Topic Modeling

Set the library and the data

```r
library(tidyr)
library(topicmodels)
library(dplyr)
```

```
   'dplyr'

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.5
v ggplot2   3.5.1     v stringr   1.5.1
v lubridate 1.9.3     v tibble    3.2.1
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 becom
```

```r
library(tm)
```

```
     NLP

   'NLP'

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

    annotate
```

```r
library(wordcloud)
```

```
     RColorBrewer
```

```r
library(ggwordcloud)
library(tidytext)
library(textrank)

topicdata<- read.csv("C:/Users/16597/Downloads/movie_plots.csv")
head(topicdata)
```

```
                              Movie.Name
1                  Pioneers of the West
2                      The Infiltrators
3         "Graviton: The Ghost Particle"
4 Moses: Fallen. In the City of Angels.
5                       The Slave Trade
6                             The 303rd

1
2
3
4 Moses: Fallen. In the City of Angels.  :  A tale of a fallen angel who was sentenced to a l
5
6
```

Creating a corpus and clean it, then creating a document term matrix.

```r
topiccorp<- Corpus(VectorSource(topicdata$Plot))
topiccorp<- tm_map(topiccorp, content_transformer(tolower))
```

```
Warning in tm_map.SimpleCorpus(topiccorp, content_transformer(tolower)):
transformation drops documents
```

```r
topiccorp<- tm_map(topiccorp, removePunctuation)
```

```
Warning in tm_map.SimpleCorpus(topiccorp, removePunctuation): transformation
drops documents
```

```r
topiccorp<- tm_map(topiccorp, removeNumbers)
```

```
Warning in tm_map.SimpleCorpus(topiccorp, removeNumbers): transformation drops
documents
```

```r
topiccorp<- tm_map(topiccorp, removeWords, stopwords("english"))
```

```
Warning in tm_map.SimpleCorpus(topiccorp, removeWords, stopwords("english")):
transformation drops documents
```

```r
topiccorp<- tm_map(topiccorp, stripWhitespace)
```

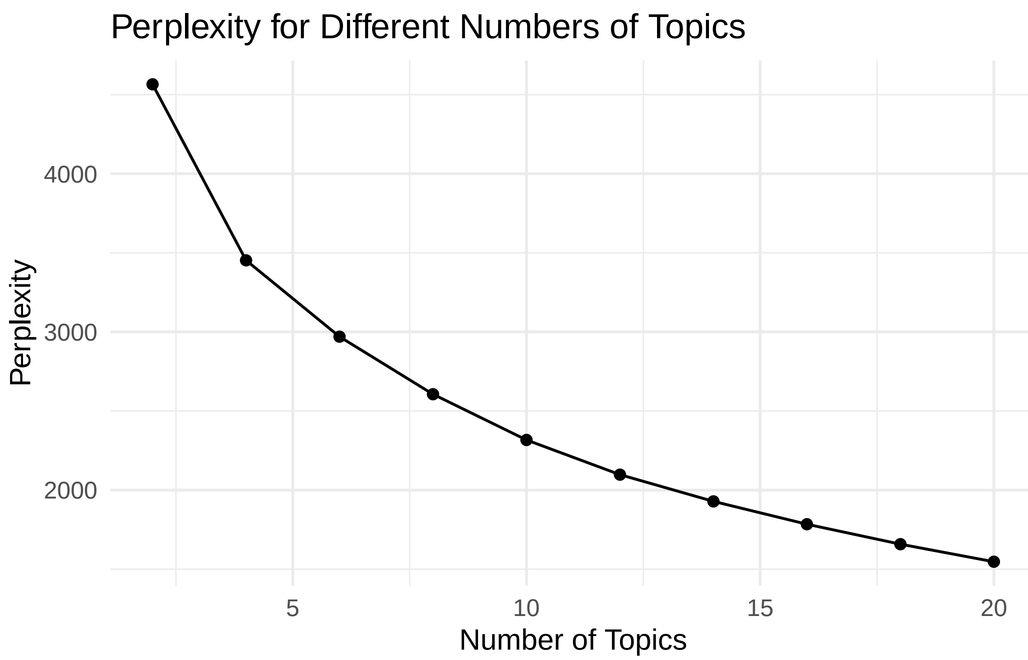
```
Warning in tm_map.SimpleCorpus(topiccorp, stripWhitespace): transformation
drops documents
```

```r
# Create the dtm
dtm <- DocumentTermMatrix(topiccorp, control = list(wordLengths = c(3, Inf)))
```

Next step is evaluating the topics, selecting the number of possible topics from the data.

```r
k_values<- seq(2, 20, by = 2)

perplexities<- numeric(length(k_values))

for (i in seq_along(k_values)) {
  k <- k_values[i]
```

```r
# Creating LDA model
  lda_model <- LDA(dtm, k = k, control = list(seed = 1234))

  perplexities[i] <- perplexity(lda_model, dtm)
}

perplexity_df <- data.frame(
  Topics = k_values,
  Perplexity = perplexities
)

# Plot the perplexity curve
ggplot(perplexity_df, aes(x = Topics, y = Perplexity)) +
  geom_line() +
  geom_point() +
  labs(title = "Perplexity for Different Numbers of Topics",
       x = "Number of Topics",
       y = "Perplexity") +
  theme_minimal()
```



Based on the perplexity plot, there is no sharp "elbow" indicating a clear inflection point, but there exists the trend. The perplexity decreases gradually as the number of topics increases, and it begins to flatten out slightly around 10–12 topics. This suggests that adding more topics beyond this point has diminishing returns in terms of perplexity reduction. Suppose there are 10 possible topics. My next step is fitting the final LDA model.

```r
# Set k (the number of topics) equal to 10
k<- 10

lda<- LDA(dtm, k = k, control = list(seed = 1234))
print(lda)
```

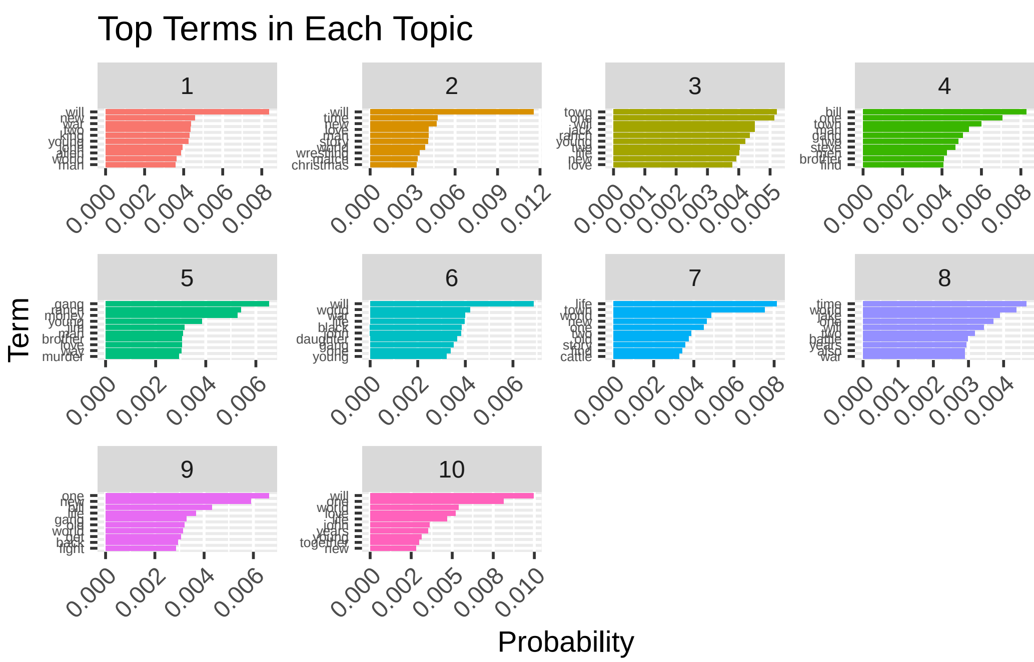
```
A LDA_VEM topic model with 10 topics.
```

```r
# Extract top terms of the topic

topics<- tidy(lda, matrix = "beta")

# Get the top 10 terms for each topic based on term probability within the topic
topterms<- topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

# Display the top terms for each topic
print(topterms)
```

```
# A tibble: 100 x 3
   topic term     beta
   <int> <chr>   <dbl>
 1     1 will  0.00836
 2     1 new   0.00457
 3     1 war   0.00436
 4     1 two   0.00433
 5     1 king  0.00429
 6     1 young 0.00425
 7     1 john  0.00394
 8     1 alien 0.00385
 9     1 world 0.00364
10     1 man   0.00357
# i 90 more rows
```

Before visualizing the word cloud, we first visualize the topterms.

```
topterms %>%
  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()+
  scale_x_continuous(labels = scales::number_format(accuracy = 0.001))+
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(size = 5))+
  labs(title = "Top Terms in Each Topic",
       x = "Probability",
       y = "Term")
```



In the mean time, I try to analyze the distribution of top topics, then I tried to build labels based on the distribution and the visualization.

```
# Get the gamma matrix to represent the topic distribution across documents
documenttopics<- tidy(lda, matrix = "gamma")

# Display the topic distribution for each document
print(documenttopics)
```

```
# A tibble: 10,770 x 3
   document topic    gamma
   <chr>    <int>    <dbl>
 1 1            1 0.000348
 2 2            1 0.994
 3 3            1 0.000752
 4 4            1 0.000166
 5 5            1 0.000565
 6 6            1 0.000583
 7 7            1 0.000421
 8 8            1 0.995
 9 9            1 0.000115
10 10           1 0.995
# i 10,760 more rows
```

```
# Label the topics

topic_keywords<- textrank_keywords(topterms %>% filter(topic == 3) %>% pull(term), ngram_max
print(topic_keywords)
```

```
$terms
[1] "one"  "new"  "will" "life"

$pagerank
$pagerank$vector
      town        one       will       jack      ranch      young        two
0.06438442 0.11619864 0.10934560 0.10579102 0.10428032 0.10428032 0.10579102
      life        new       love
0.10934560 0.11619864 0.06438442

$pagerank$value
[1] 1

$pagerank$options
NULL


$keywords
   keyword ngram freq
1 one-will     2    1
2     will     1    1
3 life-new     2    1
```

```
4       new     1    1

$keywords_by_ngram
   keyword ngram freq
1      one     1    1
2     will     1    1
3     life     1    1
4      new     1    1
5 one-will     2    1
6 life-new     2    1

attr(,"class")
[1] "textrank_keywords"
```

The first topic contains “king”, “world”, “man”, “war”, so the first label is related to War/Adventure.

The second topic contains “love”, “match”, “christmas”, so it can be Romance/Christmas.

The third topic has “town”, “two”, “life”, “ranch”, so the label can be Life and Love in the countryside.

Similarly, I can label other topics by their key words. For example, 4-Crime/Gang, 5-Money/Crime, 6-War and Family, 7-Life in Towns and Dramas, 8-World and Wars, 9-Gang Fights and Worlds, 10-Youth, Love and Life.

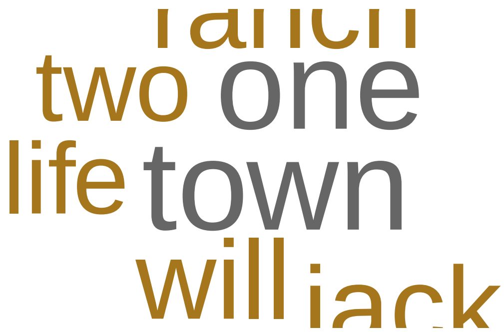
After analyzing the labels, it is easy to find that these movies are about love and life, wars, gangs and crimes.

Since I have the information I want, I can create the word clouds.

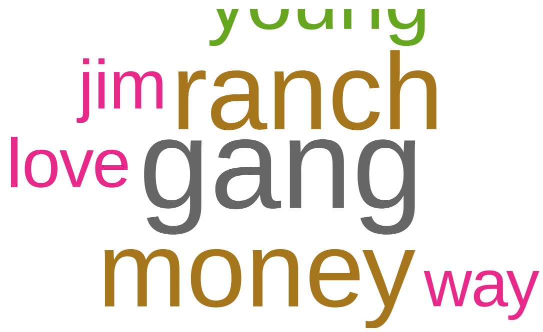
```
for(topic_num in unique(topterms$topic)){
  topic_words<-filter(topterms, topic == topic_num)

  wordcloud(
    words= topic_words$term,
    freq= topic_words$beta,
    max.words= 30,
    random.order= FALSE,
    colors= brewer.pal(8, "Dark2")
  )
}
```

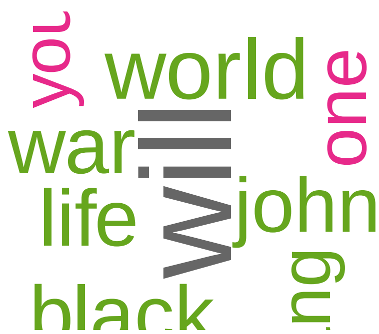

king
john
war
world
alien
will
man
new
two
young


wrestling
love
man
new
will
match
time
world
story


two
one
life
town
will

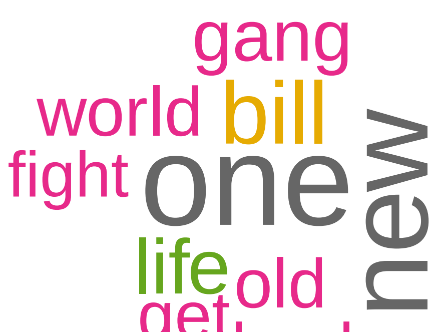

find
one
gang
bill
steve
two
man
men

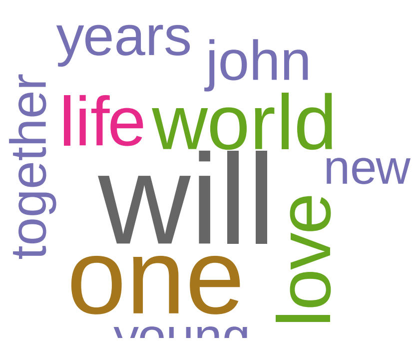

jim
ranch
love
gang
money
way


world
one
war
will
life
john


find
town
two
life
old
world
one


war
world
two
time
also
jake


gang
world
bill
fight
one
new
life
old


years
john
together
life
world
new
will
one
love