06 Topic Modeling

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 for unsupervised classification of such documents, similar to clustering on numeric data, which finds natural groups of items even when we're not sure what we're looking for.

In this project, I will learn to work with Latent Dirichlet allocation (LDA) objects from the topicmodels package, particularly tidying such models so that they can be manipulated with ggplot2 and dplyr. Further, I will also explore an example of clustering chapters from several books, where I can see that a topic model "learns" to tell the difference between the four books based on the text content.

06_01 Latent Dirich Allacation

LDA is a mathematical method for estimating both of these at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document. There are a number of existing implementations of this algorithm.

The AssociatedPress dataset provided by the topicmodels package, as an example of a DocumentTermMatrix. This is a collection of 2246 news articles from an American news agency, mostly published around 1988.

```
data("AssociatedPress")
AssociatedPress
```

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

This function returns an object containing the full details of the model fit, such as how words are associated with topics and how topics are associated with documents.

```
# set a seed so that the output of the model is predictable
ap_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))
ap_lda
```

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

06_01_01 Word-Topic Probabilities

The tidy() method, originally from the broom package (Robinson 2017), for tidying model objects. The tidytext package provides this method for extracting the per-topic-per-word probabilities, called "beta", from the model.

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

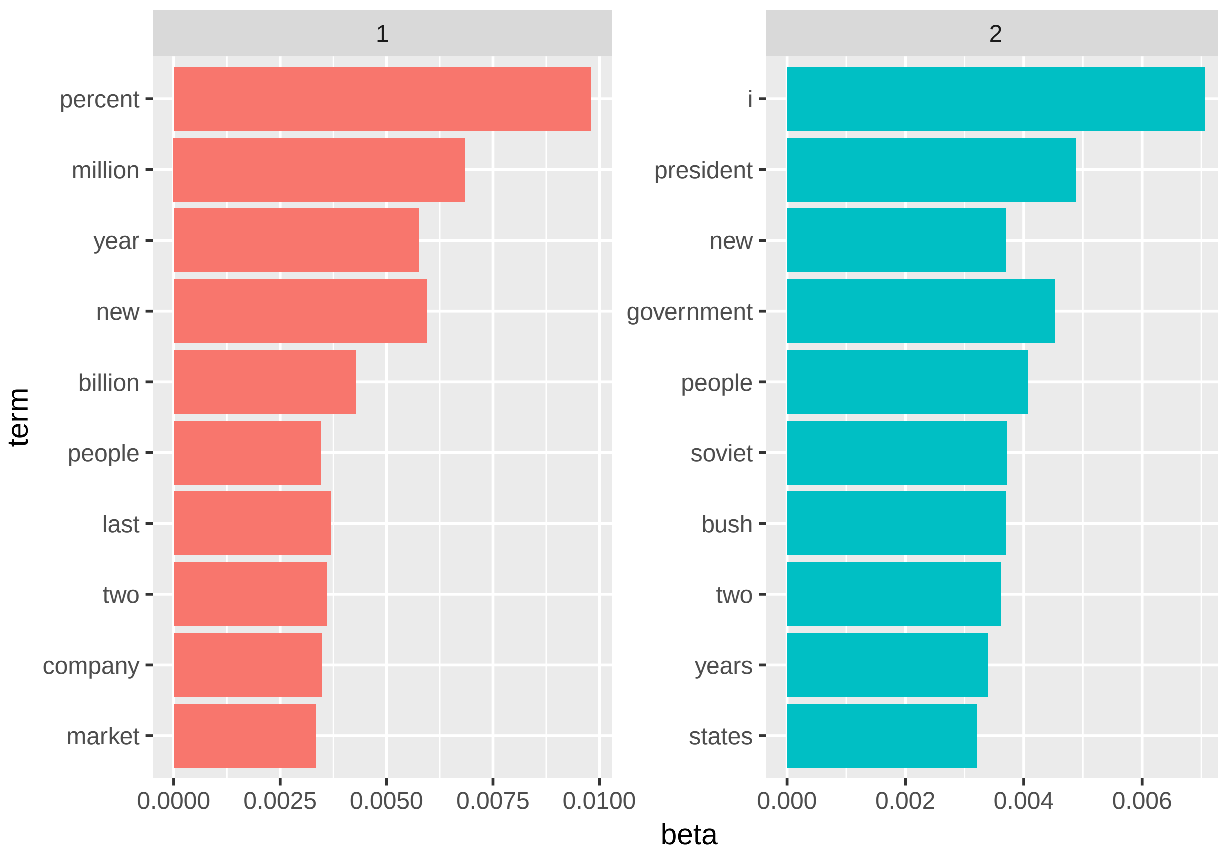
```
## # A tibble: 20,946 × 3
##    topic       term         beta
##    <int>      <chr>        <dbl>
## 1      1      aaron 1.686917e-12
## 2      2      aaron 3.895941e-05
## 3      1    abandon 2.654910e-05
## 4      2    abandon 3.990786e-05
```

```
## 5       1  abandoned 1.390663e-04
## 6       2  abandoned 5.876946e-05
## 7       1 abandoning 2.454843e-33
## 8       2 abandoning 2.337565e-05
## 9       1     abbott 2.130484e-06
## 10      2     abbott 2.968045e-05
## # ... with 20,936 more rows
```

Notice that this has turned the model into a one-topic-per-term-per-row format. For each combination, the model computes the probability of that term being generated from that topic. For example, the term "aaron" has a 1.686917×10-12 probability of being generated from topic 1, but a 3.8959408×10-5 probability of being generated from topic 2.

I want to find the 10 terms that are most common within each topic. As a tidy data frame, this lends itself well to a ggplot2 visualization.

```
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()
```



This visualization lets me understand the two topics that were extracted from the articles. The most common words in topic 1 include "percent", "million", "billion", and "company", which suggests it may represent business or financial news. Those most common in topic 2 include "president", "government", and "soviet", suggeting that this topic represents political news. One important observation about the words in each topic is that 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.

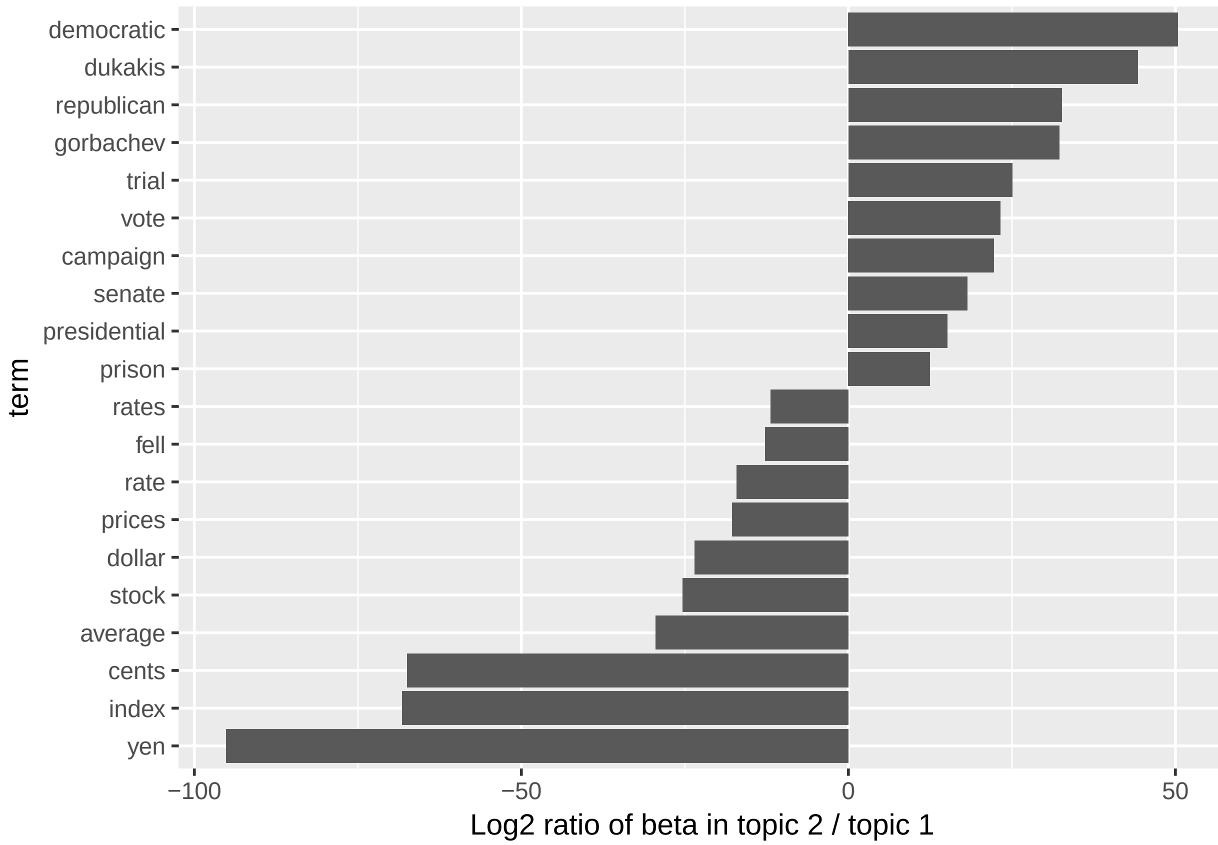
As an alternative, I could consider the terms that had the greatest difference in beta between topic 1 and topic 2.

```r
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
```

```
## # A tibble: 198 × 4
##              term       topic1       topic2   log_ratio
##             <chr>        <dbl>        <dbl>       <dbl>
## 1  administration 4.309502e-04 1.382244e-03   1.6814189
## 2             ago 1.065216e-03 8.421279e-04  -0.3390353
## 3       agreement 6.714984e-04 1.039024e-03   0.6297728
## 4             aid 4.759043e-05 1.045958e-03   4.4580091
## 5             air 2.136933e-03 2.966593e-04  -2.8486628
## 6        american 2.030497e-03 1.683884e-03  -0.2700405
## 7        analysts 1.087581e-03 5.779708e-07 -10.8778386
## 8            area 1.371397e-03 2.310280e-04  -2.5695069
## 9            army 2.622192e-04 1.048089e-03   1.9989152
## 10          asked 1.885803e-04 1.559209e-03   3.0475641
## # ... with 188 more rows
```

Next, I visualize the words with the greatest differences between the two topics.

```r
beta_spread %>%
  group_by(direction = log_ratio > 0) %>%
  top_n(10, abs(log_ratio)) %>%
  ungroup() %>%
  mutate(term = reorder(term, log_ratio)) %>%
  ggplot(aes(term, log_ratio)) +
  geom_col() +
  labs(y = "Log2 ratio of beta in topic 2 / topic 1") +
  coord_flip()
```



I can see that the words more common in topic 2 include political parties such as "democratic" and "republican", as well as politician's names such as "dukakis" and "gorbachev". Topic 1 was more characterized by currencies like "yen" and "dollar", as well as financial terms such as "index", "prices" and "rates". This helps confirm that the two topics the algorithm identified were political and financial news.

06_01_02 Document-Topic Probabilities

Besides estimating each topic as a mixture of words, LDA also models each document as a mixture of topics. I can examine the per-document-per-topic probabilities, called "gamma".

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

```
## # A tibble: 4,492 × 3
##    document topic        gamma
##       <int> <int>        <dbl>
## 1         1     1 0.2480616686
## 2         2     1 0.3615485445
## 3         3     1 0.5265844180
## 4         4     1 0.3566530023
## 5         5     1 0.1812766762
## 6         6     1 0.0005883388
## 7         7     1 0.7734215655
## 8         8     1 0.0044516994
## 9         9     1 0.9669915139
## 10       10     1 0.1468904793
## # ... with 4,482 more rows
```

Each of these values is an estimated proportion of words from that document that are generated from that topic. For example, the model estimates that only about 24.8% of the words in document 1 were generated from topic 1.

I can see that many of these documents were drawn from a mix of the two topics, but that document 6 was drawn almost entirely from topic 2, having a gamma from topic 1 close to zero. To check this answer, I could tidy() the document-term matrix and check what the most common words in that document were.

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

```
## # A tibble: 287 × 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
## # ... with 277 more rows
```

Based on the most common words, this appears to be an article about the relationship between the American government and Panamanian dictator Manuel Noriega, which means the algorithm was right to place it in topic 2 (as political/national news).

06_02 Example: The Great Library Heist

When examining a statistical method, it can be useful to try it on a very simple case where you know the "right answer". For example, we could collect a set of documents that definitely relate to four separate topics, then perform topic modeling to see whether the algorithm can correctly distinguish the four groups. This lets us double-check that the method is useful, and gain a sense of how and when it can go wrong. I will try this with some data from classic literature.

Suppose a vandal has broken into your study and torn apart four of your books:

- Great Expectations by Charles Dickens
- The War of the Worlds by H.G. Wells
- Twenty Thousand Leagues Under the Sea by Jules Verne
- Pride and Prejudice by Jane Austen

I will retrieve the text of these four books using the gutenbergr package.

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

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

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

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

```r
# 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"
```

```r
word_counts
```

```
## # A tibble: 104,721 × 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 104,711 more rows
```

06_02_01 LDA on Chapters

Right now my data frame word_counts is in a tidy form, with one-term-per-document-per-row, but the topicmodels package requires a DocumentTermMatrix. I can cast a one-token-per-row table into a DocumentTermMatrix with tidytext's cast_dtm().

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

chapters_dtm
```

```
## <<DocumentTermMatrix (documents: 193, terms: 18215)>>
## Non-/sparse entries: 104721/3410774
## Sparsity           : 97%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

I can then use the LDA() function to create a four-topic model. In this case I know I am looking for four topics because there are four books. In other problems I may need to try a few different values of k.

```
chapters_lda <- LDA(chapters_dtm, k = 4, control = list(seed = 1234))
chapters_lda
```

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

Much as I did on the Associated Press data, I can examine per-topic-per-word probabilities.

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

```
## # A tibble: 72,860 × 3
##    topic    term         beta
##    <int>   <chr>        <dbl>
## 1      1     joe 5.830326e-17
## 2      2     joe 3.194447e-57
## 3      3     joe 4.162676e-24
## 4      4     joe 1.445030e-02
## 5      1   biddy 7.846976e-27
## 6      2   biddy 4.672244e-69
## 7      3   biddy 2.259711e-46
## 8      4   biddy 4.767972e-03
## 9      1 estella 3.827272e-06
## 10     2 estella 5.316964e-65
## # ... with 72,850 more rows
```

Notice that this has turned the model into a one-topic-per-term-per-row format. For each combination, the model computes the probability of that term being generated from that topic. For example, the term "joe" has an almost zero probability of being generated from topics 1, 2, or 3, but it makes up 1.45% of topic 4.

I want to find the top 5 terms within each topic.

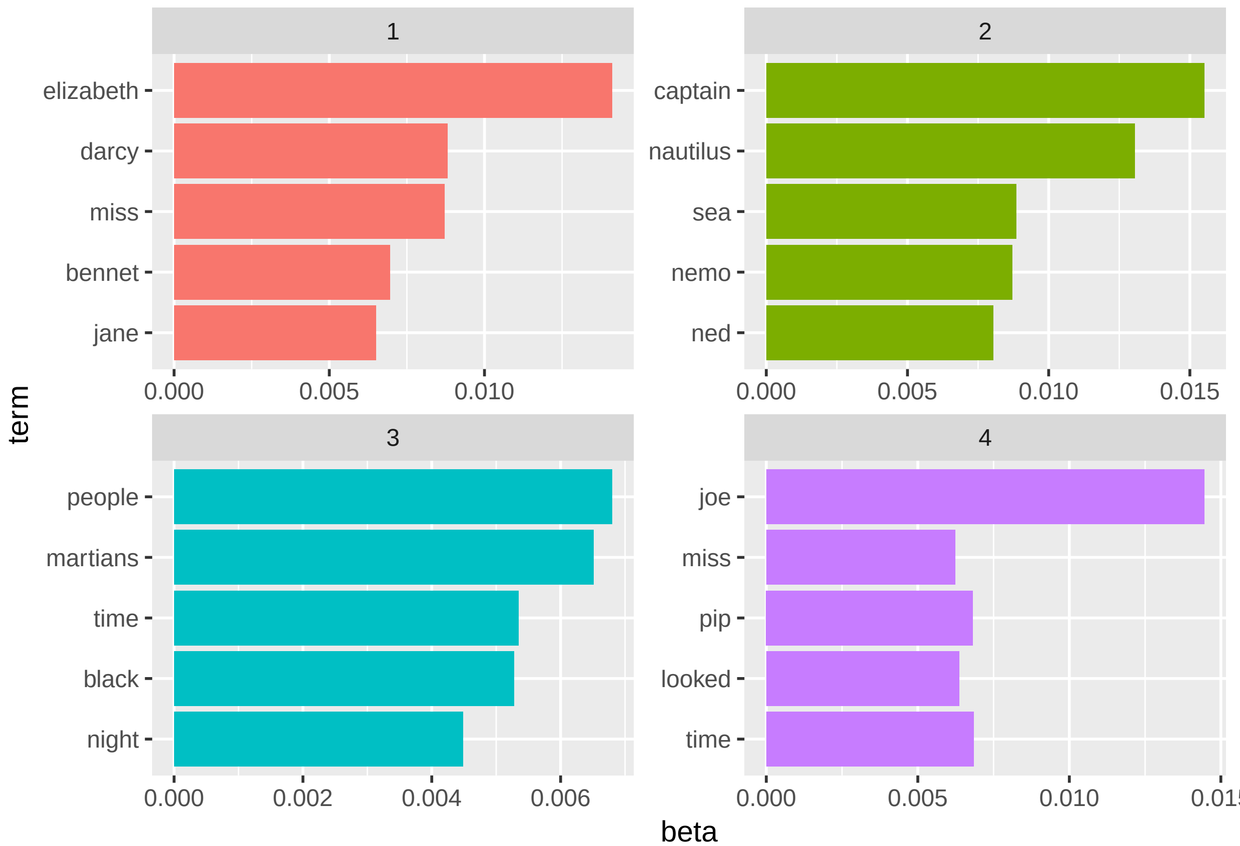
```
top_terms <- chapter_topics %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms
```

```
## # A tibble: 20 × 3
##    topic      term        beta
##    <int>     <chr>       <dbl>
## 1      1 elizabeth 0.014107538
## 2      1     darcy 0.008814258
## 3      1      miss 0.008706741
## 4      1    bennet 0.006947431
## 5      1      jane 0.006497512
## 6      2   captain 0.015507696
## 7      2  nautilus 0.013050048
## 8      2       sea 0.008850073
## 9      2      nemo 0.008708397
## 10     2       ned 0.008030799
## 11     3    people 0.006797400
## 12     3  martians 0.006512569
## 13     3      time 0.005347115
## 14     3     black 0.005278302
```

```
## 15     3     night 0.004483143
## 16     4       joe 0.014450300
## 17     4      time 0.006847574
## 18     4       pip 0.006817363
## 19     4    looked 0.006365257
## 20     4      miss 0.006228387
```

This tidy output lends itself well to a ggplot2 visualization.

```r
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()
```



These topics are pretty clearly associated with the four books! There's no question that the topic of "captain", "nautilus", "sea", and "nemo" belongs to Twenty Thousand Leagues Under the Sea, and that "jane", "darcy", and "elizabeth" belongs to Pride and Prejudice. I see "pip" and "joe" from Great Expectations and "martians", "black", and "night" from The War of the Worlds. I also notice that, in line with LDA being a "fuzzy clustering" method, there can be words in common between multiple topics, such as "miss" in topics 1 and 4, and "time" in topics 3 and 4.

06_02_02 Per-Document Classification

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

```
## # A tibble: 772 × 3
##                    document topic        gamma
##                       <chr> <int>        <dbl>
## 1     Great Expectations_57     1 1.351886e-05
## 2      Great Expectations_7     1 1.470726e-05
## 3     Great Expectations_17     1 2.117127e-05
## 4     Great Expectations_27     1 1.919746e-05
## 5     Great Expectations_38     1 3.544403e-01
## 6      Great Expectations_2     1 1.723723e-05
## 7     Great Expectations_23     1 5.507241e-01
## 8     Great Expectations_15     1 1.682503e-02
## 9     Great Expectations_18     1 1.272044e-05
## 10 The War of the Worlds_16     1 1.084337e-05
## # ... with 762 more rows
```

Each of these values is an estimated proportion of words from that document that are generated from that topic. For example, the model estimates that each word in the Great Expectations_57 document has only a 0.00135% probability of coming from topic 1 (Pride and Prejudice).

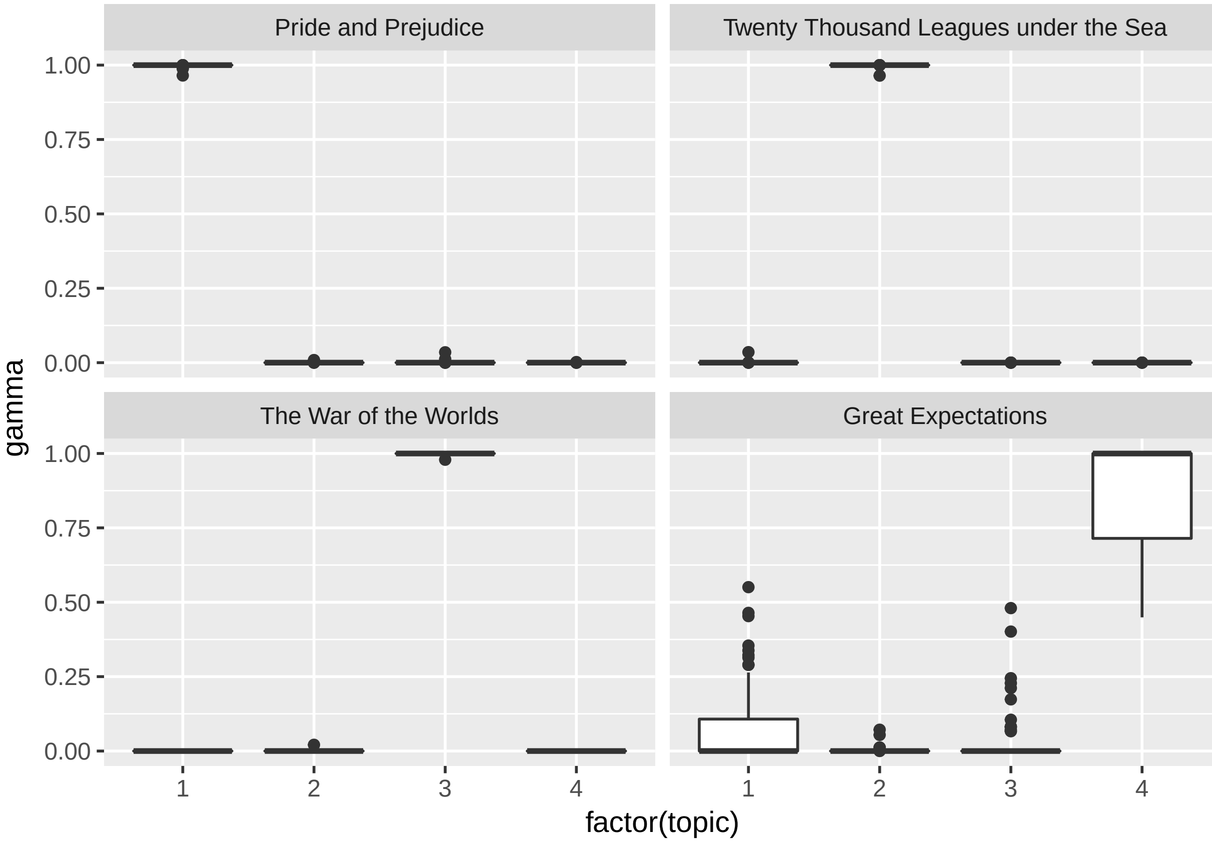
Now that I have these topic probabilities, I can see how well our unsupervised learning did at distinguishing the four books. I would expect that chapters within a book would be found to be mostly (or entirely), generated from the corresponding topic.

First I re-separate the document name into title and chapter, after which I can visualize the per-document-per-topic probability for each.

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

chapters_gamma
```

```
## # A tibble: 772 × 4
##                    title chapter topic        gamma
## *                  <chr>   <int> <int>        <dbl>
## 1     Great Expectations      57     1 1.351886e-05
## 2     Great Expectations       7     1 1.470726e-05
## 3     Great Expectations      17     1 2.117127e-05
## 4     Great Expectations      27     1 1.919746e-05
## 5     Great Expectations      38     1 3.544403e-01
## 6     Great Expectations       2     1 1.723723e-05
## 7     Great Expectations      23     1 5.507241e-01
## 8     Great Expectations      15     1 1.682503e-02
## 9     Great Expectations      18     1 1.272044e-05
## 10 The War of the Worlds      16     1 1.084337e-05
## # ... with 762 more rows
```

```
# reorder titles in order of topic 1, topic 2, etc before plotting
chapters_gamma %>%
  mutate(title = reorder(title, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ title)
```



I notice that almost 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.

It does look like some chapters from Great Expectations (which should be topic 4) were somewhat associated with other topics. Are there any cases where the topic most associated with a chapter belonged to another book? First I would find the topic that was most associated with each chapter using top_n(), which is effectively the "classification" of that chapter.

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

chapter_classifications
```

```
## # A tibble: 193 × 4
##                  title chapter topic     gamma
##                  <chr>   <int> <int>     <dbl>
## 1   Great Expectations      23     1 0.5507241
## 2  Pride and Prejudice      43     1 0.9999610
## 3  Pride and Prejudice      18     1 0.9999654
## 4  Pride and Prejudice      45     1 0.9999038
## 5  Pride and Prejudice      16     1 0.9999466
## 6  Pride and Prejudice      29     1 0.9999300
## 7  Pride and Prejudice      10     1 0.9999203
## 8  Pride and Prejudice       8     1 0.9999134
## 9  Pride and Prejudice      56     1 0.9999337
## 10 Pride and Prejudice      47     1 0.9999506
```

```
## # ... with 183 more rows
```

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

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

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

```
## # A tibble: 2 × 5
##                 title chapter topic     gamma             consensus
##                 <chr>   <int> <int>     <dbl>                 <chr>
## 1 Great Expectations      23     1 0.5507241   Pride and Prejudice
## 2 Great Expectations      54     3 0.4803234 The War of the Worlds
```

I see that only two chapters from Great Expectations were misclassified, as 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!

06_02_03 By word assignments: augment

I may want to take the original document-word pairs and find which words in each document were assigned to which topic.

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

```
## # A tibble: 104,721 × 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 104,711 more rows
```

This 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. (Extra columns added by augment always start with ., to prevent overwriting existing columns). I can combine this assignments table with the consensus book titles to find which words were incorrectly classified.

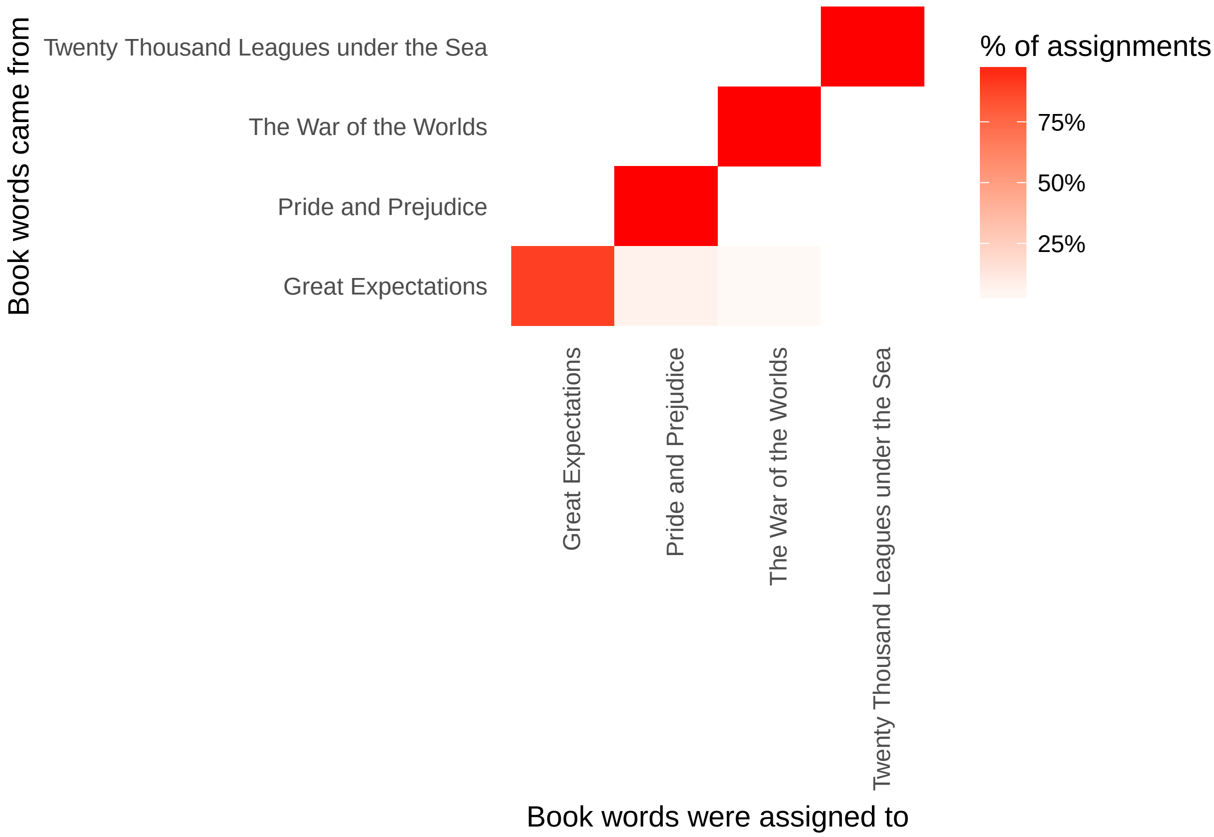
```
assignments <- assignments %>%
  separate(document, c("title", "chapter"), sep = "_", convert = TRUE) %>%
  inner_join(book_topics, by = c(".topic" = "topic"))
```

```r
assignments
```

```
## # A tibble: 104,721 × 6
##                title chapter  term count .topic          consensus
##                <chr>   <int> <chr> <dbl>  <dbl>              <chr>
## 1  Great Expectations      57   joe    88      4 Great Expectations
## 2  Great Expectations       7   joe    70      4 Great Expectations
## 3  Great Expectations      17   joe     5      4 Great Expectations
## 4  Great Expectations      27   joe    58      4 Great Expectations
## 5  Great Expectations       2   joe    56      4 Great Expectations
## 6  Great Expectations      23   joe     1      4 Great Expectations
## 7  Great Expectations      15   joe    50      4 Great Expectations
## 8  Great Expectations      18   joe    50      4 Great Expectations
## 9  Great Expectations       9   joe    44      4 Great Expectations
## 10 Great Expectations      13   joe    40      4 Great Expectations
## # ... with 104,711 more rows
```

This combination of the true book (title) and the book assigned to it (consensus) is useful for further exploration. We can, for example, visualize a confusion matrix, showing how often words from one book were assigned to another.

```r
assignments %>%
  count(title, consensus, wt = count) %>%
  group_by(title) %>%
  mutate(percent = n / sum(n)) %>%
  ggplot(aes(consensus, title, fill = percent)) +
  geom_tile() +
  scale_fill_gradient2(high = "red", label = percent_format()) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        panel.grid = element_blank()) +
  labs(x = "Book words were assigned to",
       y = "Book words came from",
       fill = "% of assignments")
```



I notice that almost all the words for Pride and Prejudice, Twenty Thousand Leagues Under the Sea, and War of the Worlds were correctly assigned, while Great Expectations had a fair number of misassigned words (which, as we saw above, led to two chapters getting misclassified).

Next, I analyze the most commonly mistaken words.

```
wrong_words <- assignments %>%
  filter(title != consensus)

wrong_words
```

```
## # A tibble: 4,535 × 6
##                                    title chapter     term count .topic
##                                    <chr>   <int>    <chr> <dbl>  <dbl>
## 1                     Great Expectations      38  brother     2      1
## 2                     Great Expectations      22  brother     4      1
## 3                     Great Expectations      23     miss     2      1
## 4                     Great Expectations      22     miss    23      1
## 5  Twenty Thousand Leagues under the Sea       8     miss     1      1
## 6                     Great Expectations      31     miss     1      1
## 7                     Great Expectations       5 sergeant    37      1
## 8                     Great Expectations      46  captain     1      2
## 9                     Great Expectations      32  captain     1      2
## 10                 The War of the Worlds      17  captain     5      2
## # ... with 4,525 more rows, and 1 more variables: consensus <chr>
```

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

```
  arrange(desc(n))
```

```
## # A tibble: 3,500 × 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   Pride and Prejudice   father    19
## 7  Great Expectations The War of the Worlds    water    19
## 8  Great Expectations   Pride and Prejudice     baby    18
## 9  Great Expectations   Pride and Prejudice  flopson    18
## 10 Great Expectations   Pride and Prejudice   family    16
## # ... with 3,490 more rows
```

I can see that a number of words were 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 (we could confirm that by examining the counts).

On the other hand, there are a few wrongly classified words that never appeared in the novel they were misassigned to. For example, I can confirm "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 × 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
```

The LDA algorithm is stochastic, and it can accidentally land on a topic that spans multiple books.

06_03 Alternative LDA Implementations

The LDA() function in the topicmodels package is only one implementation of the latent Dirichlet allocation algorithm. For example, the mallet package (Mimno 2013) implements a wrapper around the MALLET Java package for text classification tools, and the tidytext package provides tidiers for this model output as well.

The mallet package takes a somewhat different approach to the input format. For instance, it takes non-tokenized documents and performs the tokenization itself, and requires a separate file of stopwords. This means I have to collapse the text into one string for each document before performing LDA.

```
# create a vector with one string per chapter
collapsed <- by_chapter_word %>%
  anti_join(stop_words, by = "word") %>%
  mutate(word = str_replace(word, "'", "")) %>%
  group_by(document) %>%
  summarize(text = paste(word, collapse = " "))

# create an empty file of "stopwords"
file.create(empty_file <- tempfile())
```

```
## [1] TRUE
```

```
docs <- mallet.import(collapsed$document, collapsed$text, empty_file)

mallet_model <- MalletLDA(num.topics = 4)
mallet_model$loadDocuments(docs)
mallet_model$train(100)
```

Once the model is created, however, I can use the tidy() and augment() functions described in the rest of the chapter in an almost identical way. This includes extracting the probabilities of words within each topic or topics within each document.

```
# word-topic pairs
tidy(mallet_model)
```

```
## # A tibble: 71,064 × 3
##    topic    term         beta
##    <int>   <chr>        <dbl>
## 1      1 limping 2.750906e-07
## 2      2 limping 2.733162e-07
## 3      3 limping 2.574091e-07
## 4      4 limping 9.202807e-05
## 5      1  pirate 2.750906e-07
## 6      2  pirate 2.733162e-07
## 7      3  pirate 2.574091e-07
## 8      4  pirate 9.202807e-05
## 9      1  gibbet 2.750906e-07
## 10     2  gibbet 2.733162e-07
## # ... with 71,054 more rows
```

```
# document-topic pairs
tidy(mallet_model, matrix = "gamma")
```

```
## # A tibble: 772 × 3
##                  document topic      gamma
##                     <chr> <int>      <dbl>
## 1   Great Expectations_1     1 0.11269430
## 2  Great Expectations_10     1 0.09032059
## 3  Great Expectations_11     1 0.20506135
## 4  Great Expectations_12     1 0.20803698
## 5  Great Expectations_13     1 0.23848315
## 6  Great Expectations_14     1 0.19774590
## 7  Great Expectations_15     1 0.15450726
## 8  Great Expectations_16     1 0.22038917
## 9  Great Expectations_17     1 0.18792135
## 10 Great Expectations_18     1 0.13887009
## # ... with 762 more rows
```

```
# column needs to be named "term" for "augment"
term_counts <- rename(word_counts, term = word)
augment(mallet_model, term_counts)
```

```
## # A tibble: 104,721 × 4
##                    document    term     n .topic
##                       <chr>   <chr> <int>  <int>
## 1     Great Expectations_57     joe    88      4
## 2      Great Expectations_7     joe    70      4
```

```
## 3     Great Expectations_17   biddy    63      4
## 4     Great Expectations_27     joe    58      4
## 5     Great Expectations_38 estella    58      4
## 6      Great Expectations_2     joe    56      4
## 7     Great Expectations_23  pocket    53      3
## 8     Great Expectations_15     joe    50      4
## 9     Great Expectations_18     joe    50      4
## 10 The War of the Worlds_16 brother    50      3
## # ... with 104,711 more rows
```

Summary

This project introduces topic modeling for finding clusters of words that characterize a set of documents, and shows how the tidy() verb lets us explore and understand these models using dplyr and ggplot2. This is one of the advantages of the tidy approach to model exploration: the challenges of different output formats are handled by the tidying functions, and I can explore model results using a standard set of tools. In particular, I saw that topic modeling is able to separate and distinguish chapters from four separate books, and explored the limitations of the model by finding words and chapters that it assigned incorrectly.