: **CaoJuan2009**, **Arun2010**, **Griffiths2004**, and **Deveaud2014**. Each metric provides guidance on the ideal topic count, with CaoJuan2009 and Arun2010 intended to be minimized and Griffiths2004 and Deveaud2014 to be maximized. The results suggest that the optimal number of topics is around **6**. This is supported by the Griffiths2004 and Deveaud2014 metrics, which both stabilize at higher values around 6 topics, indicating improved model performance at this level. While the CaoJuan2009 and Arun2010 metrics suggest fewer topics could be sufficient, the overall recommendation leans towards selecting 6 topics for a balanced model.

```r
# Set number of topics and run the LDA model
num_topics <- 7
lda_fit <- LDA(dtm, k = num_topics, control = list(seed = 1234))

# Obtain gamma matrix and apply PCA
gamma_vals <- posterior(lda_fit)$topics
pca_analysis <- prcomp(gamma_vals, center = TRUE, scale. = TRUE)
pca_data_frame <- as.data.frame(pca_analysis$x)

# Extract main topic for each document
doc_topics <- tidy(lda_fit, matrix = "gamma") %>%
    group_by(document) %>%
    slice_max(gamma, n = 1) %>%
    ungroup()

pca_data_frame$Topic <- factor(doc_topics$topic)

# Plot PCA results
ggplot(pca_data_frame, aes(x = PC1, y = PC2, color = Topic)) +
    geom_point(alpha = 0.7) +
    labs(title = "PCA Topic Clustering", x = "Principal Component 1", y = "Principal Compone
    theme_minimal()
```



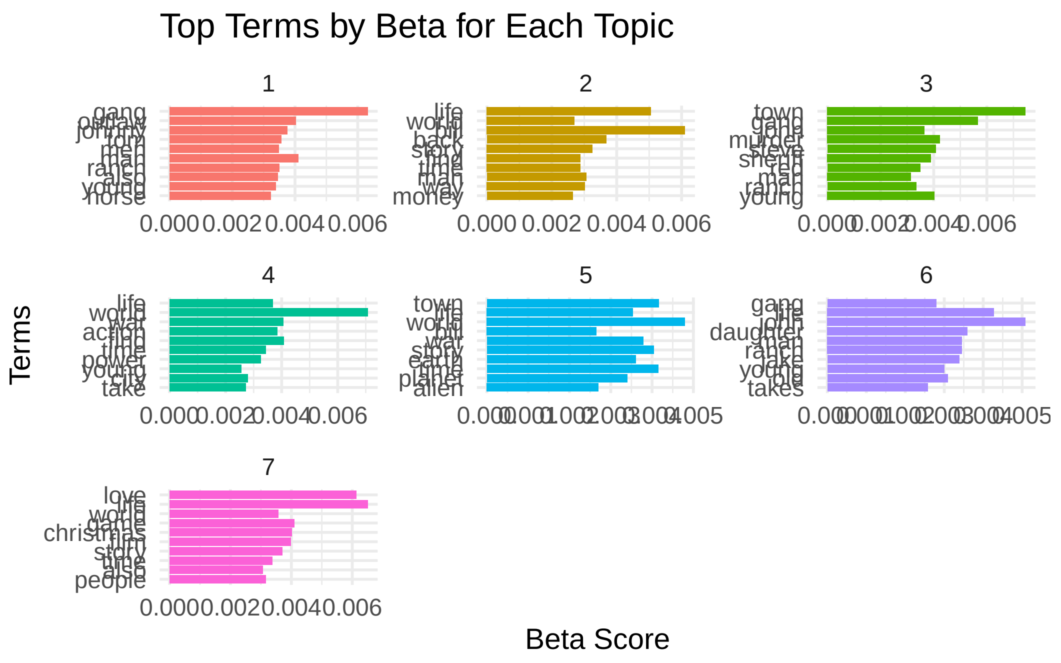
This PCA plot visualizes the clustering of topics from a topic modeling analysis, where each point represents a document positioned by the first two principal components based on its topic distribution. The colors indicate different topics (labeled 1 to 7), with documents of similar topic distributions appearing closer together in the PCA space. The clustering patterns show that certain topics are distinct, while others overlap, suggesting shared thematic elements. This visualization helps in assessing topic coherence and similarity, revealing that some topics have a more concentrated distribution while others are spread out, indicating greater variability across documents.

```r
# Generate a histogram for gamma values across topics
doc_gamma <- tidy(lda_fit, matrix = "gamma")

ggplot(doc_gamma, aes(x = gamma)) +
    geom_histogram(bins = 30, fill = "steelblue", color = "black", alpha = 0.7) +
    facet_wrap(~ topic, scales = "free_y") +
    labs(title = "Gamma Distribution by Topic", x = "Gamma Score", y = "Document Count") +
    theme_minimal()
```



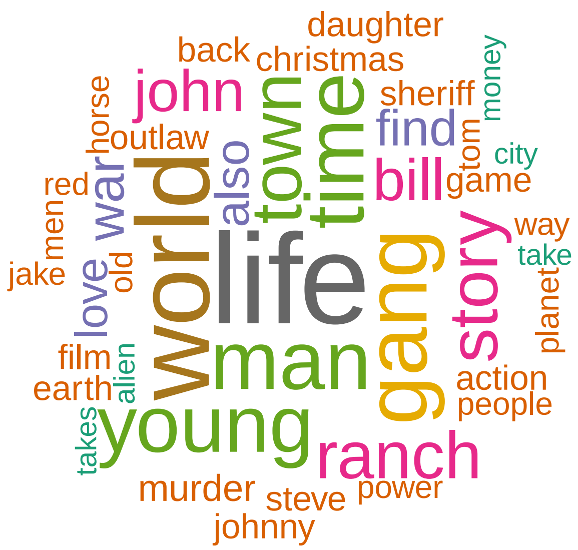
This plot shows the distribution of **gamma scores** (topic probabilities) for each of the seven topics across documents. Each subplot represents one topic, with the **X-axis** indicating the gamma score (ranging from 0 to 1) and the **Y-axis** representing the number of documents. The distribution reveals that for most topics, the majority of documents have a low gamma score, clustered around 0, with a smaller subset reaching a gamma score close to 1. This suggests that only a limited number of documents are strongly associated with each topic, while most documents have a weaker or more distributed topic association.

```r
# Extract and plot top terms by beta values for each topic
topic_beta <- tidy(lda_fit, matrix = "beta") %>%
    group_by(topic) %>%
    slice_max(beta, n = 10) %>%
    ungroup() %>%
    arrange(topic, -beta)

ggplot(topic_beta, aes(x = reorder(term, beta), y = beta, fill = factor(topic))) +
    geom_bar(stat = "identity", show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    coord_flip() +
    labs(title = "Top Terms by Beta for Each Topic", x = "Terms", y = "Beta Score") +
    theme_minimal()
```



This plot displays the **top terms by beta score** for each of the seven topics identified in a topic modeling analysis. Each panel represents a topic, with terms ranked by their **beta score** (the probability of the term appearing in that topic) along the X-axis. Higher beta scores indicate stronger associations of terms with the topic. For example, in Topic 1, terms like “gang,” “outlaw,” and “johnny” are highly relevant, while Topic 4 emphasizes words like “life,” “world,” and “war.” This visualization helps in understanding the defining terms for each topic, illustrating the unique vocabulary associated with different thematic groups across topics.

```
# Create a word cloud of top terms by aggregated beta score
all_term_freq <- topic_beta %>%
    group_by(term) %>%
    summarize(total_beta = sum(beta)) %>%
    arrange(desc(total_beta))

colors <- brewer.pal(8, "Dark2")  # Adjust the number of colors (e.g., 8) as needed

# Generate the word cloud
wordcloud(words = all_term_freq$term,
          freq = all_term_freq$total_beta,
          min.freq = 0.001,
          colors = colors,    # Use the defined color vector
          random.order = FALSE,
          rot.per = 0.35,
          scale = c(4, 0.5),
          main = "Aggregated Term Word Cloud")
```


daughter
back
christmas
money
john
town
time
horse
sheriff
find
outlaw
tom
city
also
red
war
world
bill
game
men
way
life
story
take
jake
old
love
gang
planet
film
man
alien
action
earth
people
young
takes
ranch
murder
power
steve
johnny