Modelling

LDA

```
# LDA is a process to assign for the text data, what is the probability of the topic related to each fi
```

Data: Associated Press

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.4.3
```

```
data("AssociatedPress")
AssociatedPress #data has gone through pre-processing step
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
ap_lda <- LDA(AssociatedPress, k=2, control = list(seed=1234))

library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.4.3
```

```
ap_topics <- tidy(ap_lda, matrix = "beta")
ap_topics
```

```
## # A tibble: 20,946 x 3
##    topic term           beta
##    <int> <chr>         <dbl>
##  1     1 aaron      1.69e-12
##  2     2 aaron      3.90e- 5
##  3     1 abandon    2.65e- 5
##  4     2 abandon    3.99e- 5
##  5     1 abandoned  1.39e- 4
##  6     2 abandoned  5.88e- 5
##  7     1 abandoning 2.45e-33
##  8     2 abandoning 2.34e- 5
##  9     1 abbott     2.13e- 6
## 10     2 abbott     2.97e- 5
## # i 20,936 more rows
```

Find terms that are most common within each topics

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
library(dplyr)
```

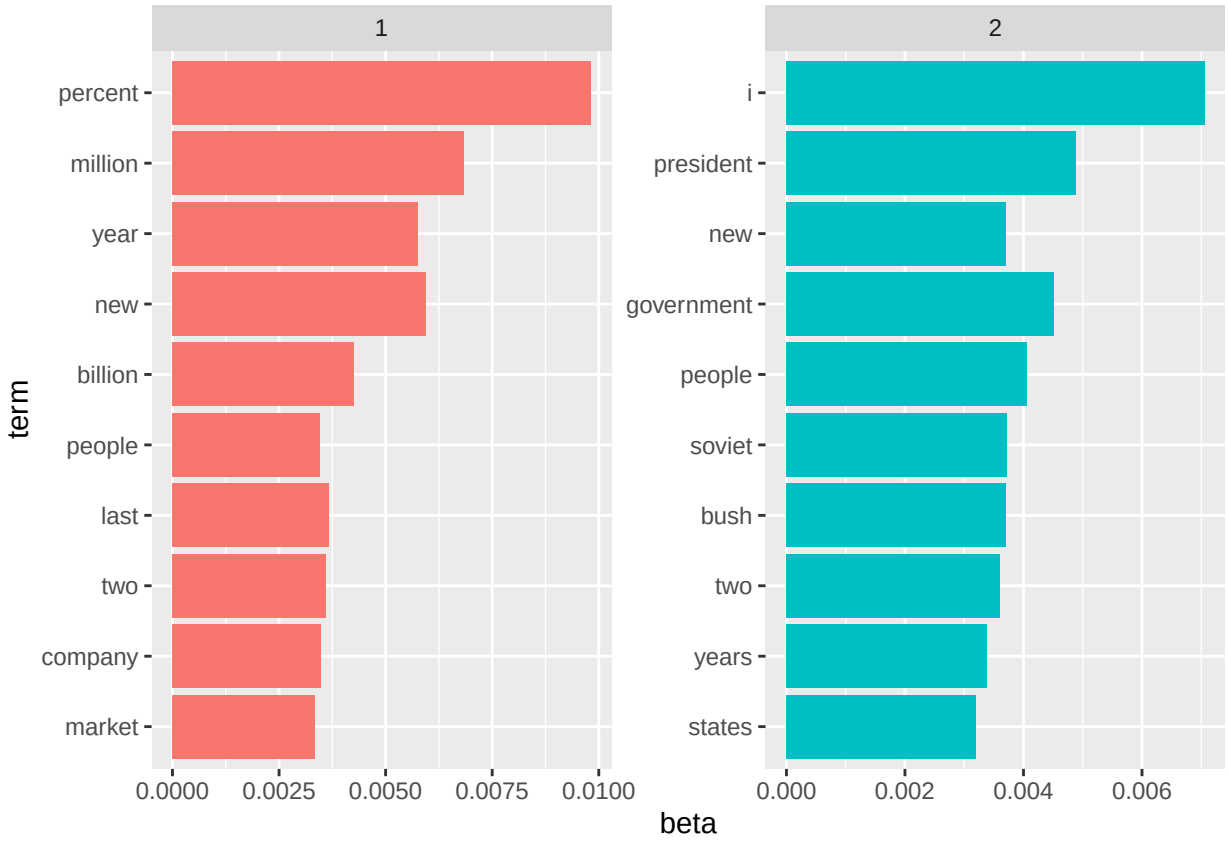
```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

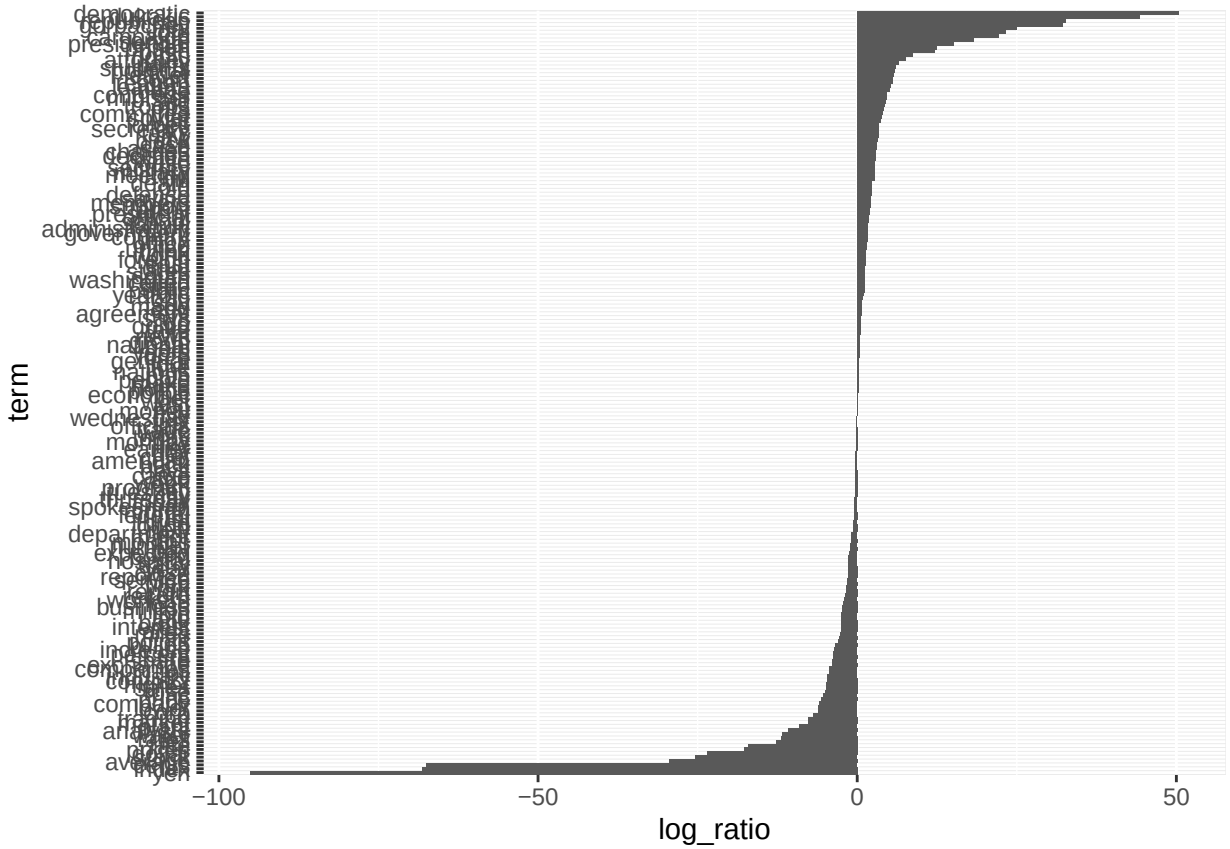
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



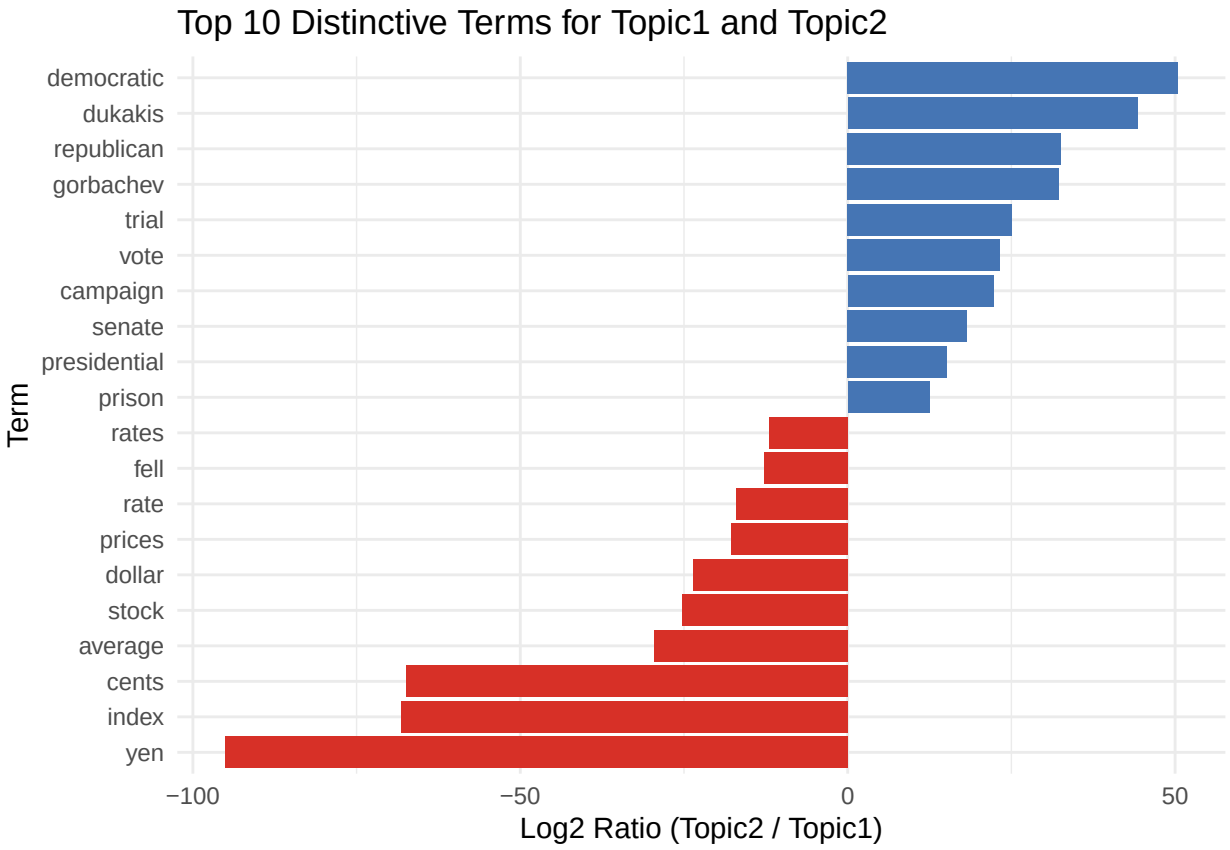
Looking into beta spread for each of the words

```
library(tidyr)

# Part 1
beta_spread <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  spread(topic, beta) %>%
  filter(topic1 > .001 | topic2 > .001) %>%
  mutate(log_ratio = log2(topic2 / topic1))

beta_spread %>% mutate(term=reorder(term,log_ratio)) %>%
  ggplot(aes(term,log_ratio))+geom_col(show.legend=FALSE)+coord_flip()
```




```
# Part 2
# Step 1: Reshape and calculate log ratio
beta_spread <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  pivot_wider(names_from = topic, values_from = beta) %>%
  filter(!is.na(topic1), !is.na(topic2)) %>%
  filter(topic1 > 0.001 | topic2 > 0.001) %>%
  mutate(log_ratio = log2(topic2 / topic1))

# Step 2: Select top 10 terms for each topic
top_topic1 <- beta_spread %>%
  slice_min(order_by = log_ratio, n = 10)

top_topic2 <- beta_spread %>%
  slice_max(order_by = log_ratio, n = 10)

top_terms <- bind_rows(top_topic1, top_topic2) %>%
  mutate(term = reorder(term, log_ratio))

# Step 3: Plot combined horizontal bar chart
ggplot(top_terms, aes(x = term, y = log_ratio, fill = log_ratio > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  labs(
    title = "Top 10 Distinctive Terms for Topic1 and Topic2",
    x = "Term",
```

```
    y = "Log2 Ratio (Topic2 / Topic1)"
  ) +
  scale_fill_manual(values = c("#d73027", "#4575b4")) +  # Red for topic1, blue for topic2
  theme_minimal()
```



Extract the per-document-per-topic-probabilities

```
ap_documents <- tidy(ap_lda, matrix = "gamma")
ap_documents
```

```
## # A tibble: 4,492 x 3
##    document topic    gamma
##       <int> <int>    <dbl>
##  1        1     1 0.248
##  2        2     1 0.362
##  3        3     1 0.527
##  4        4     1 0.357
##  5        5     1 0.181
##  6        6     1 0.000588
##  7        7     1 0.773
##  8        8     1 0.00445
##  9        9     1 0.967
## 10       10     1 0.147
## # i 4,482 more rows
```

Check the most common words in the document, eg document 6

```
tidy(AssociatedPress) %>% filter(document == 6) %>% arrange(desc(count))
```

```
## # A tibble: 287 x 3
##    document term           count
##       <int> <chr>          <dbl>
##  1        6 noriega           16
##  2        6 panama            12
##  3        6 jackson            6
##  4        6 powell             6
##  5        6 administration     5
##  6        6 economic           5
##  7        6 general            5
##  8        6 i                  5
##  9        6 panamanian         5
## 10        6 american           4
## # i 277 more rows
```

```
tidy(AssociatedPress) %>% filter(document == 2) %>% arrange(desc(count))
```

```
## # A tibble: 174 x 3
##    document term      count
##       <int> <chr>     <dbl>
##  1        2 peres        13
##  2        2 offer         9
##  3        2 official      8
##  4        2 bechtel       7
##  5        2 rappaport     7
##  6        2 israel        6
##  7        2 oil           6
##  8        2 memo          5
##  9        2 pipeline      5
## 10        2 company       4
## # i 164 more rows
```

Data: Movies

Load required libraries

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.4.2
```

```
## Loading required package: NLP
```

```
## Warning: package 'NLP' was built under R version 4.4.2
```

```
##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(topicmodels)
library(tidytext)
library(ggplot2)
library(dplyr)
library(tidyr)
```

1. Read documents and preprocess

```
mytext <- DirSource("TextMining")  # Folder must contain .txt files
mycorpus <- VCorpus(mytext) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)
```

2. Create document-term matrix (DTM)

```
dtm <- DocumentTermMatrix(mycorpus)
```

3. Remove empty documents

```
dtm <- dtm[slam::row_sums(dtm) > 0, ]
```

4. Fit LDA model with 3 topics

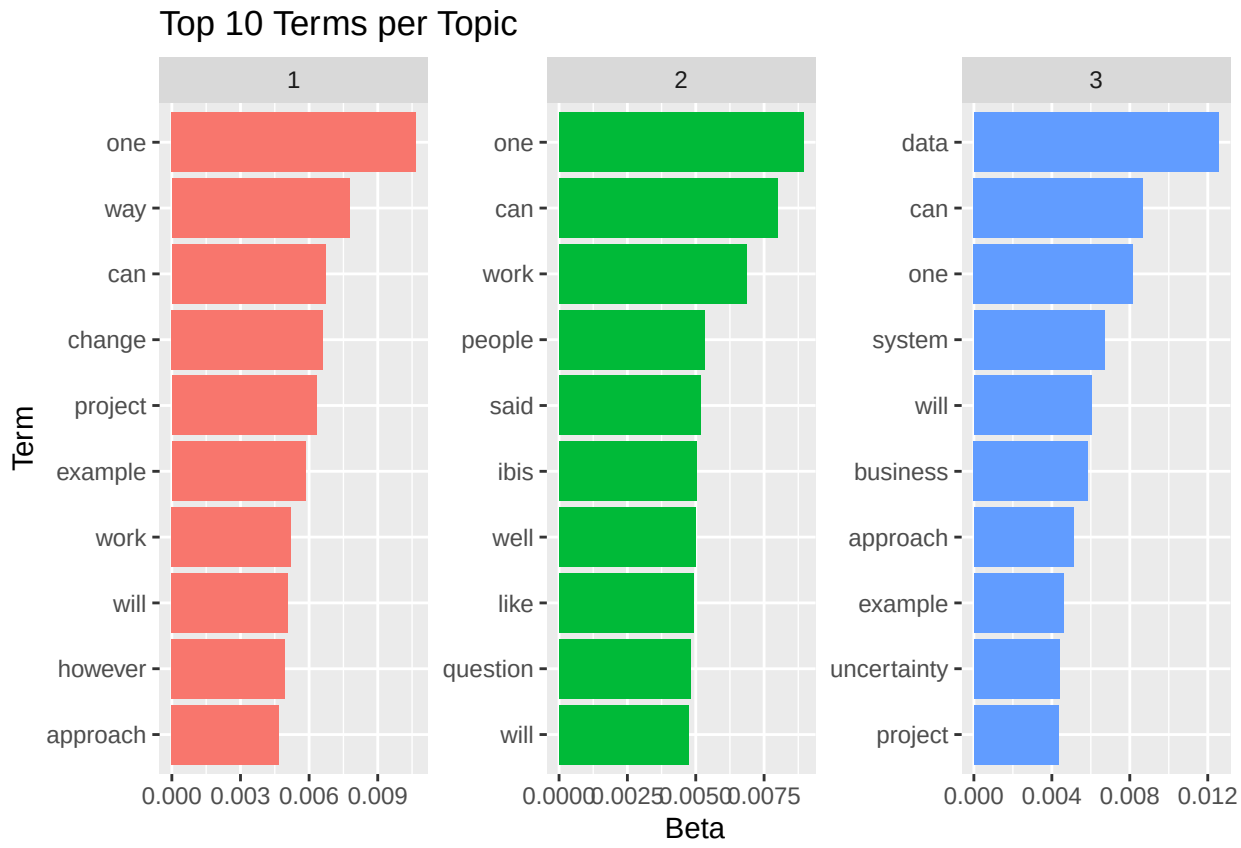
```
ap_lda <- LDA(dtm, k = 3, control = list(seed = 1234))
```

5. Extract topic-term probabilities (beta)

```
ap_topics <- tidy(ap_lda, matrix = "beta")
```

6. Plot top 10 terms per topic

```r
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  mutate(term = reorder_within(term, beta, topic))

ggplot(ap_top_terms, aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip() +
  scale_x_reordered() +
  labs(title = "Top 10 Terms per Topic", x = "Term", y = "Beta")
```

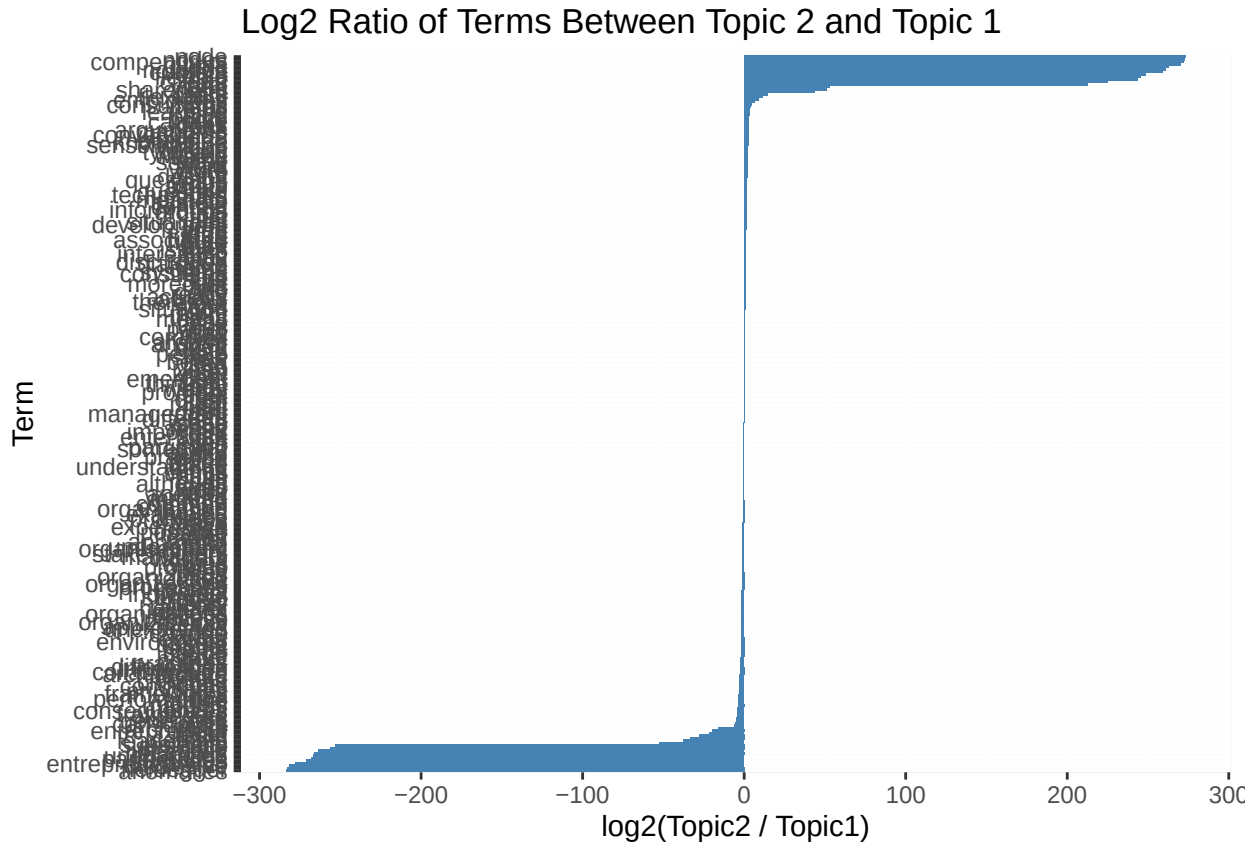


7. Compute log-ratio between topic1 and topic2 for distinctive terms

```r
beta_spread <- ap_topics %>%
  filter(topic %in% c(1, 2)) %>%
  pivot_wider(names_from = topic, values_from = beta, names_prefix = "topic") %>%
  filter(!is.na(topic1) & !is.na(topic2)) %>%
  filter(topic1 > .001 | topic2 > .001) %>%
  mutate(log_ratio = log2(topic2 / topic1),
         term = reorder(term, log_ratio))
```

8. Plot log-ratio of distinctive terms

```
ggplot(beta_spread, aes(term, log_ratio)) +
  geom_col(show.legend = FALSE, fill = "steelblue") +
  coord_flip() +
  labs(title = "Log2 Ratio of Terms Between Topic 2 and Topic 1", x = "Term", y = "log2(Topic2 / Topic1)
```



9. Extract document-topic probabilities (gamma)

```
ap_documents <- tidy(ap_lda, matrix = "gamma")
```

10. View top terms in a specific document (e.g., document 8 or 24)

```
dtm_tidy <- tidy(dtm)

dtm_tidy %>%
  filter(document == 8) %>%
  arrange(desc(count))
```

```
## # A tibble: 0 x 3
## # i 3 variables: document <chr>, term <chr>, count <dbl>
```

```r
dtm_tidy %>%
  filter(document == 24) %>%
  arrange(desc(count))
```

```
## # A tibble: 0 x 3
## # i 3 variables: document <chr>, term <chr>, count <dbl>
```