Topic Modeling

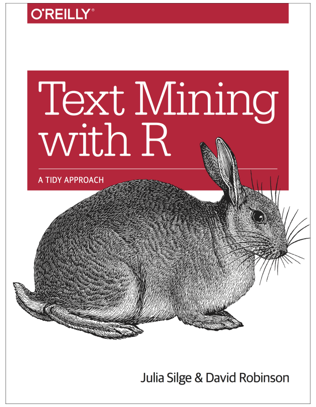
Rei Sanchez-Arias, Ph.D.

Introduction to Latent Dirichlet Allocation (LDA)

Resources and notivation

Book and packages

"Text Mining with R: A Tidy Approach" by Julia Silge and David Robinson



Examples and materials in this set of slides are adapted from this book

☑ Load the `tidyverse` and `tidytext` packages

```
library(tidyverse)
library(tidytext)
```

Often we have collections of documents (e.g. blogposts, news articles) that we would 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.

Here we work with `LDA` objects from the `topicmodels` package. Install it now from your R console: `install.packages("topicmodels")`

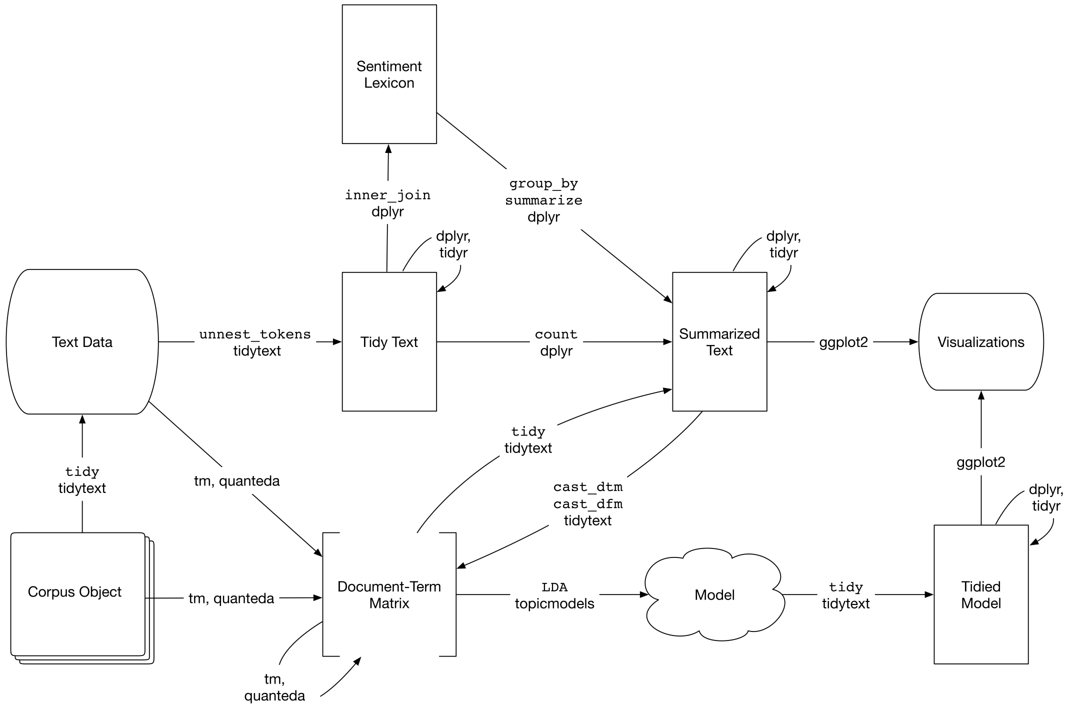
```
library(topicmodels)
```

Topic modeling with LDA

Latent Dirichlet Allocation (LDA) is a particularly popular method for fitting a topic model.

It treats each document as a mixture of topics, and 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, in a way that mirrors typical use of natural language.



https://www.tidytextmining.com/topicmodeling.html

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA)

Latent Dirichlet allocation is one of the most common algorithms for topic modeling. It is guided by two principle:

- **Every document is a mixture of topics**. We imagine that 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**. For example, we could imagine 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.

More on LDA

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. You can read the details of the algorithm in the original paper by **David Blei, Andrew Ng, and Michael Jordan** published in the Journal of Machine Learning Research.

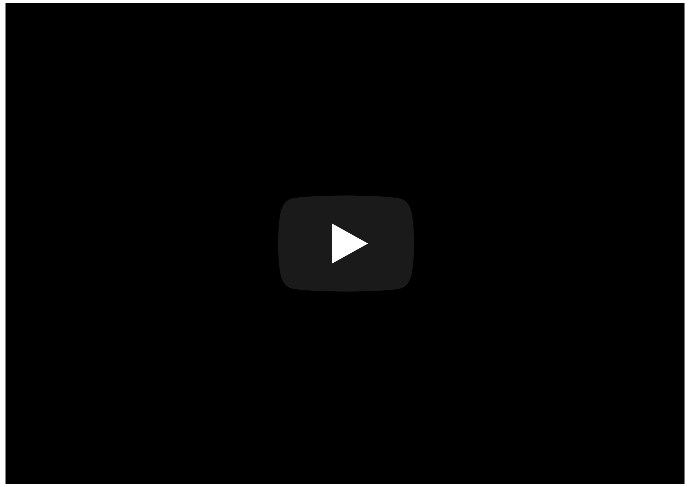
```
data("AssociatedPress")
AssociatedPress
```

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

`AssociatedPress` is an object of class `DocumentTermMatrix` provided by the package `tm`. It is a **document-term matrix** which contains the term frequency of 10473 terms in 2246 documents.

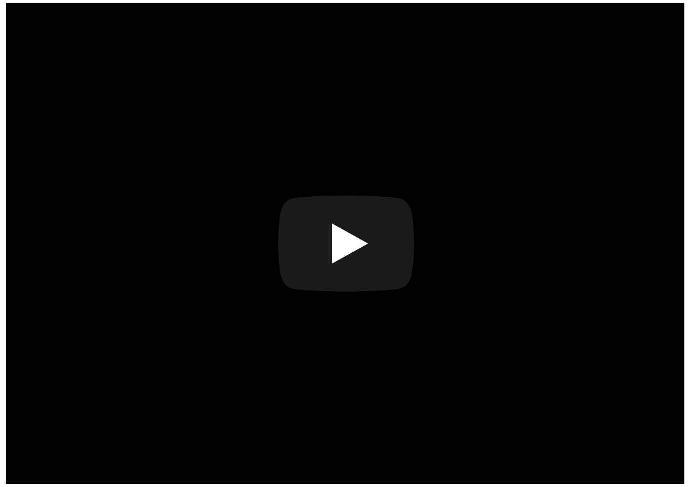
Some good videos on LDA

Prof. David Blei.
"Probabilistic Topic Models and User Behavior"



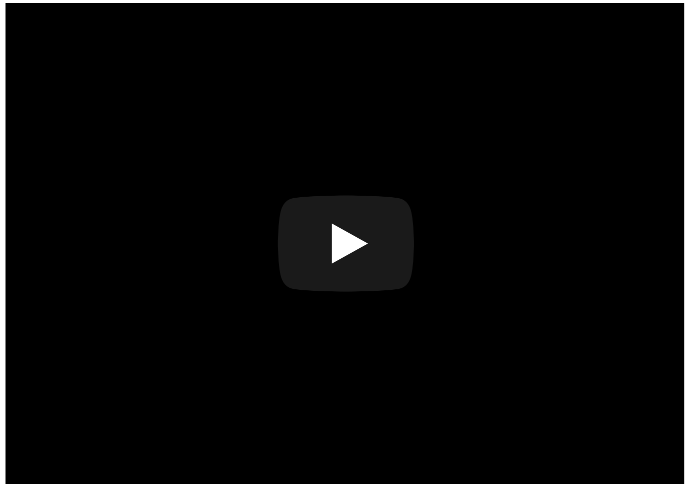
Presentation at The School of Informatics at the University of Edinburgh

Prof. Chris Bail.
"An Introduction to Topic Modeling"



Presentation part of the Summer Institute in Computational Social Science

Andrius Knispelis.
"LDA topic models: examples, intuition, and tools"



Includes discussion on LDA parameters tuning and evaluation

LDA()

We can use the `LDA()` function from the `topicmodels` package, setting `k = 2`, to create a two-topic LDA model.

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 = 12
ap_lda
```

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

Word-topic probabilities analysis

Word-topic probabilities

The `tidytext` package provides the `tidy()` method (originally from the `broom` package) for extracting the **per-topic-per-word probabilities**, called β ("beta"), from the model.

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

```r
ap_topics
```

	topic	term	beta
1	1	aaron	1.68691695574456e-12
2	2	aaron	0.0000389594077242163
3	1	abandon	0.0000265491037463324
4	2	abandon	0.0000399078632117175
5	1	abandoned	0.000139066331397393
6	2	abandoned	0.0000587694634412057
7	1	abandoning	2.45484344287377e-33

Previous 1 2 3 4 5 ... 2993

Next

One-topic-per-term-per-row

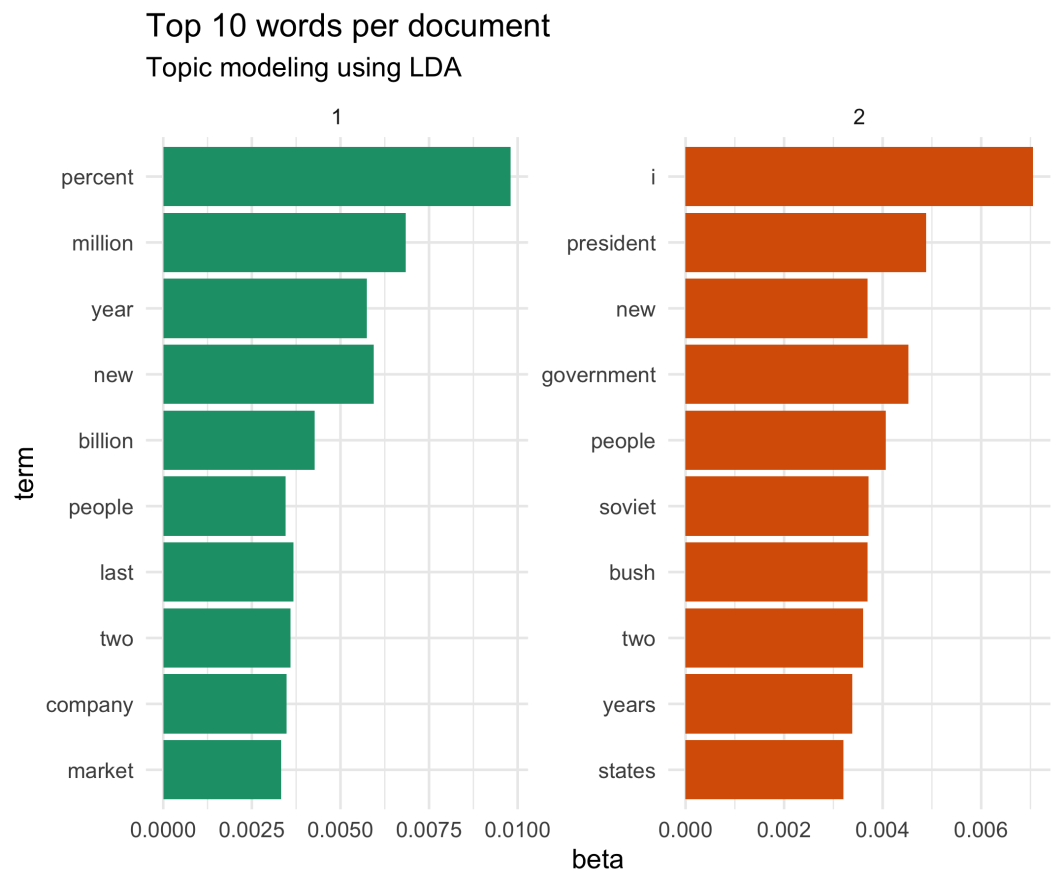
Notice that this has turned the model into a one-topic-per-term-per-row format. For each combination, the model computes the **probability of that term being generated from that topic**.

We could use `dplyr`'s `top_n()` 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 %>%
  dplyr::group_by(topic) %>%
  dplyr::top_n(10, beta) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)
```

Top 10 words per document

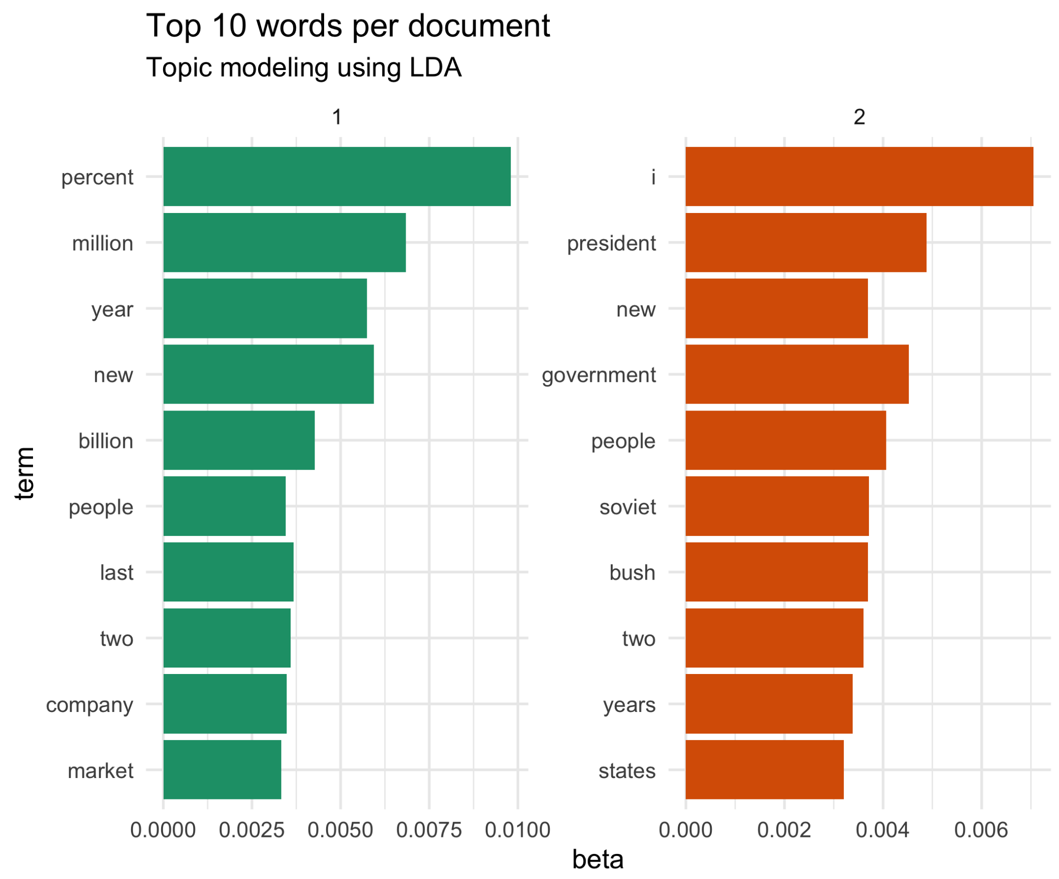
```r
ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(x = term, y = beta,
             fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  scale_fill_brewer(palette = "Dark2") +
  facet_wrap(~ topic, scales = "free") +
  labs(
    title = "Top 10 words per document",
    subtitle = "Topic modeling using LDA"
    ) +
  coord_flip() +
  theme_minimal()
```



Top words in document

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", suggesting that this topic represents **political news**.

Note: 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.



Greatest difference among topics

Alternatively, we could consider the terms that had the **greatest difference** in β between **topic 1** and **topic 2**. This can be estimated based on the **log ratio** of the two: $\log_2(\frac{\beta_2}{\beta_1})$

A log ratio is useful because it makes the difference symmetrical in the following way: β_2 being twice as large leads to a log ratio of 1, while β_1 being twice as large results in -1.

To constrain it to a set of especially relevant words, we can filter for relatively common words, such as those that have a $\beta > \frac{1}{1000}$ in at least one topic.

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

Greatest difference among topics (cont.)

	term	topic1	to
1	administration	0.000430950187459457	0.001382243627
2	ago	0.00106521639419076	0.0008421278730
3	agreement	0.000671498444274015	0.001039023824
4	aid	0.0000475904263615471	0.00104595755
5	air	0.00213693334950065	0.000296659344
6	american	0.00203049720276485	0.001683883757
7	analysts	0.00108758094151467	5.779707926219
8	area	0.0013713971871263	0.0002310279887
9	army	0.000262219225735204	0.001048088541

Previous 1 2 3 4 5 … 22

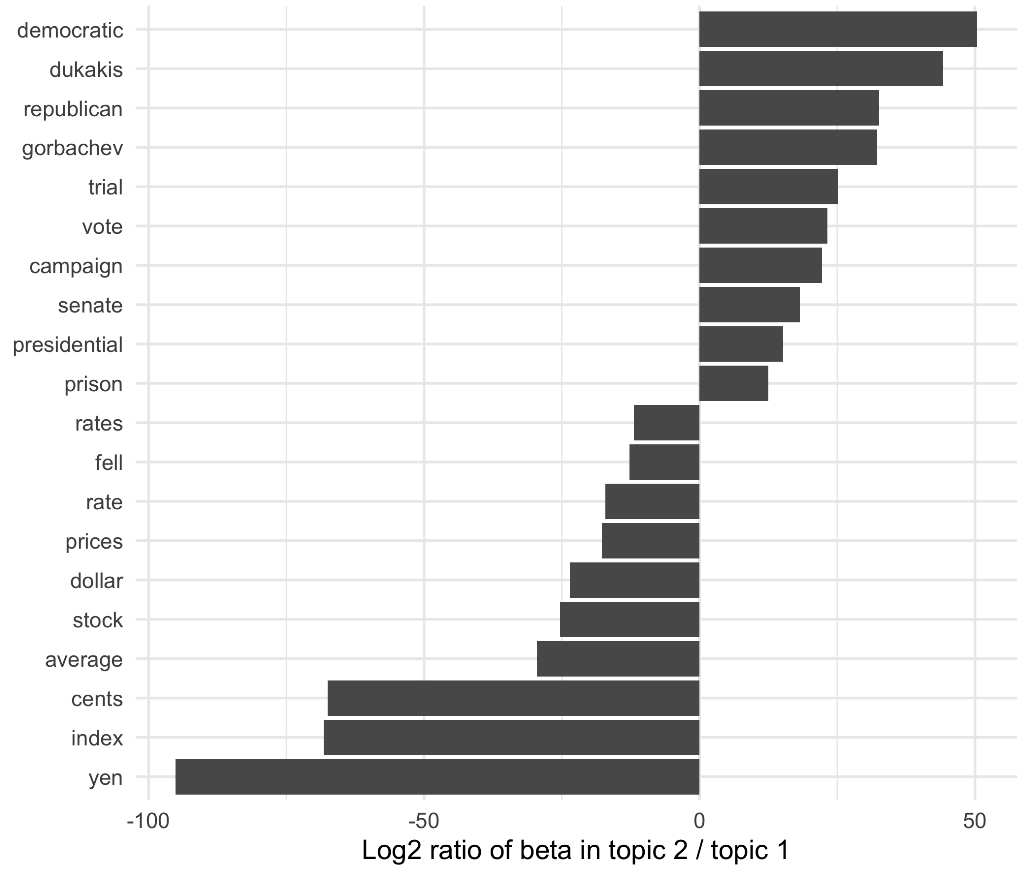
Next

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

Visualization: $\log_2(\beta_2/\beta_1)$

Notice 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”.

The two topics the algorithm identified were *political and financial news*.



Document-topic probabilities analysis

Document-topic probabilities

Besides estimating each topic as a mixture of words, LDA also models each document as a **mixture of topics**. We can examine the **per-document-per-topic probabilities**, called γ ("gamma"), with the `matrix = "gamma"` argument to `tidy()`.

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

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.

```r
ap_documents
```

	document	topic	gamma
	All	All	All
1	1	1	0.248061668636585
2	2	1	0.361548544484066
3	3	1	0.526584418041065
4	4	1	0.3566530023266
5	5	1	0.181276676169883
6	6	1	0.000588338772925416

Previous 1 2 3 4 5 … 749 Next

Per-document-per-topic probabilities

We 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 γ from topic 1 close to zero.

`tidy()` the document-term matrix and check the most common words in that document.

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

	document	term	count
1	6	noriega	16
2	6	panama	12
3	6	jackson	6
4	6	powell	6
5	6	administration	5

Previous 1 2 3 4 5 … 58

Next

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

Example: Reviews from TripAdvisor

Camp Nou TripAdvisor reviews

```r
# read data with TripAdvisor reviews
fcb <- read_csv("https://raw.githubuserconte
colnames(fcb)
```

```
## [1] "title"   "date"    "summary"
```

```r
# sample
select(fcb, summary) %>%
sample_n(size = 3)
```

```
## # A tibble: 3 x 1
##   summary
##   <chr>
## 1 I attended the match in the Camp Nou St
## 2 We had tickets for a match,  three rows
## 3 Absolutely amazing stadium. The food is
```

Perform tokenization, remove stopwords, and any other words not needed in the analysis.

```r
# Tokenization
fcb_counts <- fcb %>%
  unnest_tokens(word, summary) %>%
  anti_join(stop_words) %>%
  count(title, word, sort = TRUE) %>%
  ungroup()
# remove some words
fcb_counts <- fcb_counts %>%
  filter(!word %in% c(
    "barca", "football", "tour",
    "barcelona", "stadium",
    "camp", "nou", "fc"))
```

Document-term matrix

Right now our data frame `fcb_counts` is in a tidy form, with one-term-per-document-per-row, but the `topicmodels` package requires a `DocumentTermMatrix`. We can **cast a one-token-per-row table** into a `DocumentTermMatrix` with `tidytext`'s function `cast_dtm()`.

```
fcb_dtm <- fcb_counts %>%
             cast_dtm(title, word, n)
```

```
fcb_dtm
```

```
## <<DocumentTermMatrix (documents: 10982, terms: 12033)>>
## Non-/sparse entries: 164154/131982252
## Sparsity           : 100%
## Maximal term length: 28
## Weighting          : term frequency (tf)
```

We can now use the `LDA()` function

```
fcb_lda <- LDA(fcb_dtm, k = 4,
               control = list(seed =
fcb_lda
```

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

Per-topic-per-word probabilities

We can examine per-topic-per-word probabilities.

```
fcb_topics <- tidy(fcb_lda,
                   matrix = "beta")
```

Below we show a 100 random sample of

```
fcb_topics
```

	topic	term	beta
1	4	factory	0.0000182163760074129
2	4	hmm	0.00000650622683999297
3	4	zillion	0.00000586770170099208

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.

We could use `dplyr`'s `top_n()` to find the top 10 terms within each topic.

```
top_terms <- fcb_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
```

Top terms

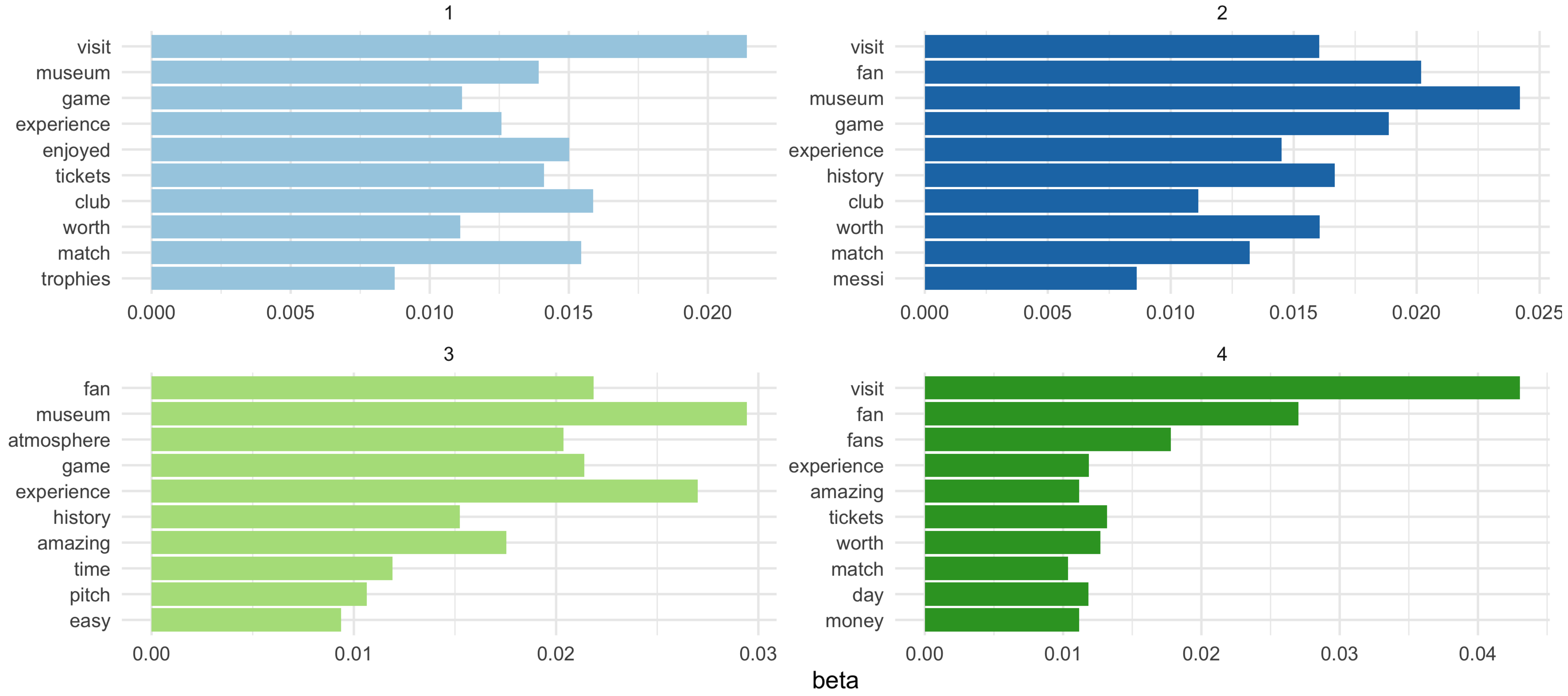
	topic	term	beta
1	1	visit	0.0214019586694283
2	1	club	0.0158698338675507
3	1	match	0.0154514110258907
4	1	enjoyed	0.0150229584942371
5	1	tickets	0.0141049224487182
6	1	museum	0.0139134666223961
7	1	experience	0.0125748814145775
8	1	game	0.0111606600239326
9	1	worth	0.0111030707935167

Previous 1 2 3 4 5 Next

```r
# visualization
top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(x = term, y = beta,
                 fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  scale_fill_brewer(palette = "Paired") +
  facet_wrap(~ topic, scales = "free") +
  labs(title = "Topic modeling in TripAdviso
        subtitle = "Latent Dirichlet Allocat
        x = "") +
  coord_flip() +
    theme_minimal()
```

Topic modeling in TripAdvisor reviews for Camp Nou stadium

Latent Dirichlet Allocation (LDA) with 4 topics


visit
museum
game
experience
enjoyed
tickets
club
worth
match
trophies
visit
fan
museum
game
experience
history
club
worth
match
messi
fan
museum
atmosphere
game
experience
history
amazing
time
pitch
easy
visit
fan
fans
experience
amazing
tickets
worth
match
day
money
beta