Topic Modeling

EDP 618 Week 12

Dr. Abhik Roy

Setting Up

1. You can retrieve the *exercising with dogs* data set and both installation and walkthrough R *scripts* by clicking on the icon below

2. Open up RStudio

3. Open up `Week 12 install.R`

4. Open `Week 12 script.R`

Take a look at the various types of files that can be imported in the tidyverse

Getting Prepped

In `Week 12 script.R`, run the following commands

1. Setting the working directory as source

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
```

2. Loading the needed packages for this walkthrough

```
library(tidyverse)
library(tidytext)
library(tm)
library(textclean)
library(topicmodels)
library(ldatuning)
library(stopwords)
library(textstem)
library(broom)
```

Alternatively if you have the **pacman** package, run `pacman::p_install("tidyverse", "tidytext", "tm", "textclean", "topicmodels", "ldatuning", "stopwords", "textstem", "broom")`

3. Bringing in the PDF data

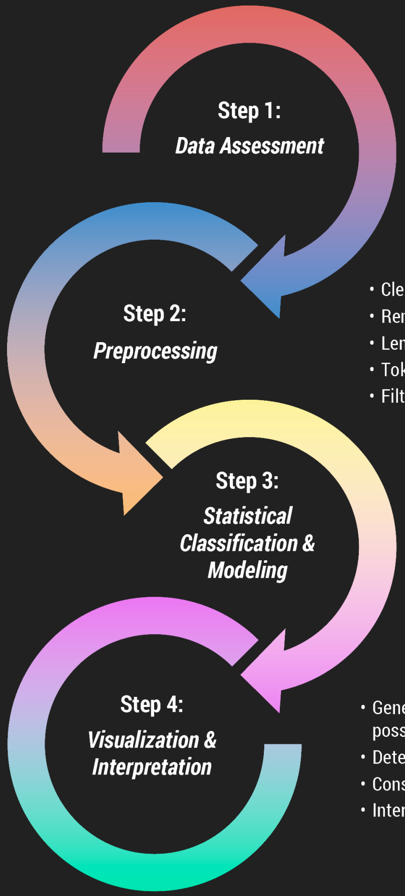
```r
exwdogs <-
  pdftools::pdf_text("exercising_with_dogs.pdf")
```

4. Retrieving stopwords

```r
data("stop_words")
```

Before We Begin

This is the process we'll cover lightly. There is a lot more going on under the hood and you may not be able to recognize all of the terms, but if you can get a basic understanding of the process, the rest can be filled in by conducting a topic model!


Step 1:
Data Assessment
Step 2:
Preprocessing
Step 3:
Statistical Classification & Modeling
Step 4:
Visualization & Interpretation

- Gather and collect text data
- Use a tidy format if possible

- Clean the data
- Remove punctuation/special characters
- Lemmatize terms
- Tokenize
- Filter stopwords

- Create a term document matrix
- Calculate a coherence score and determine an optimal number of topics
- Apply a generative (or discriminative) approach to model the text data

- Generate visuals that display terms within possible topics
- Determine topic labels
- Construct labels
- Interpret categorizations

If you didn't know, computers can't understand human languages...not directly anyway. Enter this idea below of using a medium to communicate with one (or multiple)

NLP

(Natural Language Processing)

- Provides computers with an ability to understand and process open text and spoken word
- Combines computational linguistics with statistical models

Abhik Roy

Here are a few things we won't be covering in this session so please read over the areas you lack familiarity with. Given that, it is absolutely fine if you cannot fully understand all of these ideas right now - they will hopefully become apparent as we progress

DOCUMENT

- a distinct piece of text
- could be an article, book, chapter, paragraph, sentence, etc.
- dependent on needs

Abhik Roy

CORPUS

- Latin for body
- Collection of documents

Abhik Roy

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(\underbrace{t,d}_{\text{Term frequency}}) \times \underbrace{\text{IDF}}_{\text{Inverse document frequency}}(t)$$

Number of times term t appears in a doc, d

$$\log \frac{1 + \overset{\text{\# of documents}}{n}}{1 + \underbrace{df(d,t)}_{\text{Document frequency of the term } t}} + 1$$

Chris Albon

LDA

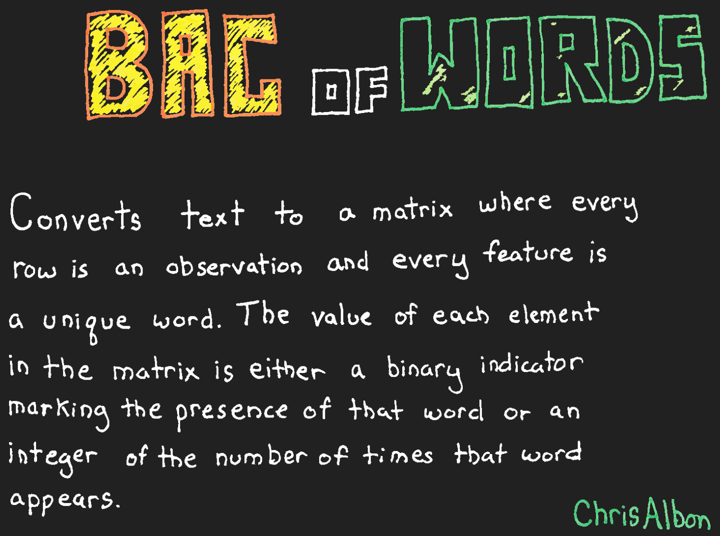
(Latent Dirichlet Allocation)

- each document is made of topics
- each topic is made of words
- discovers topics into a collection of documents
- tags each document with topics

Abhik Roy

Here are some basic terms you should try to keep while going through the walkthrough. Again it is completely fine if you do not understand what these mean in context right now!

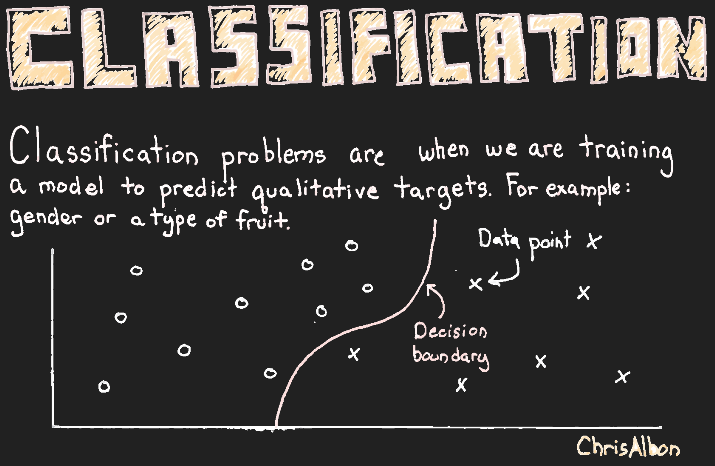
MIXING RESEARCH METHODS



BAG OF WORDS

Converts text to a matrix where every row is an observation and every feature is a unique word. The value of each element in the matrix is either a binary indicator marking the presence of that word or an integer of the number of times that word appears.

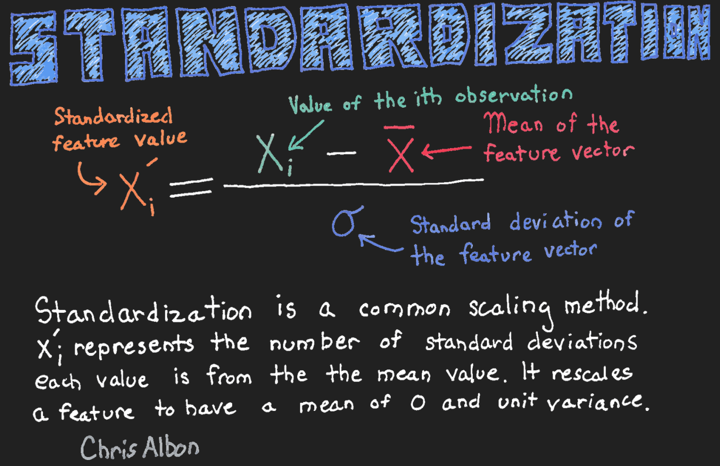
Chris Albon



CLASSIFICATION

Classification problems are when we are training a model to predict qualitative targets. For example: gender or a type of fruit.

Chris Albon



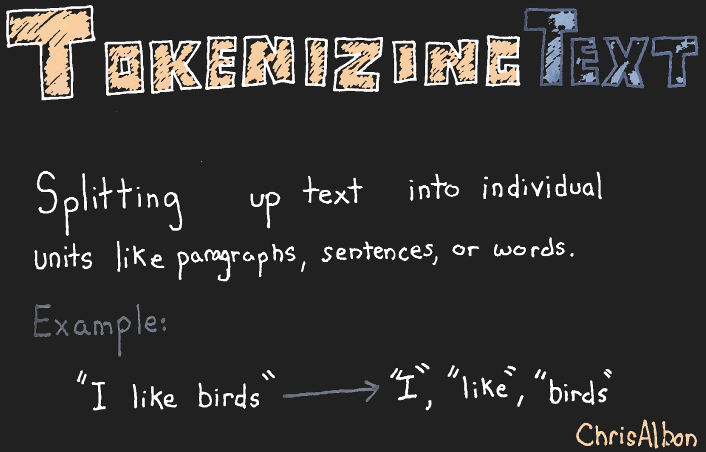
STANDARDIZATION

$$x_i' = \frac{x_i - \bar{x}}{\sigma}$$

Standardized feature value
Value of the ith observation
Mean of the feature vector
Standard deviation of the feature vector

Standardization is a common scaling method. x_i' represents the number of standard deviations each value is from the mean value. It rescales a feature to have a mean of 0 and unit variance.

Chris Albon



TOKENIZING TEXT

Splitting up text into individual units like paragraphs, sentences, or words.

Example:

"I like birds" $\longrightarrow$ "I", "like", "birds"

Chris Albon

Topic Modeling

A type of probabilistic statistical model for

(a) discovering the abstract "topics"
- or *hidden semantic structures* -
that occur in a collection of documents

(b) dimensionality reduction

The Most Annoying Thing About Data

The 80/20 Rule[1]: *Most data scientists spend only 20 percent of their time on actual data analysis and 80 percent of their time finding, cleaning, and reorganizing huge amounts of data*

[1] Loosely based on an idea called **Pareto's Principle** which states that *roughly 80% of outcomes come from 20% of causes*

Step 1: Assessing Data

1. Take a look at the data set and think about categorizing terms that may skew how terms are assessed

   the names of the dogs are not important so we could replace all of them simply with the word **dog**

   canines are prevalent in the data so we could Remo word ~~**dog**~~ altogether

2. Open up an empty text document and try going through on your own to consider terms that could be collapsed

Step 2: Preprocessing

```r
exwdogs %>%
read_lines() %>%  # Parse text into individual lines
as_tibble_col("text") %>%  # Create a single tidy column
slice(24:n()) %>%  # Remove unnecessary text
mutate(text = textclean::replace_non_ascii(text)) %>%  # Convert to a standard format
mutate(text = str_to_lower(text)) %>%  # Convert all words to lower case
mutate(text = str_remove_all(text, "[[:digit:]]")) %>%  # Remove all numbers
mutate(text = str_remove_all(text, "[[:punct:]]")) %>%  # Remove all punctuation
mutate(text = str_remove_all(text, "stage")) %>%  # Remove term
mutate(text = str_replace_all(text, "sri ranganatha temple", "temple")) %>%  # Replace term
mutate(text = str_replace_all(text, "shakti", "dog")) %>%  # Replace term
mutate(text = str_replace_all(text, "foxhounds", "dog")) %>%  # Replace term
mutate(text = str_replace_all(text, "swami", "dog")) %>%  # Replace term
mutate(text = str_replace_all(text, "max", "dog")) %>%  # Replace term
mutate(text = lemmatize_strings(text)) %>%  # Lemmatize term
mutate(text = str_remove_all(text, c("dog"))) %>%  # Remove term
mutate(text = str_remove_all(text, c("human"))) %>%  # Remove term
mutate(text = str_squish(text)) %>%  # Remove whitespace
mutate(text = na_if(text, "")) %>%  # Replace blanks with NA
drop_na()  # Drop all columns with NA
```

```
## # A tibble: 167 × 1
##    text
##    <chr>
##  1 crazy pave day temple unbearably hot join by thirsty drink my
##  2 cement mix water then come over for scratch overwhelm by how much i miss first
##  3 time i have touch another since his death strong mnemonic smell touch do bernese
##  4 mountain have double coat too feel like it she seem to recognise something be up
##  5 very much a person like cry and make her fur damp she stay with me til i stop
##  6 cry then go to entrance and lie down so hot how do she cope in here why do she
```

Assigning a Variable

Let's save the entire cleaning process

```
exwdogs_cleaned <-
  exwdogs %>%
  read_lines() %>%
  as_tibble_col("text") %>%
  slice(24:n()) %>%
  mutate(text = textclean::replace_non_ascii(text)) %>%
  mutate(text = str_to_lower(text)) %>%
  mutate(text = str_remove_all(text, "[[:digit:]]")) %>%
  mutate(text = str_remove_all(text, "[[:punct:]]")) %>%
  mutate(text = str_remove_all(text, "stage")) %>%
  mutate(text = str_replace_all(text, "sri ranganatha temple", "temple")) %>%
  mutate(text = str_replace_all(text, "shakti", "dog")) %>%
  mutate(text = str_replace_all(text, "foxhounds", "dog")) %>%
  mutate(text = str_replace_all(text, "swami", "dog")) %>%
  mutate(text = str_replace_all(text, "max", "dog")) %>%
  mutate(text = lemmatize_strings(text)) %>%
  mutate(text = str_remove_all(text, c("dog"))) %>%
  mutate(text = str_remove_all(text, c("human"))) %>%
  mutate(text = str_squish(text)) %>%
  mutate(text = na_if(text, "")) %>%
  drop_na()
```

What Just Happened?

Let's try doing something similar but with shorter and simpler text taken from the very funny skit Sharknado Pitch Meeting

```
example_text <-
  c("Excerpt from Sharknado Pitch Meeting.
     Creator: Ryan George.

     (1) It’s peer reviewed.
     (2) Multiple scientists looked over that and approved of it?
     (3) No some drunk guy on the pier checked it out. He loved it!
     (4) That is technically peer reviewed. I think we’re good.

     --The End--
    ")
```

```r
example_text %>%
  read_lines() %>%  # Parse text into individual lines
  as_tibble_col("text") %>%  # Create a single tidy column
  slice(4:n()) %>%  # Remove unnecessary text
  mutate(text = textclean::replace_non_ascii(text)) %>%  # Convert to a standard format
  mutate(text = str_to_lower(text)) %>%  # Convert all words to lower case
  mutate(text = str_remove_all(text, "[[:digit:]]")) %>%  # Remove all numbers
  mutate(text = str_remove_all(text, "[[:punct:]]")) %>%  # Remove all punctuation
  mutate(text = str_remove_all(text, "the end")) %>%  # Remove term
  mutate(text = str_replace_all(text, "multiple scientists", "scientists")) %>%  # Replace term
  mutate(text = str_replace_all(text, "its", "paper")) %>%  # Replace term
  mutate(text = str_replace_all(text, "it", "paper")) %>%  # Replace term
  mutate(text = str_replace_all(text, "that", "paper")) %>%  # Replace term
  mutate(text = lemmatize_strings(text)) %>%  # Lemmatize term
  mutate(text = str_remove_all(text, c("paper"))) %>%  # Remove term
  mutate(text = str_squish(text)) %>%  # Remove whitespace
  mutate(text = na_if(text, "")) %>%  # Replace blanks with NA
  drop_na()  # Drop all columns with NA
```

```
## # A tibble: 4 × 1
##   text
##   <chr>
## 1 peer review
## 2 scientist look over and approve of
## 3 no some drink guy on the pier check out he love
## 4 be technically peer review i think be good
```

Normalization of Remaining Wording

> is used to reduce word randomness which allows some level of standardization to help to reduce the amount of different information that a computer has to process therefore improving efficiency
>
> the overall goal is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form
>
> two popular normalization techniques are ***lemmatization*** and ***stemming***

Lemmatization vs. *Stemming*

Lemmatization

the process of reducing words to their base word

(takes more time)

Stemming

the process of reducing words to their word stem or root form by removing word endings or other affixes

(takes less time)

Example

the term *better* has the lemma *good*

Example

the term *flooding* has the stem *flood*

For a great rundown of this topic, avoid the syntax and read over Text Normalization for Natural Language Processing (NLP)

Sentence	My	friend	had	a	beautiful	singing	voice
Lemmatization	my	friend	have	a	beautiful	singing	voice
Stemming	my	friend	had	a	beauti	sing	voic

Tokenizing Handled Data

A process of distinguishing and classifying sections of a string of input characters

What you should take from this is that **unnesting** data successfully is a requirement to be able to **tokenize**. While the next set of commands should look familiar, please consider taking a bit of time to really see what occurs in each step

```r
exwdogs_cleaned %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  add_column(document = 1)
```

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

## # A tibble: 690 × 3
##    word          n document
##    <chr>     <int>    <dbl>
##  1 temple       16        1
##  2 feel         11        1
##  3 eye          10        1
##  4 care          9        1
##  5 animal        8        1
##  6 tail          8        1
##  7 sense         7        1
##  8 canine        6        1
##  9 emotional     6        1
## 10 hear          6        1
## # … with 680 more rows
```

Assign a Variable

Let's save the tokenized data frame

```
exwdogs_tokens <-
  exwdogs_cleaned %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  add_column(document = 1)
```

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

Step 3: Statistical Classification and Modeling

```r
exwdogs_tokens %>%
  cast_dtm(document, word, n)
```

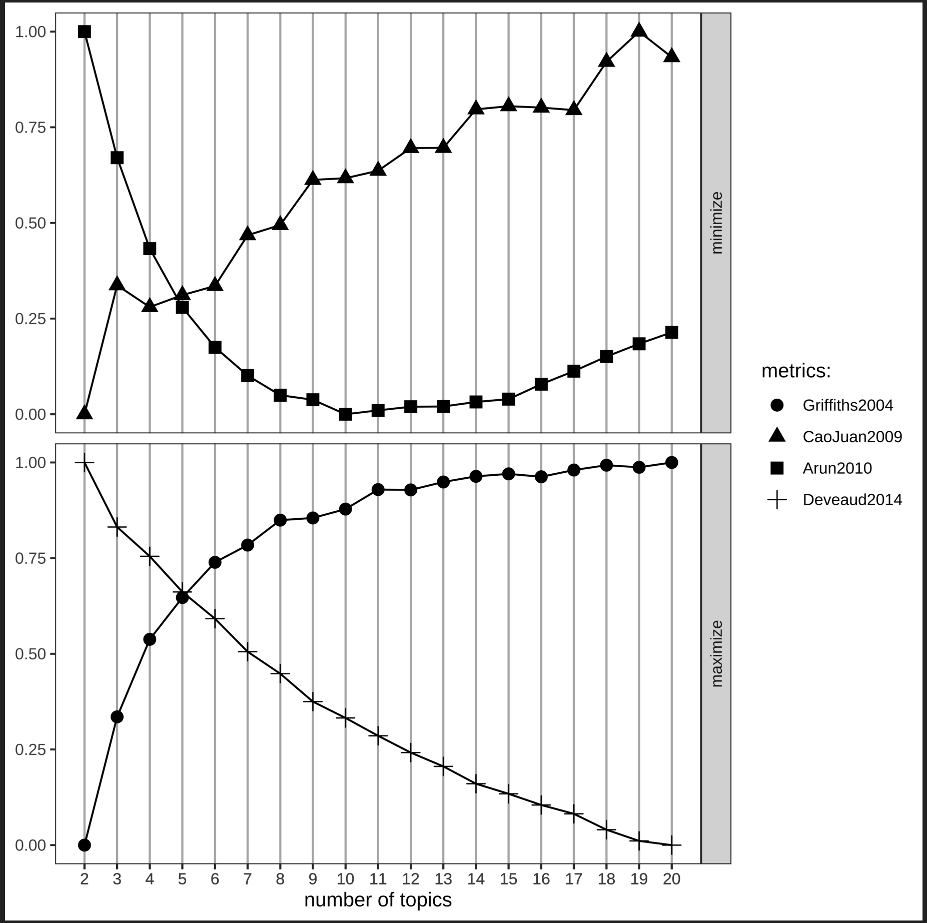
```
## <<DocumentTermMatrix (documents: 1, terms:
## Non-/sparse entries: 690/0
## Sparsity           : 0%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

Assigning a Variable

```
exwdogs_dtm <-
  exwdogs_tokens %>%
  cast_dtm(document, word, n)
```

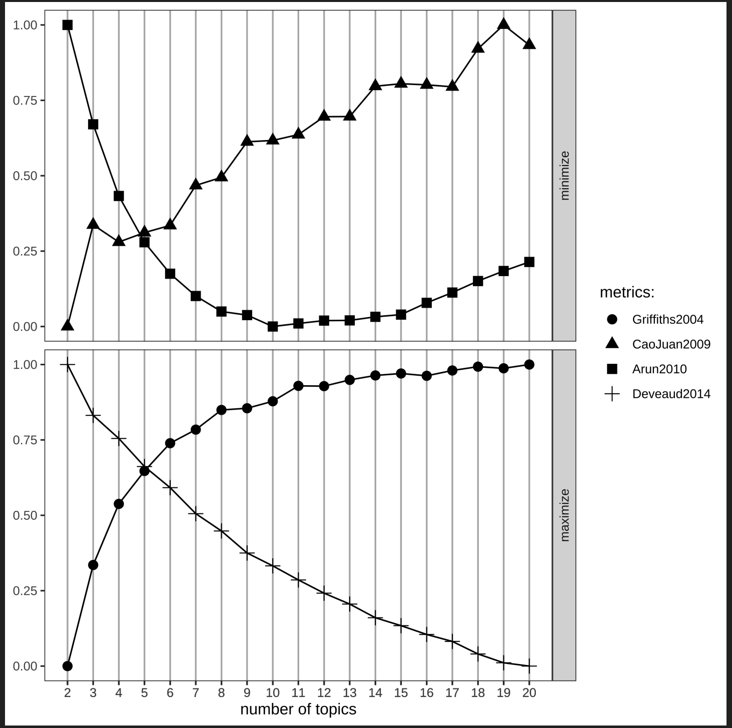
```r
FindTopicsNumber(
  exwdogs_dtm,
  topics = seq(from = 2, to = 20, by = 1),
  metrics = c("Griffiths2004",
              "CaoJuan2009",
              "Arun2010",
              "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2L,
  verbose = TRUE
  ) %>%
  FindTopicsNumber_plot()
```

```
## fit models... done.
## calculate metrics:
##   Griffiths2004... done.
##   CaoJuan2009... done.
##   Arun2010... done.
##   Deveaud2014... done.
```



Assigning a Variable

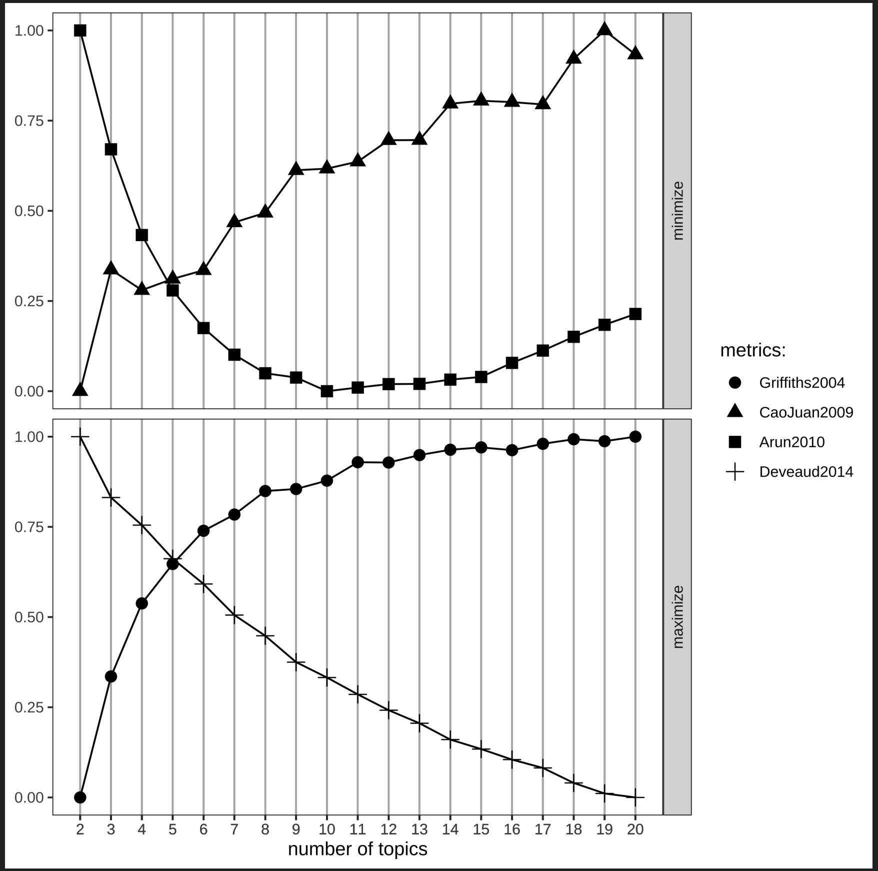
```
exwdogs_topic_est <-
  FindTopicsNumber(
  exwdogs_dtm,
  topics = seq(from = 2, to = 20, by = 1), # amend these
  metrics = c("Griffiths2004",
              "CaoJuan2009",
              "Arun2010",
              "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2L,
  verbose = TRUE
  ) %>%
  FindTopicsNumber_plot()
```



The estimate for a total number of topics can be a lowest single value or range of values. We do this by observing where the metric curves tend to plateau and get as close to each other as possible along the horizontal axis. This is known as a limit

From the plot, the metrics symbolized by △ and + are already diverging from each other. While they may head back towards the horizontal axis in the future, the metrics symbolized by □ and ○ look to be the closest between 11 and 15

So let's start by modeling 11 topics!



```r
LDA(exwdogs_dtm,
    k = 11,   # Number of topics
    control = list(seed = 1234)) %>%
  tidy(matrix = "beta")
```

```
## # A tibble: 7,590 × 3
##    topic term      beta
##    <int> <chr>    <dbl>
##  1     1 temple 0.0165
##  2     2 temple 0.0221
##  3     3 temple 0.0191
##  4     4 temple 0.00465
##  5     5 temple 0.0204
##  6     6 temple 0.0173
##  7     7 temple 0.00208
##  8     8 temple 0.0247
##  9     9 temple 0.0198
## 10    10 temple 0.00473
## # … with 7,580 more rows
```

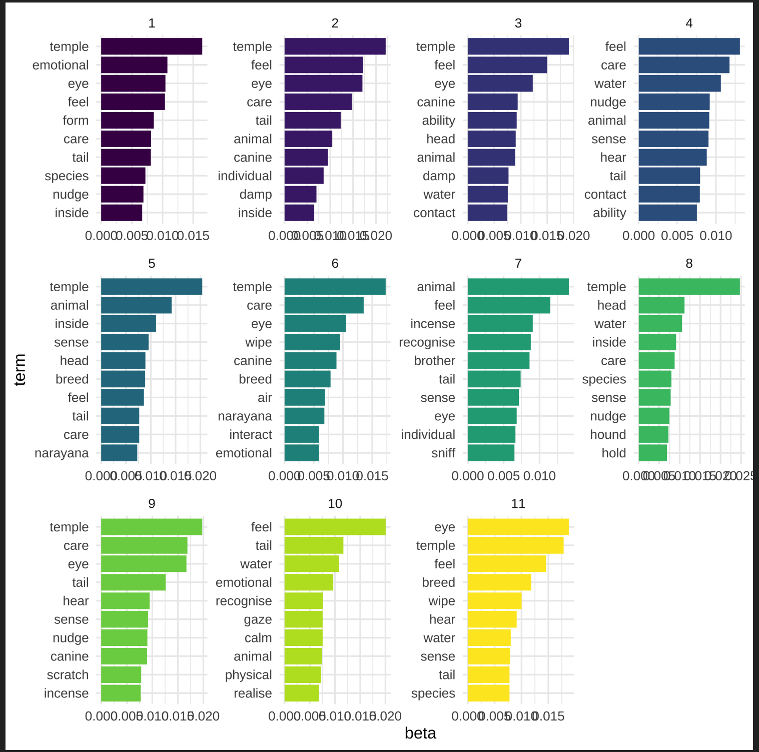
Assigning a Variable

Let's save the topics list

```
exwdogs_topics <-
  LDA(exwdogs_dtm,
      k = 11, # Amend this to test a certain number of topics
      control = list(seed = 1234)) %>%
  tidy(matrix = "beta")
```

Step 4: Visualization and Interpretation

```r
exwdogs_topics %>%
group_by(topic) %>%
slice_max(beta, n = 10) %>%
ungroup() %>%
arrange(topic, -beta) %>%
mutate(term = reorder_within(term, beta, topic)) %>%
ggplot(aes(beta, term, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
scale_fill_viridis_d() +
facet_wrap(~ topic, scales = "free") +
scale_y_reordered() +
theme_minimal()
```

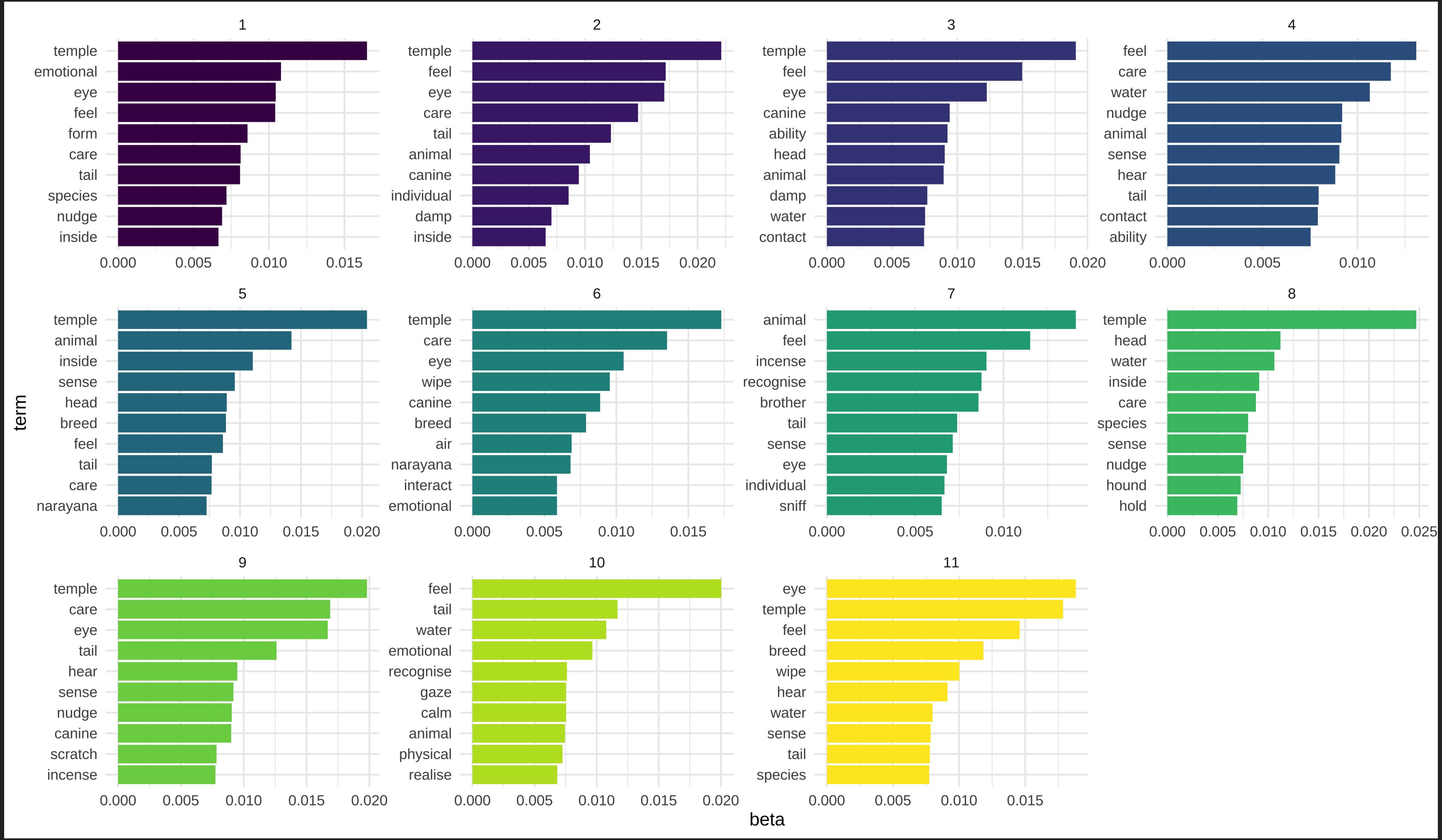


MIXING RESEARCH METHODS

Assigning a Variable

Let's save the plot

```
exwdogs_top_terms <-
  exwdogs_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  scale_fill_viridis_d() +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered() +
  theme_minimal()
```



You can save high (or really any) resolution visuals easily using `ggsave`

What Just Happened?

> LDA is a form of (unsupervised) learning that views documents as bags-of-words (BoW) where order does not matter. Not having to track the placement of every term saves a lot of time and computational energy
>
> LDA works by first making a key assumption: the way a document was generated was by picking a set of topics and then for each topic picking a set of words

Steps to Finding Topics

In a nutshell for each document m

1. Assume there are k topics across all of the documents
2. Create a distribution α where the k topics are symmetric or asymmetrically spread across each document m by assigning each word a topic
3. For each word w in every document m, assume its topic is is associated incorrectly but every other word is assigned the correct topic
4. Probabilistically assign word w a topic based on two things:
    - what topics are in document m
    - Create a distribution β to assess how many times word w has been assigned a particular topic across all of the documents
5. Repeat this process a number of times for each document until saturation

Interpret

> Much like you would assess a factor or component, the topics are unlabeled and it is up to you to figure out what they could mean. Not every topic may be directly applicable, but should still be interpreted and reported. Discarding topics means that you are removing potentially relevant information

Here is a brief assessment of some possible topics that are represented in the topic model with reference to *dogs*

Topic	Label
1	assessing humans' emotional states
2	adressing the needs of crying people
3	eye contact between humans and dogs
4	ability to care for humans
5	aptitude to sense others emotional states without being physcially present
6	physically removing emotional artifacts
7	ability to read the world through smell
8	skill in refocusing attention on them
9	need for physical touch
10	focusing attention via the use of staring
11	senses and responses affect on emotional states

Your assessment would likely differ to varying degrees and that is the point - in that qualitative concepts such as triangulation and saturation still play a large and impactful role in the interpretation phase. Note with a much larger text data set, this task could be significantly easier

That's It!

Any questions?