ST720 Data Science

~~Converting to and from non-tidy formats~~

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Introduction

- In text mining, we often have collections of documents, such as blog posts or news articles, that we'd like to divide into natural groups so that we can understand them separately.
- Topic modeling is a method which finds natural groups of items even when we're not sure what we're looking for.

Latent Drichilet Allocation (LDA)

- LDA is a particularly popular method for fitting a topic model.
  - treats each document as a mixture of topics,
  - each topic as a mixture of words.
- This allows documents to "overlap" each other in terms of content, rather than being separated into discrete groups.
- Learn to work with LDA objects from the `topicmodels` package, particularly tidying such models so that they can be manipulated with `ggplot2` and `dplyr`.

Latent Drichilet Allocation (LDA)

- Latent Dirichlet allocation is one of the most common algorithms for topic modeling.
- Two principles of LDA.
  - **Every document is a mixture of topics**: Each document may contain words from several topics in particular proportions. For example, in a two-topic model we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."
  - **Every topic is a mixture of words**: A two-topic model of American news, with one topic for "politics" and one for "entertainment." The most common words in the politics topic might be "President", "Congress", and "government", while the entertainment topic may be made up of words such as "movies", "television", and "actor". Importantly, words can be shared between topics; a word like "budget" might appear in both equally.

Latent Drichilet Allocation (LDA)

- Revisit `AssociatedPress` dataset in `topicmodels` package.

```
library(topicmodels)

data("AssociatedPress")
AssociatedPress
```

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

Latent Drichilet Allocation (LDA)

- We can use `LDA()` function.

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

```
## A LDA_VEM topic model with 2 topics.
```

- Here we assume the number of topics is `k = 2`.
- Fitting the model was the `easy part`: the rest of the analysis will involve exploring and interpreting the model.

Word-topic probabilities

- The tidytext package provides this method for extracting the per-topic-per-word probabilities, called β from the model. (Probability of that term being generated from that topic)

```
ap_topics <- tidy(ap_lda, matrix = "beta")
print(ap_topics, n = 5)
```

```
## # 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
## # ... with 2.094e+04 more rows
```

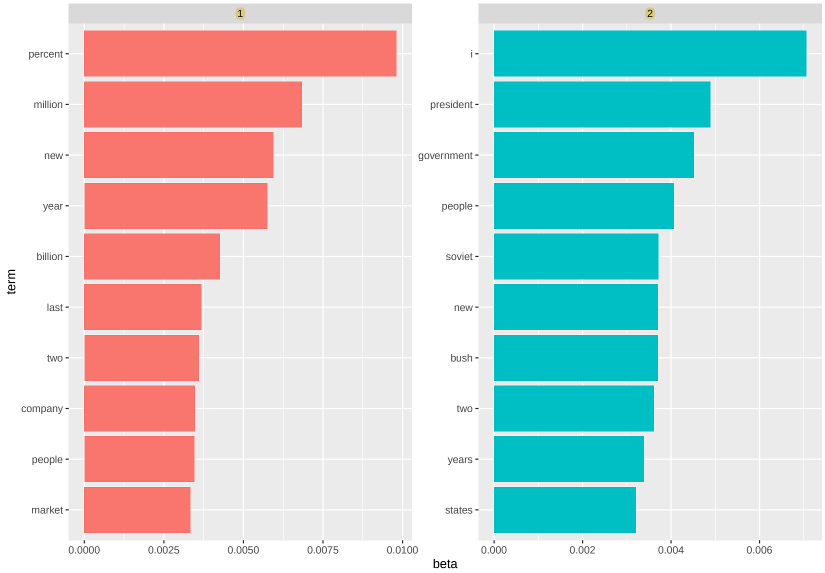
Word-topic probabilities

- Let's find the 10 terms that are most common within each topic.

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

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

Word-topic probabilities



Word-topic probabilities

- Let us understand two topics that were extracted from the articles.
  - Topic 1 includes "percent", "million", "billion", and "company", which suggests it may represent business or financial news.
  - Topic 2 include "president", "government", and "soviet", suggesting that this topic represents political news.
- Some words, such as "new" and "people", are common within both topics.
- This is an advantage of topic modeling as opposed to "hard clustering" methods: topics used in natural language could have some overlap in terms of words.

Word-topic probabilities

consider the terms that had the greatest difference in β between topic 1 and topic 2 by calculating $\log_2(\beta_2/\beta_1)$.

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

print(beta_spread, n = 5)
```

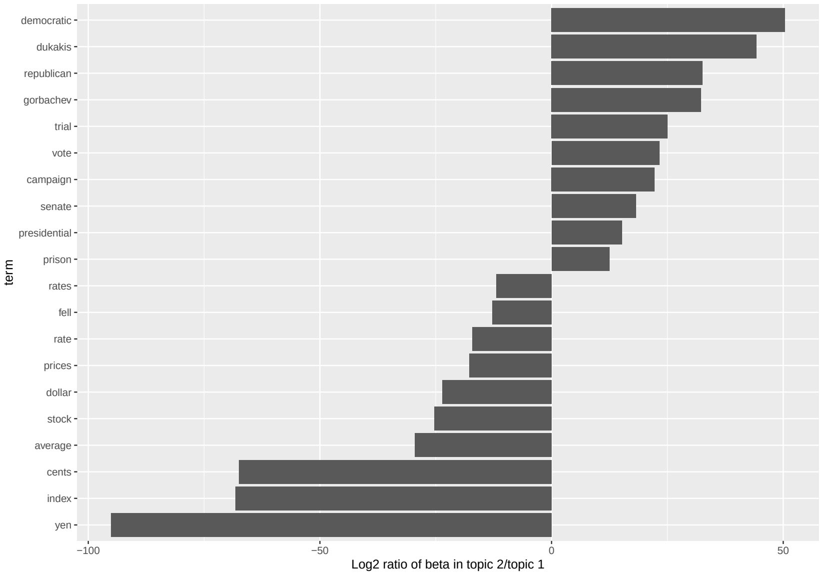
```
## # A tibble: 198 x 4
##   term              topic1   topic2 log_ratio
##   <chr>              <dbl>    <dbl>     <dbl>
## 1 administration 0.000431  0.00138      1.68
## 2 ago            0.00107   0.000842    -0.339
## 3 agreement      0.000671  0.00104      0.630
## 4 aid            0.0000476 0.00105      4.46
## 5 air            0.00214   0.000297    -2.85
## # ... with 193 more rows
```

Word-topic probabilities

- The words with the greatest differences between the two topics are visualized.

```
beta_spread %>%
    mutate(status = ifelse(log_ratio > 0, "pos", "neg")) %>%
    mutate(abs_log = abs(log_ratio)) %>%
    group_by(status) %>%
    top_n(10, abs_log) %>%
    ungroup() %>%
    mutate(term = reorder(term, log_ratio)) %>%
    ggplot(aes(term, log_ratio)) +
       geom_col(show.legend = FALSE) +
       coord_flip() +
       scale_x_reordered() +
       ylab("Log2 ratio of beta in topic 2/topic 1")
```

Word-topic probabilities



Word-topic probabilities

- Topic 2: “democratic” and “republican”, “dukakis” and “gorbachev”.
- Topic 1: “yen” and “dollar”, “index”, “prices” and “rates”.
- The two topics identified by LDA are political and financial news.

Document-topic probabilities

- LDA also models each document as a mixture of topics.
- Per-document-per-topic probabilities, γ: proportion of words from that document that are generated from that topic.
- About 24.8% of words in document 1 are from topic 1.

```r
ap_documents <- tidy(ap_lda, matrix = "gamma")
print(ap_documents, n = 5)
```

```
## # 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
## # ... with 4,487 more rows
```

Document-topic probabilities

- Many of these documents were drawn from a mix of the two topics.
- Document 6 was drawn almost entirely from topic 2, having a γ from topic 1 close to zero.

```r
tidy(AssociatedPress) %>%
  filter(document == 6) %>%
  arrange(desc(count)) %>%
  print(n = 5)
```

```
## # 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
## # ... with 282 more rows
```

- It turns out that document 6 about the relationship between the American government and Panamanian dictator Manuel Noriega.

Example: the great library heist

```
titles <- c("Twenty Thousand Leagues under the Sea", "The War of
            "Pride and Prejudice", "Great Expectations")

books <- gutenberg_works(title %in% titles) %>%
  gutenberg_download(meta_fields = "title")
```

```
## Determining mirror for Project Gutenberg from http://www.gute

## Using mirror http://aleph.gutenberg.org
```

Example: the great library heist

- divide into documents, each representing one chapter

```
by_chapter <- books %>%
  group_by(title) %>%
  mutate(chapter = cumsum(str_detect(text,
                    regex("^chapter ", ignore_case = TRUE)))) %
  ungroup() %>%
  filter(chapter > 0) %>%
  unite(document, title, chapter)
```

- Split into words

```
by_chapter_word <- by_chapter %>% unnest_tokens(word, text)
```

- find document-word counts

```
word_counts <- by_chapter_word %>%
  anti_join(stop_words) %>%
  count(document, word, sort = TRUE) %>%
  ungroup()
```

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

Example: the great library heist

```
word_counts
```

```
## # A tibble: 104,722 x 3
##    document                 word         n
##    <chr>                    <chr>    <int>
##  1 Great Expectations_57    joe         88
##  2 Great Expectations_7     joe         70
##  3 Great Expectations_17    biddy       63
##  4 Great Expectations_27    joe         58
##  5 Great Expectations_38    estella     58
##  6 Great Expectations_2     joe         56
##  7 Great Expectations_23    pocket      53
##  8 Great Expectations_15    joe         50
##  9 Great Expectations_18    joe         50
## 10 The War of the Worlds_16 brother     50
## # ... with 104,712 more rows
```

LDA on chapters

- word_counts is in a tidy form, but the topicmodels package requires a DocumentTermMatrix.

```
chapters_dtm <- word_counts %>%
  cast_dtm(document, word, n)

chapters_dtm
```

```
## <<DocumentTermMatrix (documents: 193, terms: 18215)>>
## Non-/sparse entries: 104722/3410773
## Sparsity           : 97%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

LDA on chapters

- Apply `LDA()` function to create a four-topic model.

```
chapters_lda <- LDA(chapters_dtm, k = 4, control = list(seed = 1
chapters_lda
```

```
## A LDA_VEM topic model with 4 topics.
```

LDA on chapters

- Examine per-topic-per-word probabilities: The probability of a term being generated from a topic.

```
chapter_topics <- tidy(chapters_lda, matrix = "beta")
chapter_topics
```

```
## # A tibble: 72,860 x 3
##    topic term        beta
##    <int> <chr>      <dbl>
##  1     1 joe     1.44e-17
##  2     2 joe     5.96e-61
##  3     3 joe     9.88e-25
##  4     4 joe     1.45e- 2
##  5     1 biddy   5.14e-28
##  6     2 biddy   5.02e-73
##  7     3 biddy   4.31e-48
##  8     4 biddy   4.78e- 3
##  9     1 estella 2.43e- 6
## 10     2 estella 4.32e-68
## # ... with 72,850 more rows
```

LDA on chapters

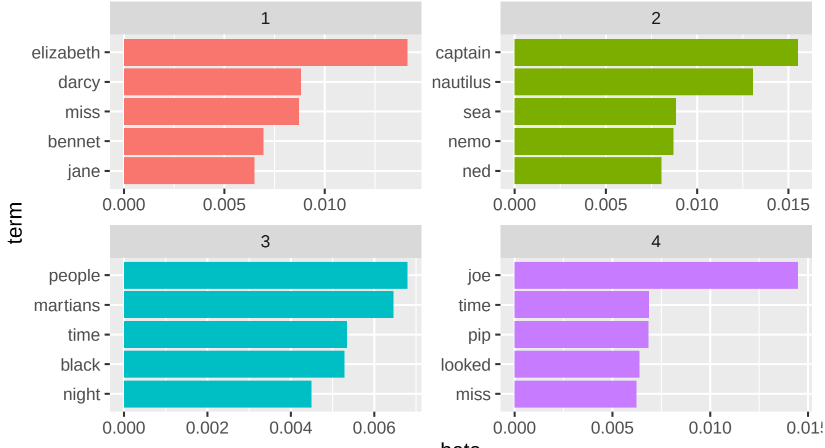
- Use dplyr top_n() to find the top 5 terms within each topic.

```
top_terms <- chapter_topics %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
print(top_terms, n = 5)
```

```
## # A tibble: 20 x 3
##   topic term          beta
##   <int> <chr>        <dbl>
## 1     1 elizabeth 0.0141
## 2     1 darcy     0.00881
## 3     1 miss      0.00871
## 4     1 bennet    0.00694
## 5     1 jane      0.00649
## # ... with 15 more rows
```

LDA on chapters

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



LDA on chapters

- Topics are pretty clearly associated with the four books.
  - Topic 1 belongs to Pride and Prejudice.
  - Topic 2 belongs to Twenty Thousand Leagues Under the Sea,
  - We see "pip" and "joe" in Topic 4 from Great Expectations and "martians", "black", and "night" in Topic 3 from The War of the Worlds.
  - There are words in common between multiple topics, such as "miss" in topics 1 and 4, and "time" in topics 3 and 4.

Per-document classification

- Each document in this analysis represented a single chapter.
- Which topics are associated with each document?
- Can we put the chapters back together in the correct books?
- Let's examining the per-document-per-topic probabilities, γ.

```r
chapters_gamma <- tidy(chapters_lda, matrix = "gamma")
print(chapters_gamma, n = 5)
```

```
## # A tibble: 772 x 3
##   document                 topic      gamma
##   <chr>                    <int>      <dbl>
## 1 Great Expectations_57        1 0.0000134
## 2 Great Expectations_7         1 0.0000146
## 3 Great Expectations_17        1 0.0000210
## 4 Great Expectations_27        1 0.0000190
## 5 Great Expectations_38        1 0.355
## # ... with 767 more rows
```

Per-document classification

- Re-separate the document name into title and chapter, after which we can visualize the per-document-per-topic probability.

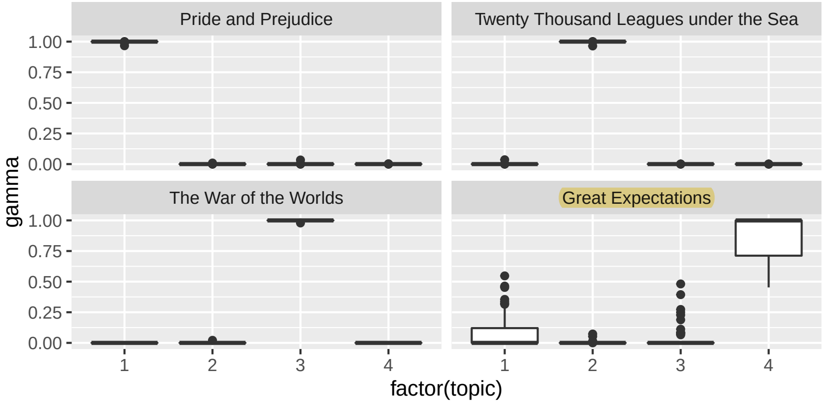
```
chapters_gamma <- chapters_gamma %>%
  separate(document, c("title", "chapter"), sep = "_", convert =

chapters_gamma
```

```
## # A tibble: 772 x 4
##    title                 chapter topic     gamma
##    <chr>                   <int> <int>     <dbl>
##  1 Great Expectations         57     1 0.0000134
##  2 Great Expectations          7     1 0.0000146
##  3 Great Expectations         17     1 0.0000210
##  4 Great Expectations         27     1 0.0000190
##  5 Great Expectations         38     1 0.355
##  6 Great Expectations          2     1 0.0000171
##  7 Great Expectations         23     1 0.547
##  8 Great Expectations         15     1 0.0124
##  9 Great Expectations         18     1 0.0000126
## 10 The War of the Worlds      16     1 0.0000107
```

Per-document classification

```
chapters_gamma %>%
  mutate(title = reorder(title, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ title)
```



Per-document classification

- All of the chapters from Pride and Prejudice, War of the Worlds, and Twenty Thousand Leagues Under the Sea were uniquely identified as a single topic each.
- Some chapters from Great Expectations (which should be topic 4) were associated with other topics.
- Are there any cases where the topic most associated with a chapter belonged to another book?

Per-document classification

```
chapter_classifications <- chapters_gamma %>%
  group_by(title, chapter) %>%
  top_n(1, gamma) %>%
  ungroup()
print(chapter_classifications, n = 8)
```

```
## # A tibble: 193 x 4
##   title                chapter topic gamma
##   <chr>                  <int> <int> <dbl>
## 1 Great Expectations        23     1 0.547
## 2 Pride and Prejudice       43     1 1.000
## 3 Pride and Prejudice       18     1 1.000
## 4 Pride and Prejudice       45     1 1.000
## 5 Pride and Prejudice       16     1 1.000
## 6 Pride and Prejudice       29     1 1.000
## 7 Pride and Prejudice       10     1 1.000
## 8 Pride and Prejudice        8     1 1.000
## # ... with 185 more rows
```

Per-document classification

- Then compare each to the "consensus" topic for each book (the most common topic among its chapters), and see which were most often misidentified.

```r
book_topics <- chapter_classifications %>%
  count(title, topic) %>%
  group_by(title) %>%
  top_n(1, n) %>%
  ungroup() %>%
  transmute(consensus = title, topic)
book_topics
```

```
## # A tibble: 4 x 2
##   consensus                                 topic
##   <chr>                                     <int>
## 1 Great Expectations                            4
## 2 Pride and Prejudice                           1
## 3 The War of the Worlds                         3
## 4 Twenty Thousand Leagues under the Sea         2
```

Per-document classification

```
chapter_classifications %>%
  inner_join(book_topics, by = "topic") %>%
  filter(title != consensus)
```

```
## # A tibble: 2 x 5
##   title                 chapter topic gamma consensus
##   <chr>                   <int> <int> <dbl> <chr>
## 1 Great Expectations         23     1 0.547 Pride and Prejudice
## 2 Great Expectations         54     3 0.481 The War of the World
```

- Only two chapters from Great Expectations were misclassified.
- LDA described one as coming from the "Pride and Prejudice" topic (topic 1) and one from The War of the Worlds (topic 3).
- That's not bad for unsupervised clustering!

By word assignments: `augment`

- LDA assigns each word in each document to a topic.
- The more words in a document are assigned to that topic, generally, the more weight (γ) will go on that document-topic classification.
- Take the original document-word pairs and find which words in each document were assigned to which topic. (`augment()`)

By word assignments: augment

```
assignments <- augment(chapters_lda, data = chapters_dtm)
assignments
```

```
## # A tibble: 104,722 x 4
##    document                 term  count .topic
##    <chr>                    <chr> <dbl>  <dbl>
##  1 Great Expectations_57 joe      88      4
##  2 Great Expectations_7  joe      70      4
##  3 Great Expectations_17 joe       5      4
##  4 Great Expectations_27 joe      58      4
##  5 Great Expectations_2  joe      56      4
##  6 Great Expectations_23 joe       1      4
##  7 Great Expectations_15 joe      50      4
##  8 Great Expectations_18 joe      50      4
##  9 Great Expectations_9  joe      44      4
## 10 Great Expectations_13 joe      40      4
## # ... with 104,712 more rows
```

- Returns a tidy data frame of book-term counts, but adds an extra column: `.topic`, with the topic each term was assigned to within each document.

By word assignments: augment

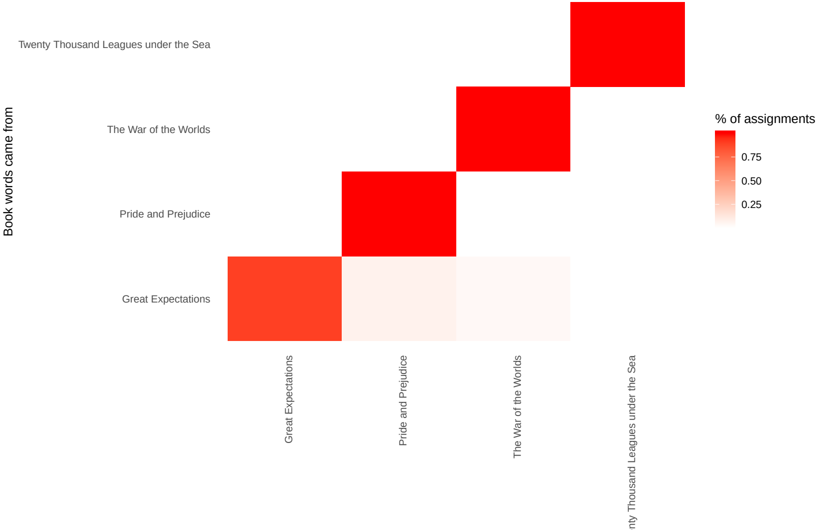
- Combine this assignments table with the consensus book titles to find which words were incorrectly classified.

```
assignments <- assignments %>%
  separate(document, c("title", "chapter"), sep = "_", convert =
  inner_join(book_topics, by = c(".topic" = "topic"))
print(assignments, n = 7)
```

```
## # A tibble: 104,722 x 6
##   title                chapter term  count .topic consensus
##   <chr>                  <int> <chr> <dbl>  <dbl> <chr>
## 1 Great Expectations        57 joe      88      4 Great Expecta
## 2 Great Expectations         7 joe      70      4 Great Expecta
## 3 Great Expectations        17 joe       5      4 Great Expecta
## 4 Great Expectations        27 joe      58      4 Great Expecta
## 5 Great Expectations         2 joe      56      4 Great Expecta
## 6 Great Expectations        23 joe       1      4 Great Expecta
## 7 Great Expectations        15 joe      50      4 Great Expecta
## # ... with 1.047e+05 more rows
```

By word assignments: `augment`

- This combination of the true book (title) and the book assigned to it (consensus) is useful for further exploration.
- Let's look at the confusion table. (visulaized by `geom_tile()`)



By word assignments: augment

- What were the most commonly mistaken words?

```
wrong_words <- assignments %>%
  filter(title != consensus)
print(wrong_words, n = 5)
```

```
## # A tibble: 4,617 x 6
##   title                               chapter term   count .topic co
##   <chr>                                 <int> <chr>  <dbl>  <dbl> <c
## 1 Great Expectations                       38 broth~     2      1 Pr
## 2 Great Expectations                       22 broth~     4      1 Pr
## 3 Great Expectations                       23 miss       2      1 Pr
## 4 Great Expectations                       22 miss      23      1 Pr
## 5 Twenty Thousand Leagues und~              8 miss       1      1 Pr
## # ... with 4,612 more rows
```

By word assignments: augment

```
wrong_words %>%
  count(title, consensus, term, wt = count) %>%
  ungroup() %>%
  arrange(desc(n))
```

```
## # A tibble: 3,551 x 4
##    title                consensus              term         n
##    <chr>                <chr>                  <chr>    <dbl>
##  1 Great Expectations Pride and Prejudice     love        44
##  2 Great Expectations Pride and Prejudice     sergeant    37
##  3 Great Expectations Pride and Prejudice     lady        32
##  4 Great Expectations Pride and Prejudice     miss        26
##  5 Great Expectations The War of the Worlds boat        25
##  6 Great Expectations The War of the Worlds tide        20
##  7 Great Expectations The War of the Worlds water       20
##  8 Great Expectations Pride and Prejudice     father      19
##  9 Great Expectations Pride and Prejudice     baby        18
## 10 Great Expectations Pride and Prejudice     flopson     18
## # ... with 3,541 more rows
```

By word assignments: `augment`

- Often assigned to the Pride and Prejudice or War of the Worlds cluster even when they appeared in Great Expectations.
- For some of these words, such as “love” and “lady”, that’s because they’re more common in Pride and Prejudice.

By word assignments: augment

- A few wrongly classified words that never appeared in the novel they were misassigned to.
- For example, "flopson" appears only in Great Expectations, even though it's assigned to the "Pride and Prejudice" cluster.

```
word_counts %>%
  filter(word == "flopson")
```

```
## # A tibble: 3 x 3
##   document                word        n
##   <chr>                   <chr>   <int>
## 1 Great Expectations_22 flopson    10
## 2 Great Expectations_23 flopson     7
## 3 Great Expectations_33 flopson     1
```

Summary

- Topic modeling finds clusters of words that characterize a set of documents.
- `tidy()` verb lets us explore and understand these models using `dplyr` and `ggplot2`, which is one of the advantages of the tidy approach to model exploration
- `LDA()` in `topicmodels` package is not the only one implementation. See `mallet` package (Mimno 2013) as well.