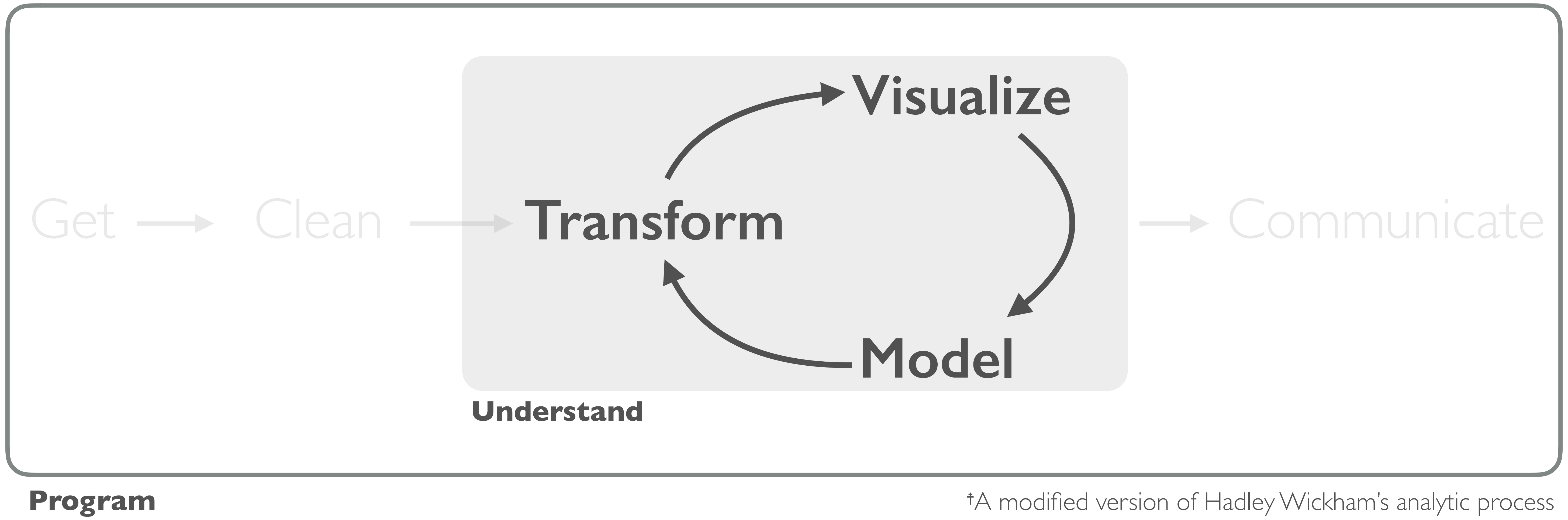
ALTERNATIVE VIEWS OF ASSOCIATION & STRUCTURE



†A modified version of Hadley Wickham's analytic process

"Grasping the structure of a subject is understanding it in a way that permits many other things to be related to it meaningfully. To learn structure in short, is to learn how things are related."

– Jerome Bruner

MANY ALTERNATIVES VIEWS TO ASSESS ASSOCIATION & STRUCTURE

- Term frequency - document frequency (tf-idf)
- Word networks
- Cluster analysis
- Topic modeling
- and more!

Unsupervised modeling approaches

PREREQUISITES

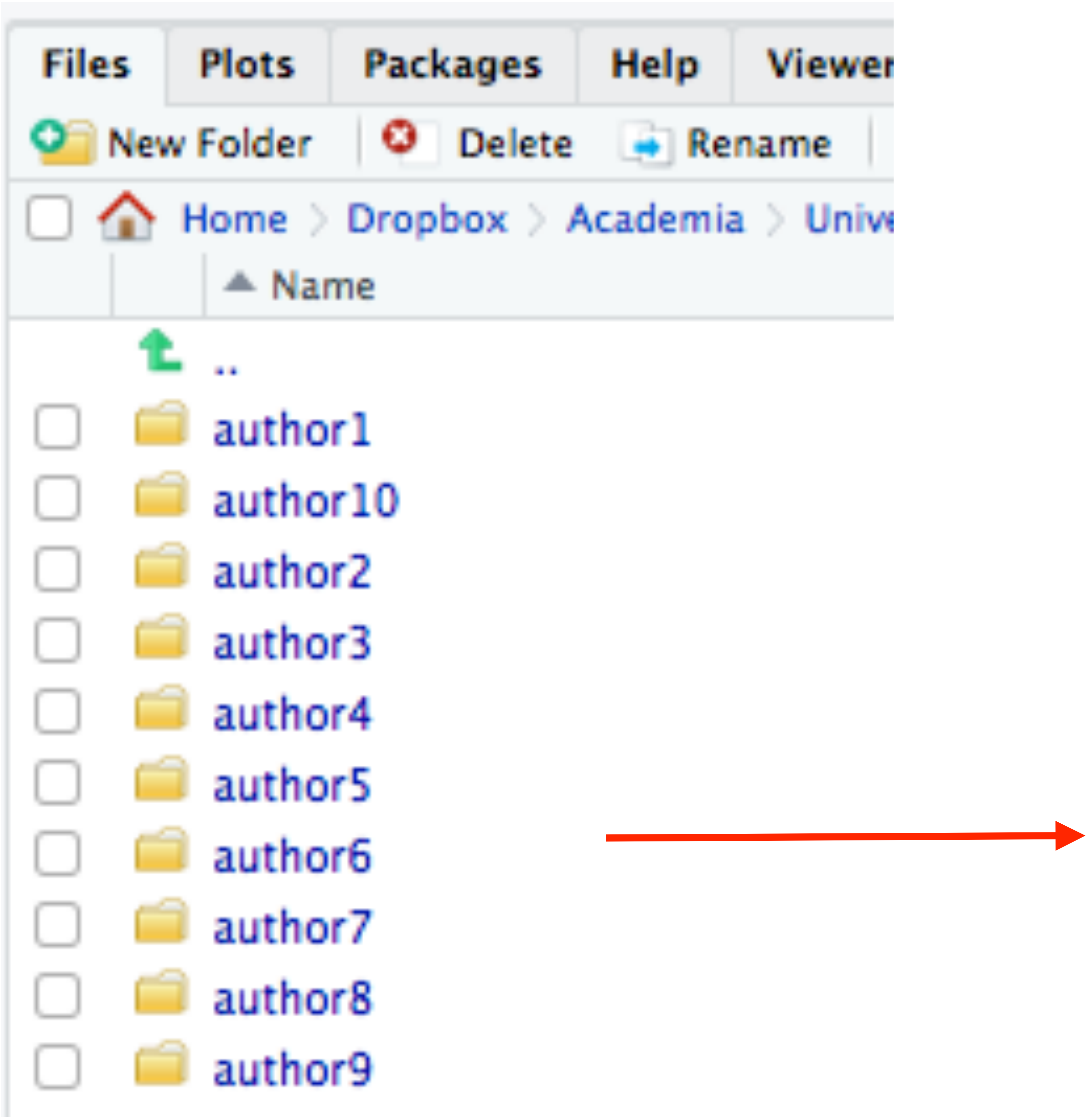
PACKAGE PREREQUISITE

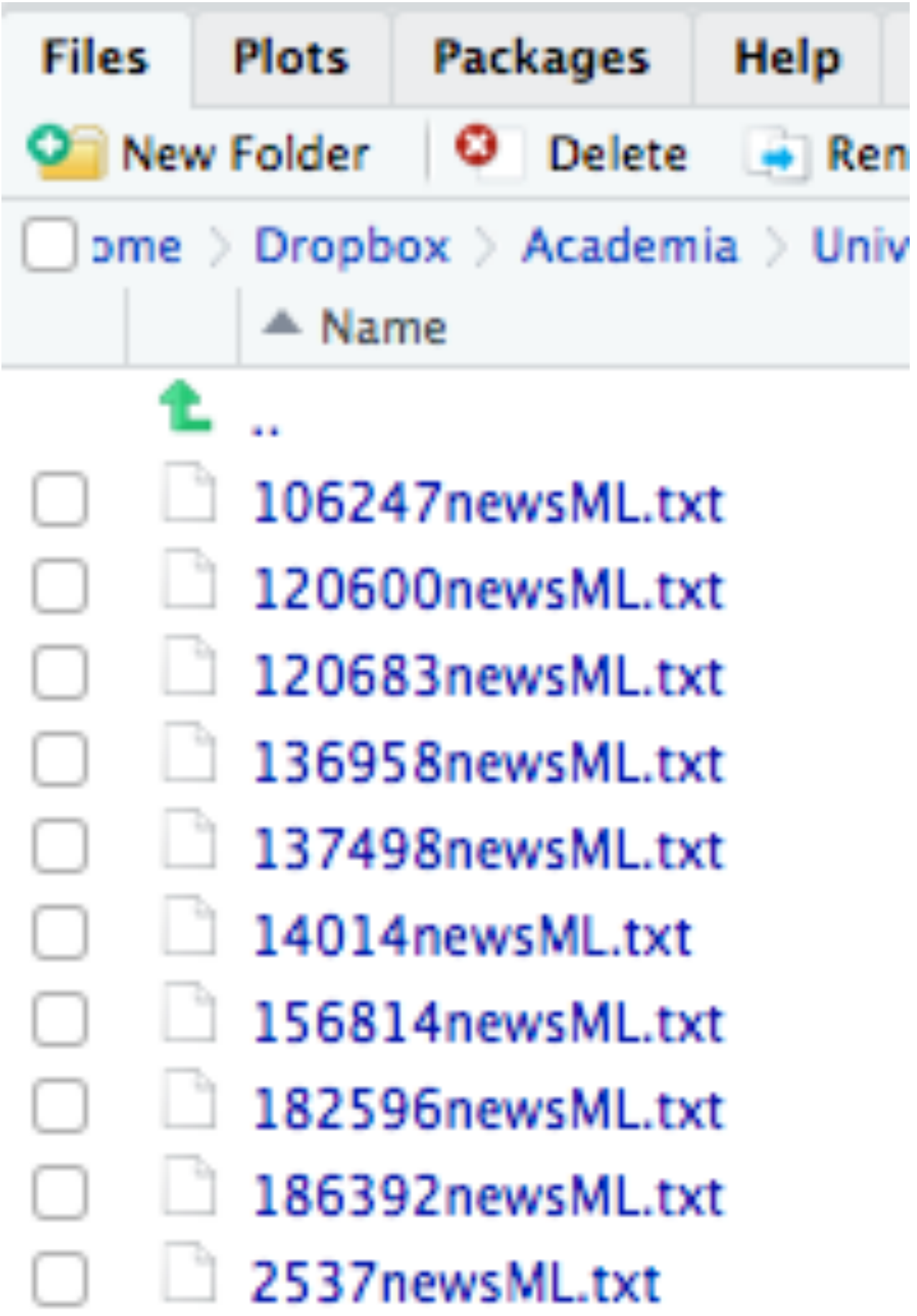
```
library(tidyverse)       # data wrangling & plotting
library(tidytext)        # efficient text manipulation
library(harrypotter)     # text for demonstration
```

DATA PREREQUISITE

```
# example data (harry potter)
ps_df <- tibble(
  chapter = seq_along(philosophers_stone),
  text    = philosophers_stone
)

# Airbnb data
url1 <- "https://raw.githubusercontent.com/kwartler/text_mining/master/bos_airbnb_1k.csv"
reviews <- read_csv(url1)

# Resume data
url2 <- “https://raw.githubusercontent.com/kwartler/text_mining/master/1yr_plus_final4.csv"
reviews <- read_csv(url2)
```

DATA PREREQUISITE





tf-idf

Finding what's unique to a particular document

TERM VS. DOCUMENT FREQUENCY

- So far we have focused on identifying the frequency of individual terms within a document along with the sentiments that these words provide.
- It is also important to understand the frequency of words *within* a document relative to all documents.
- A popular approach used by many search engine/queries is:

$$tf\text{-}idf(t, d, D) = \underbrace{tf(t, d)}_{\text{frequency of term } (t) \text{ in document } (d)} \times \underbrace{idf(t, D)}_{\text{inverse document frequency of term } (t)\text{: log(total documents / document where term } t \text{ appears)}}$$

TERM VS. DOCUMENT FREQUENCY

single customer review

corpus of 100 reviews

Word	tf	idf	tf-dif
Retailer	20	log(100/90) = 0.105	20 × 0.105 = 2.107
Ignored	3	log(100/5) = 2.996	3 × 2.996 = 8.987

COMPUTING TF-IDF

```
# compute the tf-idf for chapter in Philosopher's Stone
ps_df %>%
  unnest_tokens(word, text) %>%
  count(chapter, word) %>%
  bind_tf_idf(term_col = word, document_col = chapter, n_col = n) %>%
  arrange(desc(tf_idf))
# A tibble: 20,504 x 6
   chapter       word     n          tf       idf      tf_idf
     <int>      <chr> <int>       <dbl>     <dbl>       <dbl>
 1       1    dursley    45 0.009736045 1.7346011 0.016888154
 2      11      flint    14 0.004194128 2.8332133 0.011882860
 3       3     vernon    51 0.013226141 0.8873032 0.011735597
 4      15      ronan    20 0.003918495 2.8332133 0.011101933
 5      14    norbert    20 0.005762028 1.7346011 0.009994820
 6       3      uncle    54 0.014004149 0.6359888 0.008906482
 7       2      piers    13 0.003761574 2.1400662 0.008050017
 8       2     dudley    42 0.012152778 0.6359888 0.007729030
 9       5 ollivander    18 0.002721911 2.8332133 0.007711756
10       1        cat    20 0.004327131 1.7346011 0.007505846
# ... with 20,494 more rows
```

`bind_tf_idf` computes:

- tf
- idf
- tf-idf

We don't even need to remove stop words!

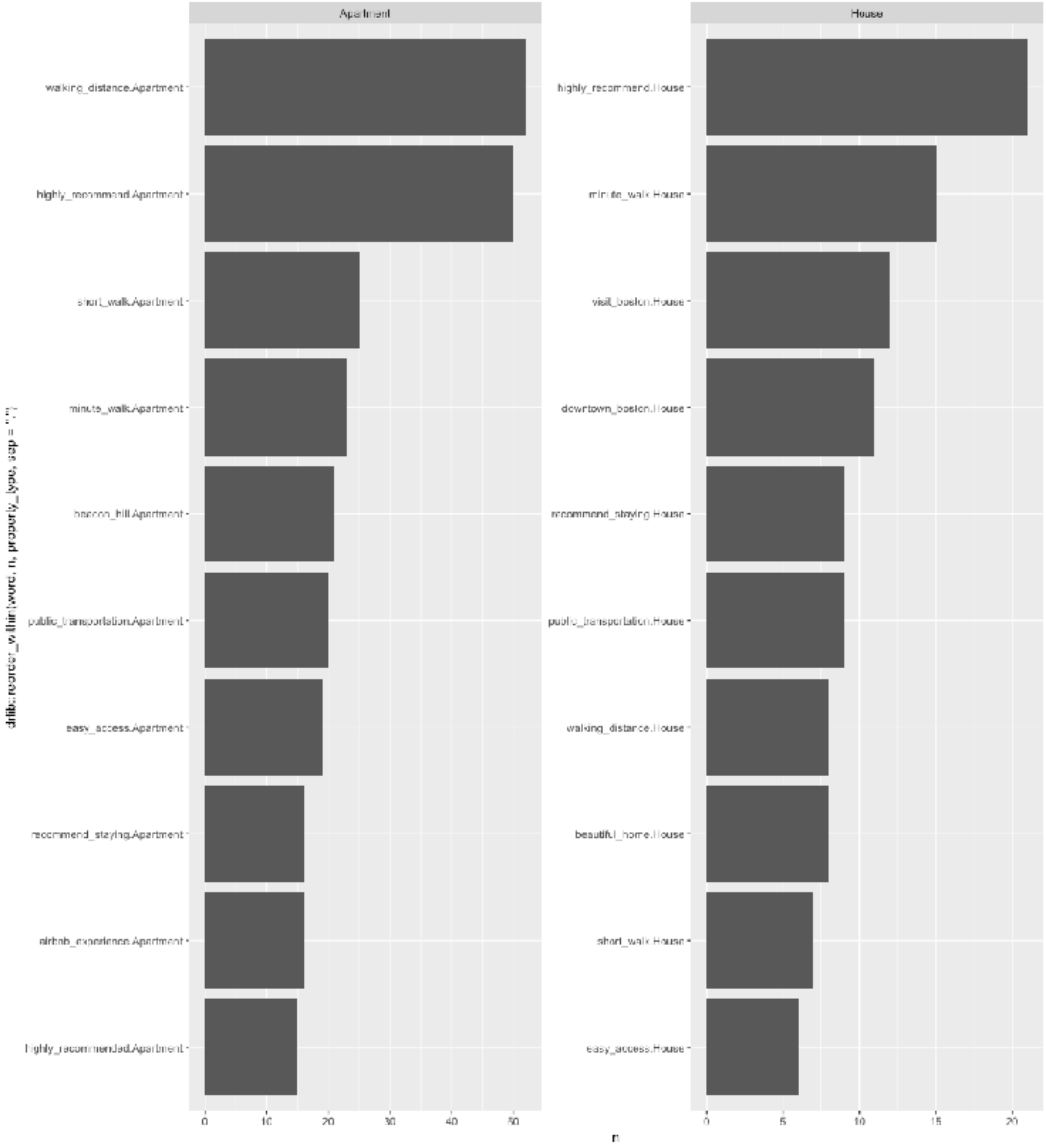
TERM FREQUENCY OF AIRBNB REVIEWS

```r
# plot top 10 term frequencies for Apartments vs. House
reviews %>%
  select(property_type, comments) %>%
  filter(property_type %in% c("Apartment", "House")) %>%
  unnest_tokens(word, comments, token = "ngrams", n = 2) %>%
  separate(word, into = c("word1", "word2"), sep = " ") %>%
  filter(
    !word1 %in% stop_words$word,
    !word2 %in% stop_words$word
  ) %>%
  unite(word, word1, word2) %>%
  count(property_type, word, sort = TRUE) %>%
  group_by(property_type) %>%
  top_n(10) %>%
  ggplot(aes(drlib::reorder_within(word, n, property_type, sep = "."), n)) +
  geom_col() +
  facet_wrap(~ property_type, scales = "free") +
  coord_flip()
```

What is this doing?
Walk me through each step.

TERM FREQUENCY OF AIRBNB REVIEWS

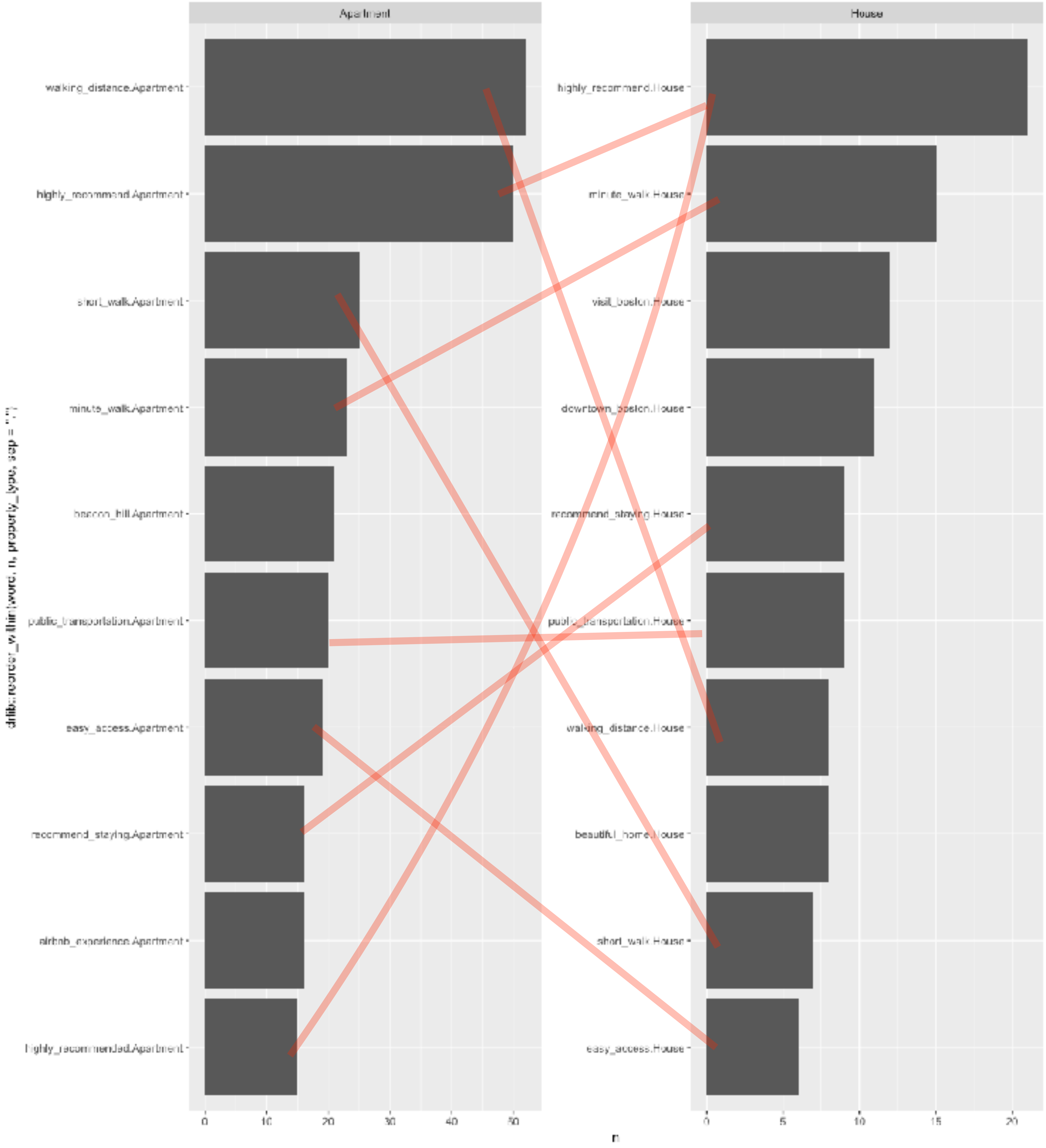
```r
# plot top 10 term frequencies for Apartments vs. House
reviews %>%
  select(property_type, comments) %>%
  filter(property_type %in% c("Apartment", "House")) %>%
  unnest_tokens(word, comments, token = "ngrams", n = 2) %>%
  separate(word, into = c("word1", "word2"), sep = " ") %>%
  filter(
    !word1 %in% stop_words$word,
    !word2 %in% stop_words$word
  ) %>%
  unite(word, word1, word2) %>%
  count(property_type, word, sort = TRUE) %>%
  group_by(property_type) %>%
  top_n(10) %>%
  ggplot(aes(drlib::reorder_within(word, n, property_type, sep = "."), n)) +
  geom_col() +
  facet_wrap(~ property_type, scales = "free") +
  coord_flip()
```



TERM FREQUENCY OF AIRBNB REVIEWS

Frequently used words used throughout all reviews pop up:

- “highly recommended”
- “walking distance”
- “public transportation”
- etc.



YOUR TURN!

Compute the tf-idf for Airbnb bigrams and compare to the previous slides top 10 term frequencies?

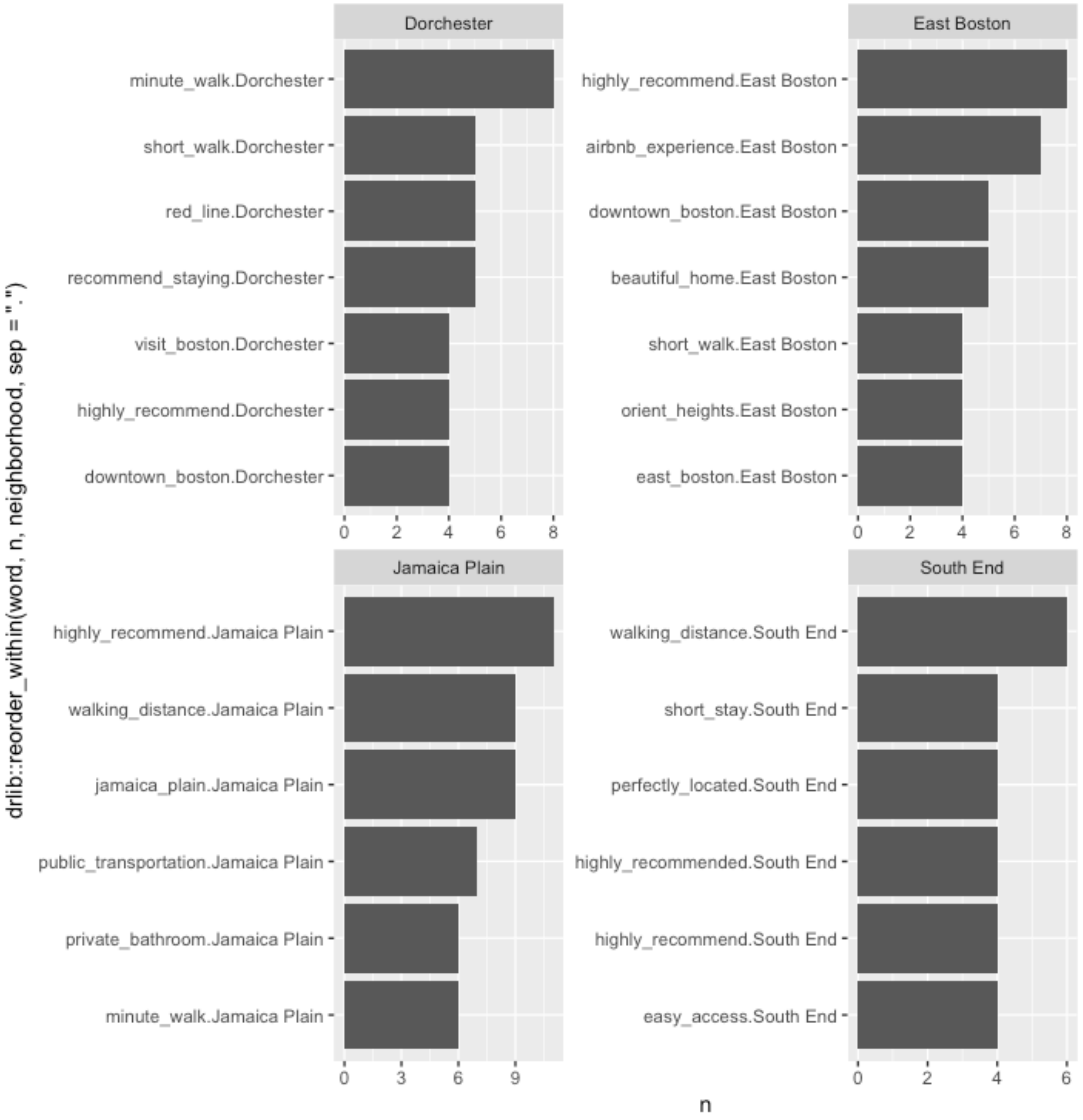
TOP 4 REVIEWED BOSTON NEIGHBORHOODS

```
# plot top 10 term frequencies for Neighborhoods
reviews %>%
  count(neighbourhood_cleansed, sort = TRUE)
# A tibble: 25 x 2
   neighbourhood_cleansed     n
   <chr>                  <int>
 1 Jamaica Plain            106
 2 South End                 94
 3 Dorchester                91
 4 East Boston               76
 5 Charlestown               75
 6 South Boston              66
 7 Beacon Hill               64
 8 Back Bay                  58
 9 Allston                   51
10 North End                 51
# ... with 15 more rows
```

What if we want to understand the unique differences between the top 4 most reviewed neighborhoods?

TERM FREQUENCY OF BOSTON NEIGHBORHOODS

```r
# plot top 10 term frequencies for Neighborhoods
reviews %>%
  select(neighborhood = neighbourhood_cleansed, comments) %>%
  filter(neighborhood %in% c("Jamaica Plain", "South End",
                                       "Dorchester", "East Boston")) %>%
  unnest_tokens(word, comments, token = "ngrams", n = 2) %>%
  separate(word, into = c("word1", "word2"), sep = " ") %>%
  filter(
    !word1 %in% stop_words$word,
    !word2 %in% stop_words$word
  ) %>%
  unite(word, word1, word2) %>%
  count(neighborhood, word, sort = TRUE) %>%
  group_by(neighborhood) %>%
  top_n(5) %>%
  ggplot(aes(drlib::reorder_within(word, n, neighborhood, sep = "."), n)) +
  geom_col() +
  facet_wrap(~ neighborhood, scales = "free") +
  coord_flip()
```



YOUR TURN!

Compute the tf-idf for these reviews and compare to the previous slides top 10 term frequencies?

TAKE-AWAY

When doing exploratory analysis with term frequency:

- Don’t just rely on most commonly used words as the more common a word is throughout a corpus the less meaningful it is (Zipf’s Law).
- Comparing term frequency with tf-idf can provide you insights into what is unique about a particular sub-group of a corpus.

WORD RELATIONSHIPS

Finding and visualizing relationships between words

ADDITIONAL PACKAGE PREREQUISITE

```
library(tm)                # document term matrix and word association
library(widyr)             # word association
library(igraph)            # creating word networks
library(ggraph)            # creating word networks
```

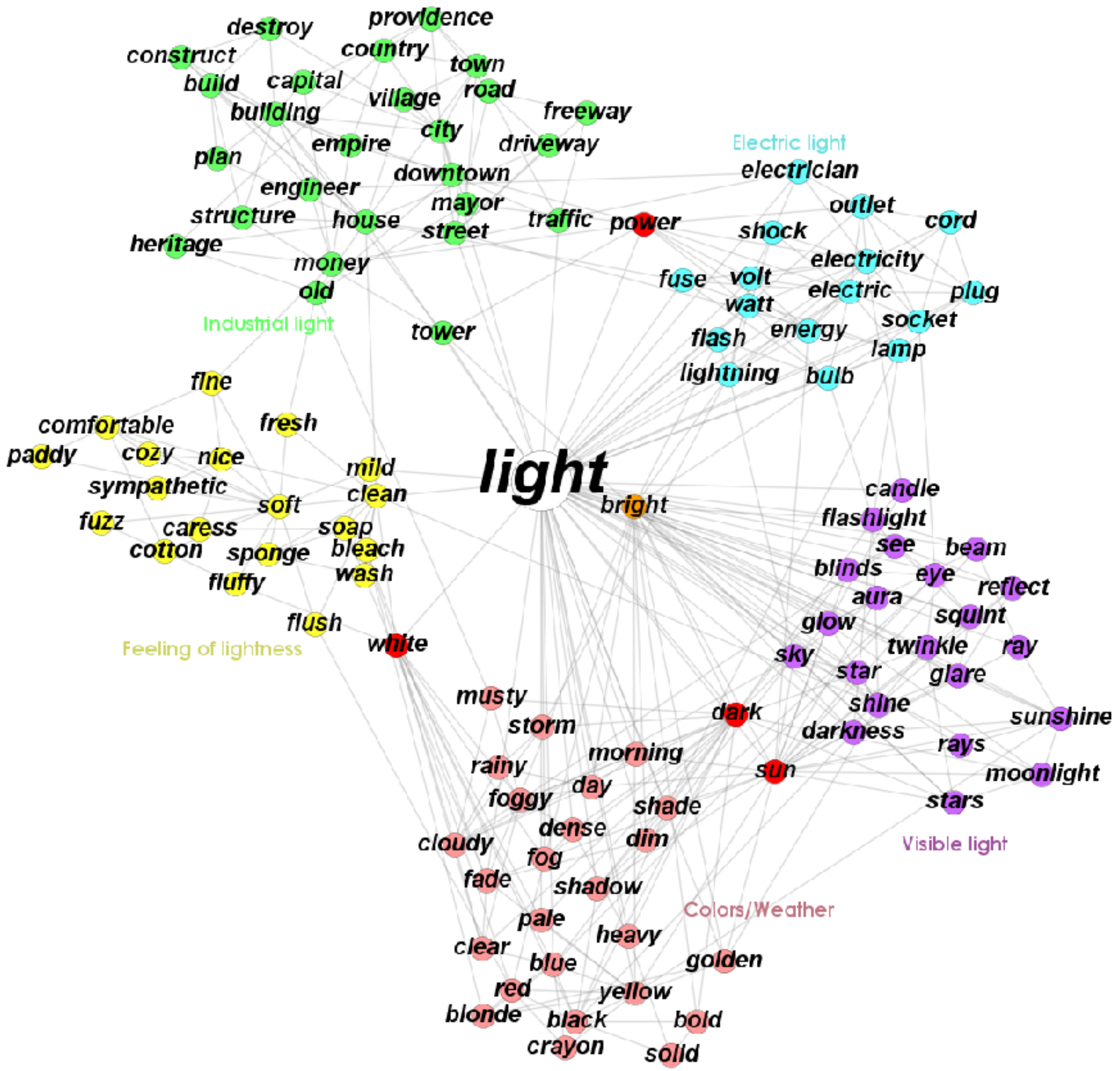
WORD NETWORKS

Benefit: Illustrates connections between words

- Relationship strength
- Term cohesion
- Potential subgroups
- Key influencers

Disadvantage: curse of dimensionality

- *p*: Poor with term diversity
- *n*: Computationally inefficient as observations grow



WORD ASSOCIATION

```
ps_dtm <- ps_df %>%
  unnest_tokens(word, text) %>%
  count(chapter, word) %>%
  cast_dtm(chapter, word, n)

as.matrix(ps_dtm)[1:10, 1:9]
    Terms
Docs    a able about above across act acting admiring affect
  1   112    2    14     1      2   1      1        1      1
  2    93    2    11     0      0   0      1        0      0
  3    73    1     6     0      2   0      0        0      0
  4   110    1    13     0      0   0      0        0      0
  5   178    0    14     2      2   0      0        0      0
  6   140    2    15     0      1   0      0        0      0
  7   122    0    15     1      5   0      0        0      0
  8    75    1    10     0      2   0      0        0      0
  9   108    1    21     0      1   0      0        1      0
  10   92    0    19     1      1   0      0        0      0
```

- **DTM** Document term matrix
  - A common approach to hold text data to perform modeling
  - Each row is a document in our corpus
  - Each column is an ngram
  - Each element is the count of that ngram in the particular document.

WORD ASSOCIATION

```
ps_dtm <- ps_df %>%
  unnest_tokens(word, text) %>%
  count(chapter, word) %>%
  cast_dtm(chapter, word, n)

tm::findAssocs(ps_dtm, "wand", .9)
$wand
      feather        inches         knuts       archway
         0.93          0.93          0.93          0.92
       bronze      malkin's             2           382
         0.92          0.92          0.91          0.91
     adalbert    apothecary        armpit      arsenius
         0.91          0.91          0.91          0.91
       awaits     awkwardly           b.c       bagshot
         0.91          0.91          0.91          0.91
        banks           bar       barrels     bartender
         0.91          0.91          0.91          0.91
        basic      bathilda     beechwood      befuddle
         0.91          0.91          0.91          0.91
```

- We can use this to find words highly associated (correlated) with one another.
- `tm::findAssocs`
  - term of interest
  - correlation limit

This can tell you which words have similar variance in word usage across <u>all documents</u>.

YOUR TURN!

Find the words most correlated with "izzy" in the Airbnb comments variable.

hint: you will need to lower the correlation limit

WORD ASSOCIATION

- But what if we don’t want to pre-specify the words?
- Or we want to identify all word pairs that have a certain correlation limit?

WORD ASSOCIATION

```
ps_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

# A tibble: 28,585 x 2
   chapter word
     <int> <chr>
 1       1 boy
 2       1 lived
 3       1 dursley
 4       1 privet
 5       1 drive
 6       1 proud
 7       1 perfectly
 8       1 normal
 9       1 people
10       1 expect
# ... with 28,575 more rows
```

- Going back to our simple tidied structure…

WORD ASSOCIATION

```
ps_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  pairwise_cor(word, chapter, sort = TRUE)

# A tibble: 29,381,820 x 3
   item1     item2    correlation
   <chr>     <chr>          <dbl>
 1 tantrum   director        1.00
 2 tyke      director        1.00
 3 silly     director        1.00
 4 lunchtime director        1.00
 5 disturb   director        1.00
 6 nephew    director        1.00
 7 hugged    director        1.00
 8 beady     director        1.00
 9 streets   director        1.00
10 lemon     director        1.00
# ... with 29,381,810 more rows
```

- Going back to our simple tidied structure…
- We can use `widyr::pairwise_cor` to identify all word pair correlations.

Unfortunately, low frequency words will often have very high correlation.

WORD ASSOCIATION

```
ps_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  group_by(word) %>%
  filter(n() >= 50) %>%
  pairwise_cor(word, chapter) %>%
  filter(!is.na(correlation))

# A tibble: 1,640 x 3
   item1    item2 correlation
   <chr>    <chr>       <dbl>
 1 dursley  boy         0.299
 2 people   boy        -0.161
 3 dursleys boy         0.604
 4 dudley   boy         0.685
 5 potter   boy         0.165
 6 house    boy         0.387
 7 cloak    boy        -0.0154
 8 floor    boy         0.566
```

- We can filter out lower frequency words to reduce the number of observations
- Now we have commonly used word associations.

YOUR TURN!

Find all word pairs that are frequently used (≥ 50) and highly correlated (> .80).

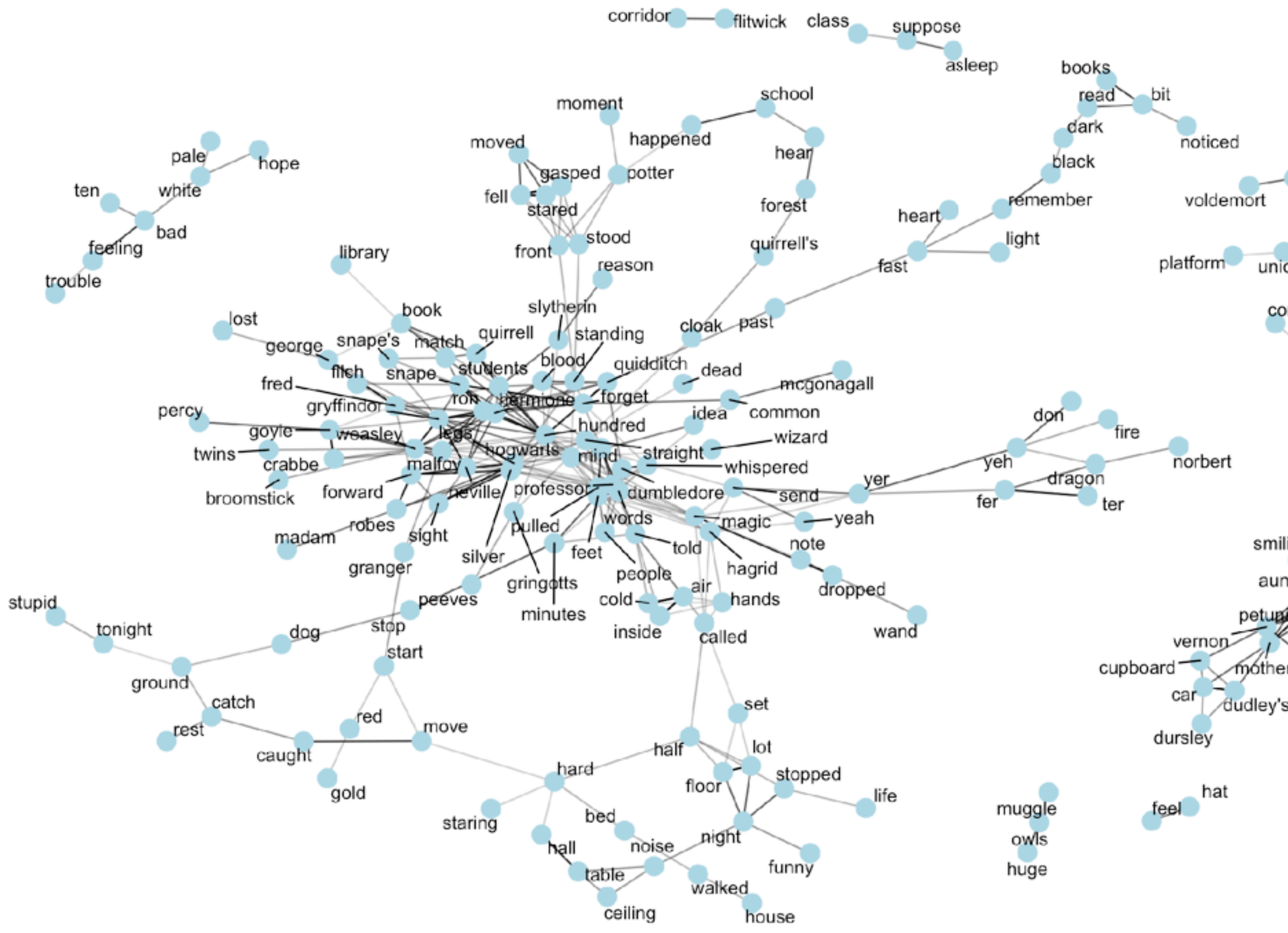
WORD ASSOCIATION NEWORK

```
ps_network <- ps_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, chapter) %>%
  filter(
    !is.na(correlation),
    correlation > .65
    )
```

- We can easily add onto this to develop a word network.
- First, let's find all words that are used more than 20 times and have a correlation of .65 or higher.
- Second, we'll use this info to plot a word network graph.

WORD ASSOCIATION NEWORK

```
library(igraph)
library(ggraph)

set.seed(123)

ps_network %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(
    aes(edge_alpha = correlation),
    show.legend = FALSE
    ) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```



YOUR TURN!

Using the Airbnb review data:

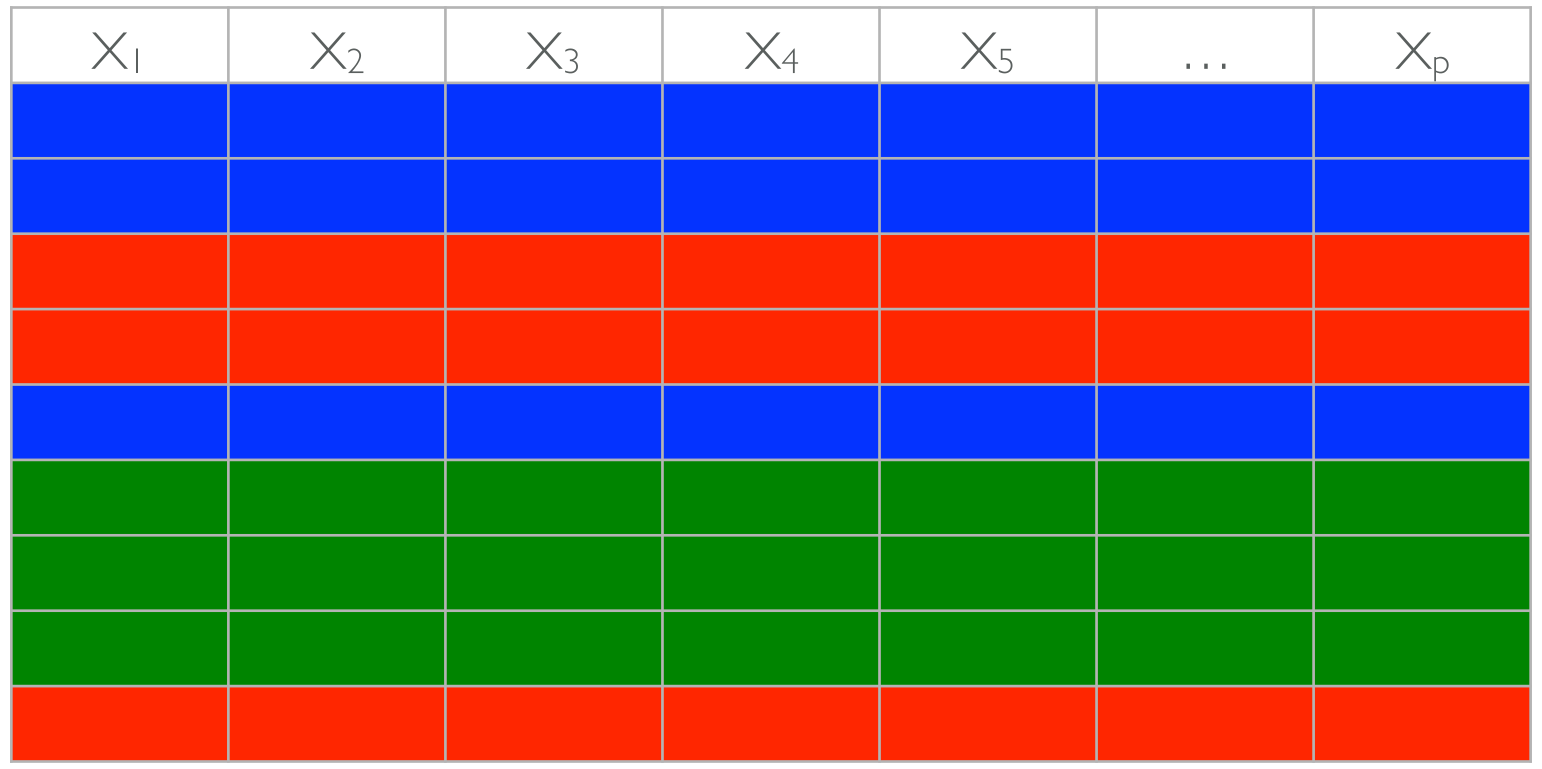
1. *Unnest the comment text for each reviewer_id (unigram)*
2. *Filter out words that are only used once*
3. *Compute pairwise correlation at the reviewer_id level*
4. *Filter for just those pairwise words with correlation > .80*
5. *Create a word network plot*

CLUSTER ANALYSIS

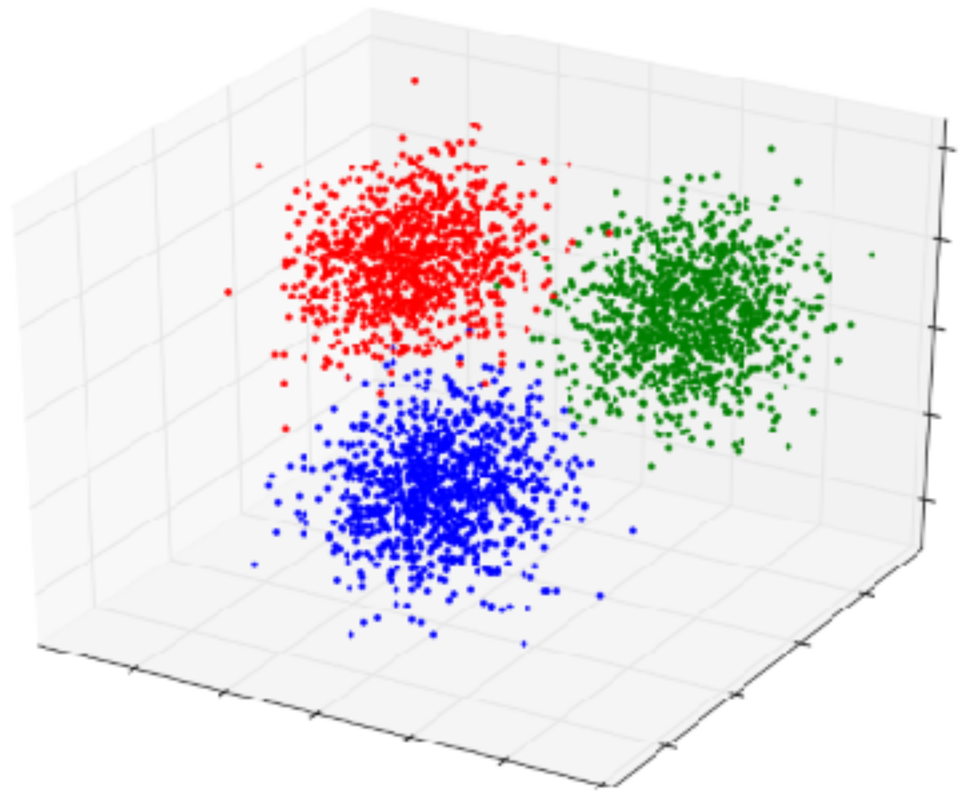
Finding common subgroups

THE IDEA

- **Clustering** refers to a very broad set of techniques for finding <u>*subgroups*</u> in a data set.



- Aggregates "similar" observations into groups such that $k < n$.



- Goal: minimize within group variance, maximize between group variance

TYPES OF CLUSTERING

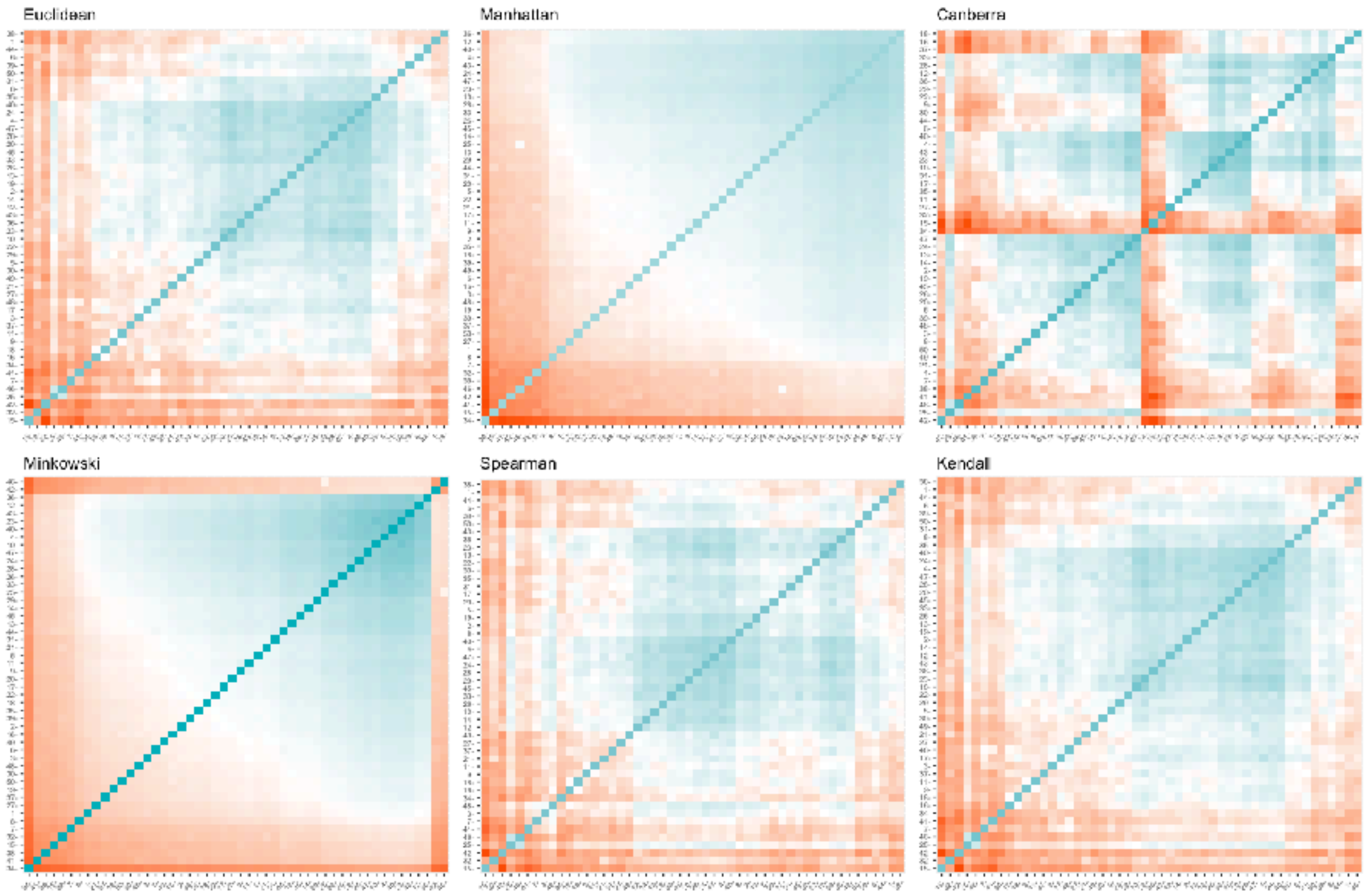
- Many types of clustering algorithms exist
- More common approaches:
  - K-means
  - Hierarchical
  - Partition around mediods (PAM)
- Less common and more domain specific
  - Spherical

Primary difference revolves around the mechanism used to partition the data

DISTANCE MEASURES

- Once the data is partitioned, ***distance measures*** are used to measure within and between cluster variability.

- Multiple distance measures can be used



DISTANCE MEASURES

- Once the data is partitioned, ***distance measures*** are used to measure within and between cluster variability.
- Multiple distance measures can be used
- Euclidean distance is by far the most common

$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Others
  - Manhattan
  - Pearson
  - Spearman
  - etc.

DISTANCE MEASURES

- Once the data is partitioned, ***distance measures*** are used to measure within and between cluster variability.
- Multiple distance measures can be used
- Euclidean distance is by far the most common

$$d_{euc}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Others
  - Manhattan
  - Pearson
  - Spearman
  - etc.

ADDITIONAL PACKAGE PREREQUISITE

```
library(factoextra)         # cluster analysis
library(skmeans)            # cluster analysis
library(cluster)            # cluster analysis
library(clue)               # cluster analysis
```

ADDITIONAL DATA USED

- For this section we'll use the following resume data for our examples

```
# resume files
url <- "https://raw.githubusercontent.com/kwartler/text_mining/master/1yr_plus_final4.csv"
resumes <- read_csv(url)

resumes
# A tibble: 50 x 2
     num text
   <int> <chr>
 1     1 "Responsible for handling large cash amounts on a daily basis for many different types of uni…
 2     2 "\x82    Attends Amazon Summit Training in Seattle, WA\xa0\x82    Host 2 events on campus per…
 3     3 "~ target dot com\xa0\xa0Independently maintain the shoe department, customer service, stock …
 4     4 "\xa0Assisting customers with their online orders via phone calls and chat. General customer …
 5     5 "Mentor in training new hires.\xa0Mentored other team members to multitasking and obtain team…
 6     6 "\xa0Assist customers with any inquires about their order and or purchase received. Assist cu…
 7     7 "Managed a group of 20+ individual contracted guest service team members in a call center env…
 8     8 "\xa0    Mentor and counselor for the youth.\xa0    Assisted supervisor with memorandums\xa0 …
```

SETTING UP THE DATA

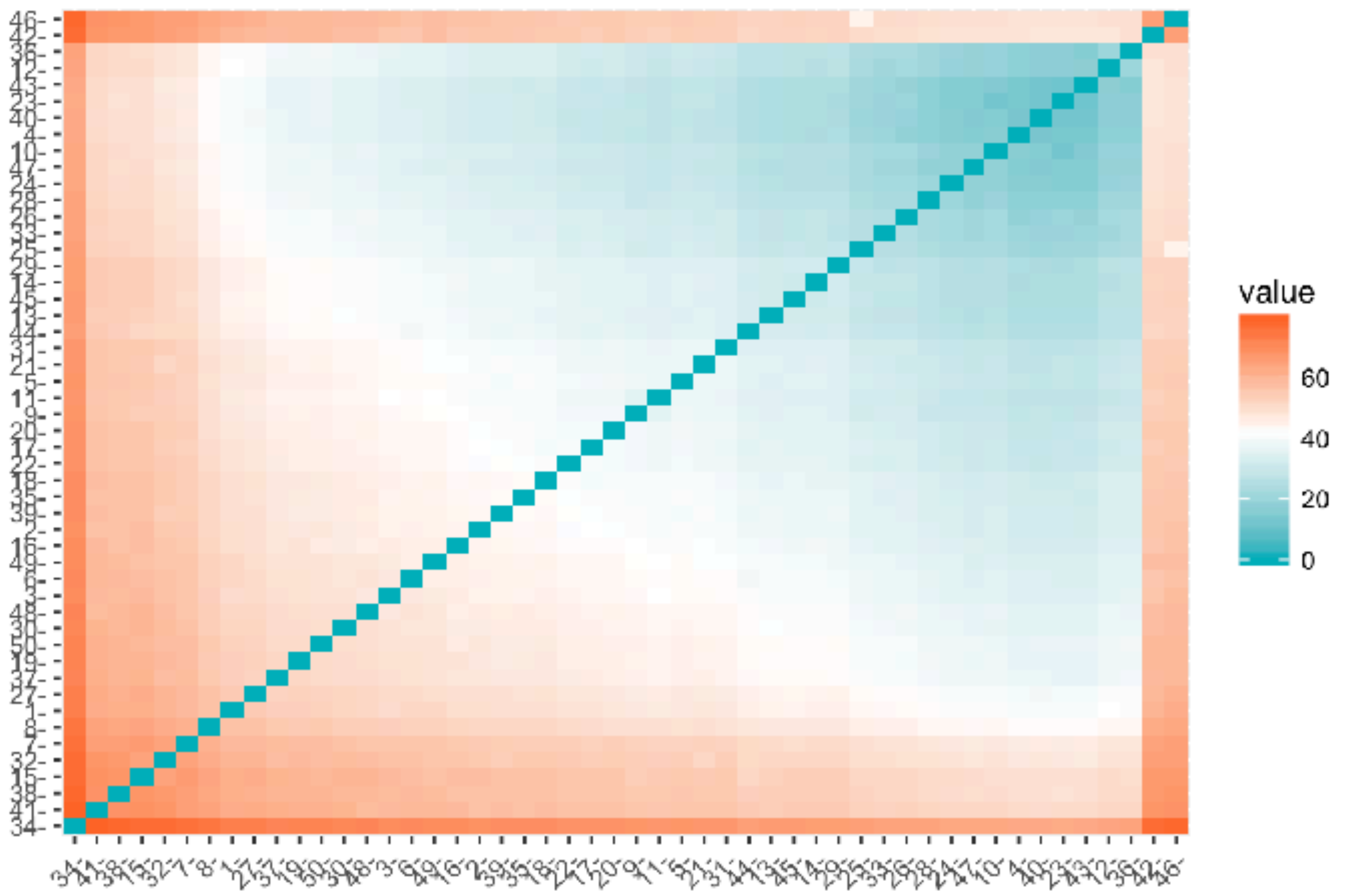
```
resumes_dtm <- resumes %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  filter(!str_detect(word, "[[:digit:]]")) %>%
  count(num, word) %>%
  cast_dtm(num, word, n) %>%
  scale()

# what does our dtm look like?
dim(resumes_dtm)
[1]  50 980

resumes_dtm[1:5, 1:4]
    Terms
Docs     amounts associates   attitude    bankers
   1   6.9296465  2.7021640  4.8497423  6.9296465
   2   0.1414214  0.2349708  0.2020726  0.1414214
```

- **DTM** A DTM sets up our data structure
- **scale** normalizes our word counts
- The result is an N x P matrix:
  - Rows = documents
  - Columns = words
  - Values = (x - mean(x)) / sd(x)

MEASURING SIMILARITY

```
distance <- get_dist(resumes_dtm)

fviz_dist(
  distance,
  gradient = list(
    low = "#00AFBB",
    mid = "white",
    high = “#FC4E07"
    )
  )
```

- **get_dist** measures the similarity between observations (default: Euclidean)
- **fviz_dist** plots distance measures



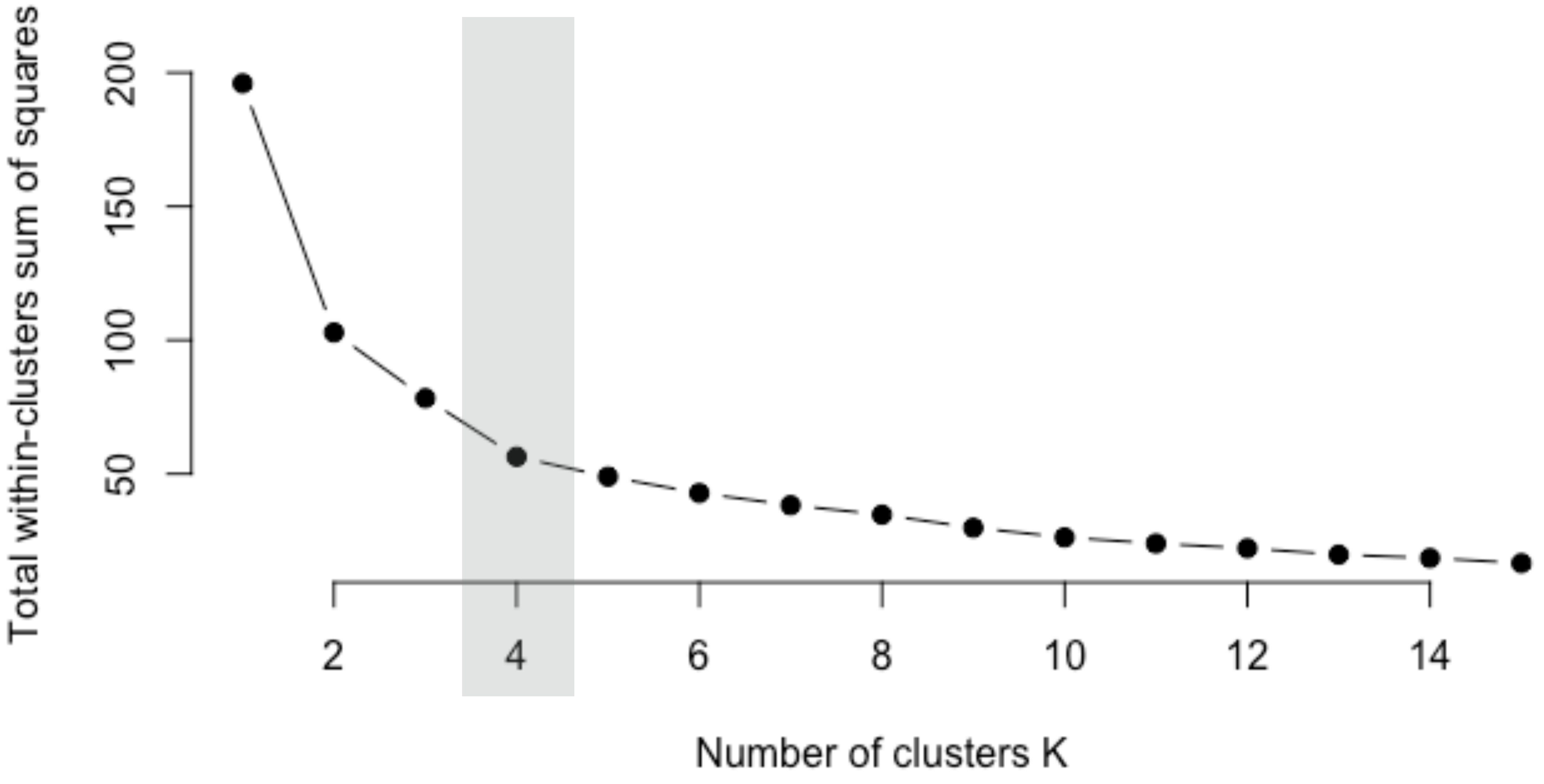
APPLYING K-MEANS

```
k3 <- kmeans(resumes_dtm, centers = 3, nstart = 25)

str(k3)
List of 9
 $ cluster     : Named int [1:50] 3 3 3 3 3 3 3 ...
  ..- attr(*, "names")= chr [1:50] "1" "2" "3" ...
 $ centers     : num [1:3, 1:980] -0.14142 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:980] "amounts" "associates" ...
 $ totss       : num 48020
 $ withinss    : num [1:3] 0 0 41887
 $ tot.withinss: num 41887
 $ betweenss   : num 6133
 $ size        : int [1:3] 1 1 48
 $ iter        : int 3
```

- **kmeans** performs k-means clustering (default: Euclidean)
  - User specifies k (centers)
  - nstart > 1 allows convergence

Extremely misbalanced
Maybe wrong k?

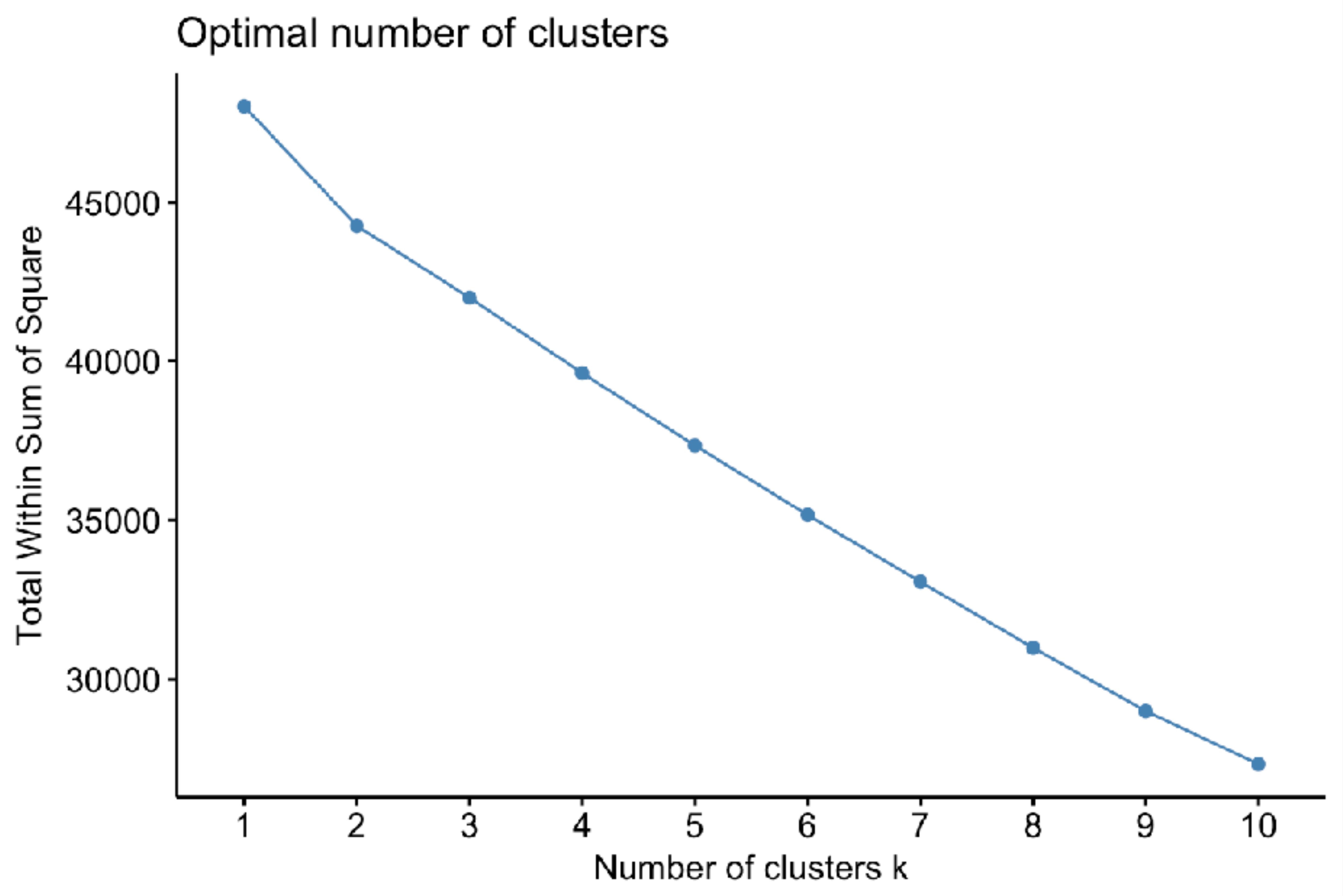
IS THERE AN OPTIMAL K?



- Multiple approaches to identify preferred k
- **WSS** extracts the within-cluster sum of squared differences
  - Look for the bend where there are diminishing returns.

IS THERE AN OPTIMAL K?

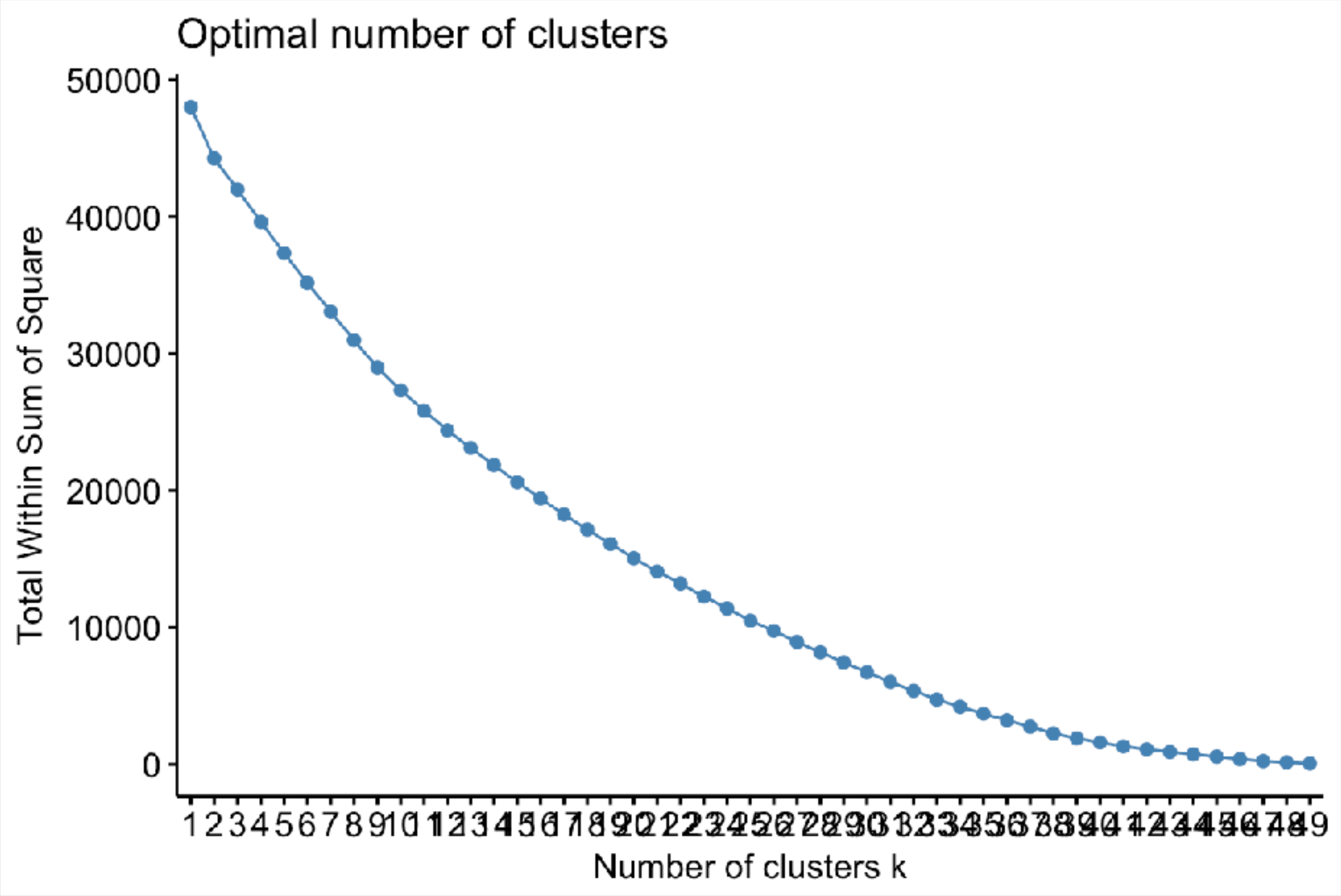
```
fviz_nbclust(resumes_dtm, kmeans, method = "wss")
```



- We can do this easily with `fviz_nbclust`
- Specify `method = “wss”`
- However, our results do not show a diminishing return

IS THERE AN OPTIMAL K?

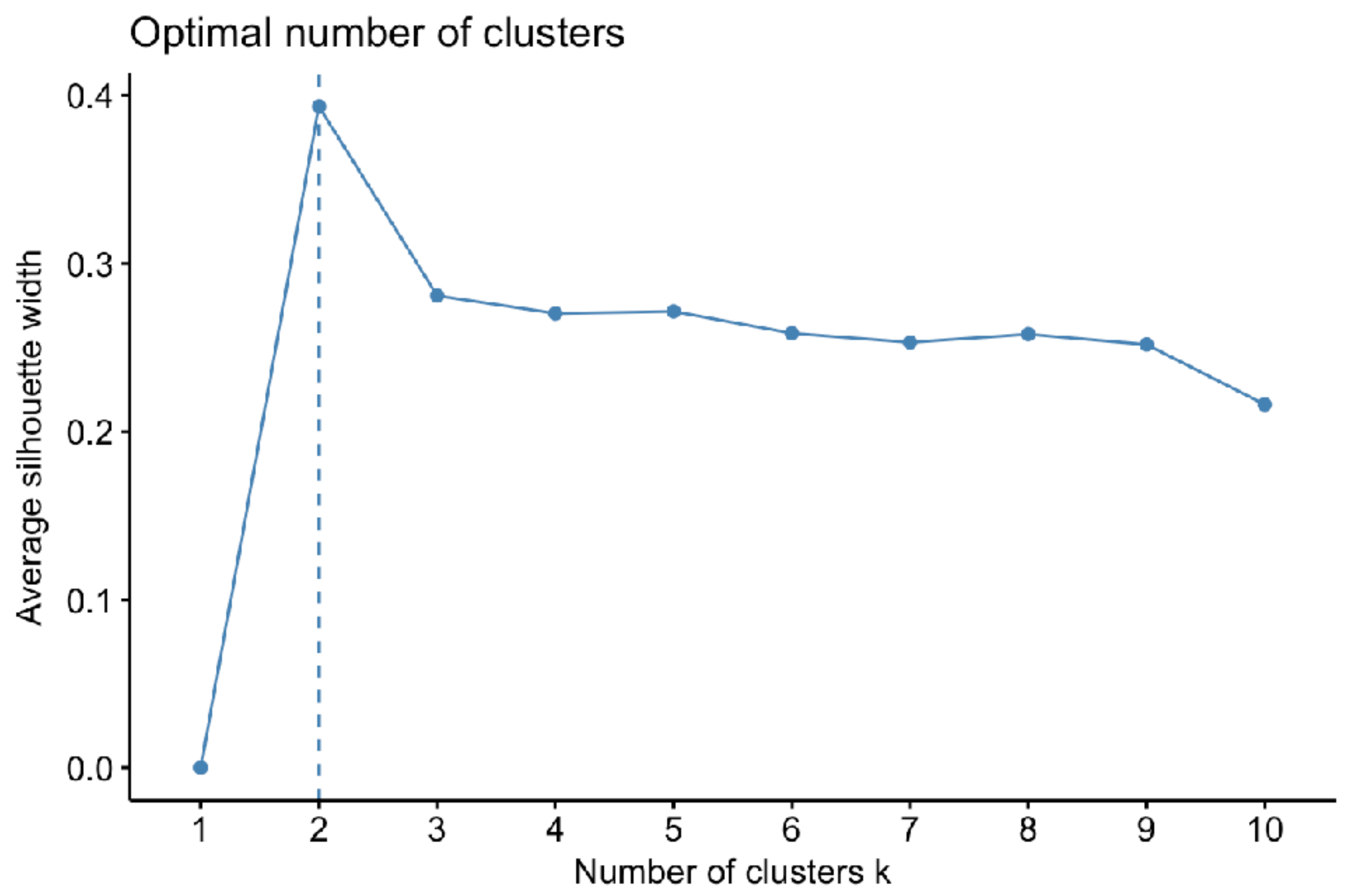
```
fviz_nbclust(resumes_dtm, kmeans, method = "wss",
             k.max = 49)
```



- We can do this easily with `fviz_nbclust`
- However, our results do not show a diminishing return
- We don't find diminishing returns until close to $k = n - 1$

IS THERE AN OPTIMAL K?

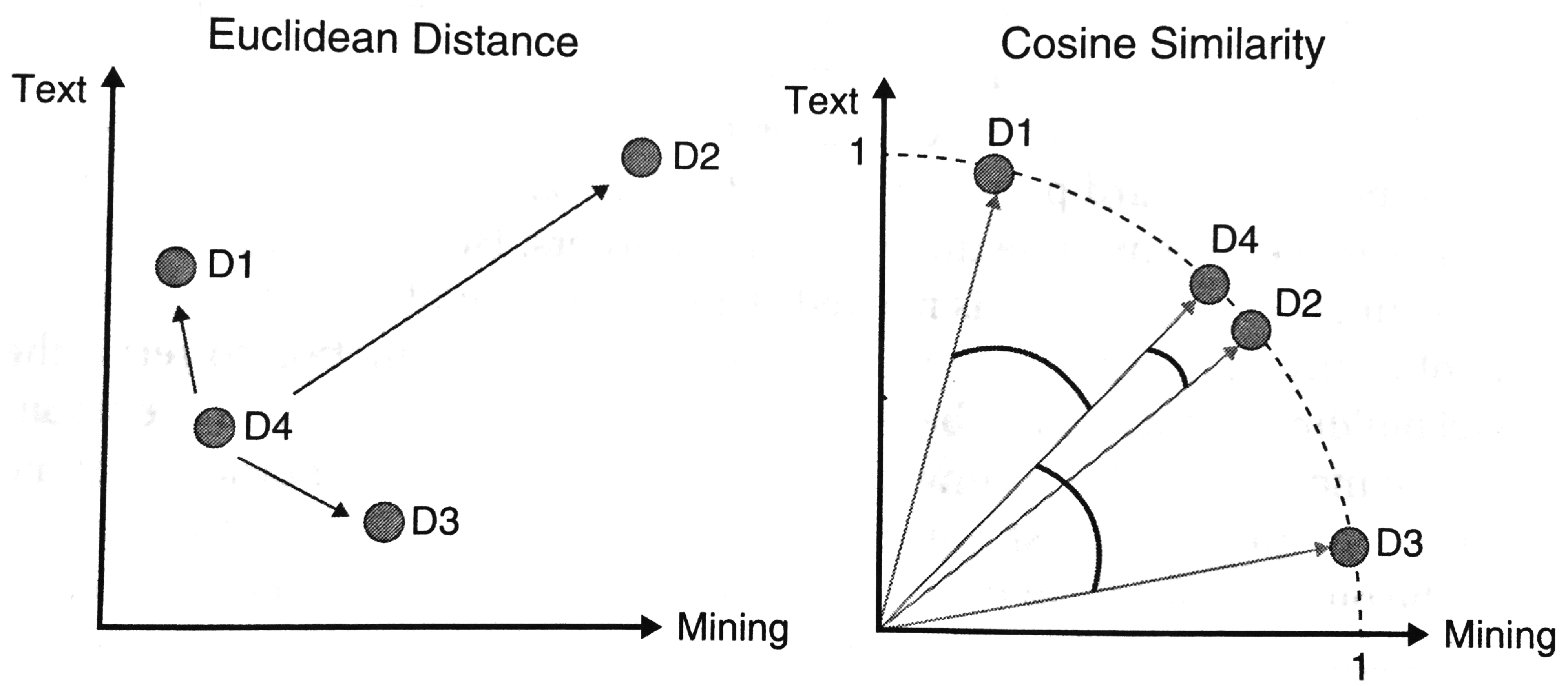
```
fviz_nbclust(resumes_dtm, kmeans,
             method = "silhouette")
```



- **Silhouette** is an alternative measure to assess optimal k
  - In short, silhouette method assesses the quality of each clusters by assessing how well each object lies within the cluster.
  - It does so across all values of k and the k with the highest avg silhouette is preferred

ANOTHER OPTION?

- Our results suggest that k-means is not doing a very good job of finding subgroups.
- This can be common with text data because they are often very sparse and traditional clustering techniques do not do very well with sparse data.
- ***Spherical k-means*** clustering handles sparse data very well
- Uses the cosine of the angle for the distance measure



APPLYING SPHERICAL K-MEANS

```
sk3 <- skmeans(
  resumes_dtm,
  k = 3,
  m = 1.2,
  control = list(nruns = 5, verbose = TRUE)
  )

table(sk3$cluster)
 1  2  3
11 14 25
```

- **skmeans** performs spherical k-means clustering
  - *k = clusters*
  - *m = "fuzzification parameter"*
  - *nruns = convergence*

We're finding more balanced results

AN ALTERNATIVE SILHOUETTE PLOT

```
silhouette(sk3) %>% plot()
```

- This is an alternative silhouette plot (actually, the more common one)
  - Three defined clusters
  - Cluster silhouette width
  - Overall avg silhouette width

Remember, the goal is to maximize average silhouette width

LET'S OPTIMIZE *K* BASED ON SILHOUETTE

```
tuning_grid <- expand.grid(
  k = 2:10,
  m = seq(1, 2, by = 0.1),
  silhouette = NA
)
    k   m silhouette
1   2 1.0         NA
2   3 1.0         NA
3   4 1.0         NA
4   5 1.0         NA
5   6 1.0         NA
6   7 1.0         NA
7   8 1.0         NA
8   9 1.0         NA
9  10 1.0         NA
10  2 1.1         NA
```

- First, we'll create a tuning grid

LET'S OPTIMIZE *K* BASED ON SILHOUETTE

```
for (i in 1:nrow(tuning_grid)) {
  model <- skmeans(
    resumes_dtm, tuning_grid[i, 1],
    m = tuning_grid[i, 2],
    control = list(nruns = 5))

  tuning_grid[i, 3] <- median(silhouette(model)[, 3])
}

tuning_grid %>% filter(silhouette == max(silhouette))
  k   m silhouette
1 2 1.1 0.02155586
2 2 1.2 0.02155586
```

- Second, we loop through and
  - apply **skmeans** for each k and m combination
  - compute avg silhouette
- Now we can filter for the tuning parameters that maximize avg silhouette

LET'S OPTIMIZE *K* BASED ON SILHOUETTE

```
sk2 <- skmeans(
  resumes_dtm,
  k = 2,
  m = 1.2,
  control = list(nruns = 5, verbose = TRUE)
  )

table(sk2$cluster)
 1  2
32 18
```

- Reapply skmeans with optimal k and m

DESCRIBING THE CLUSTERS

```
sk2_results <- t(cl_prototypes(sk2))
sort(sk2_results[, 1], decreasing = TRUE)[1:10]
   satisfying       inbound        taking          home
   0.09469097    0.09314427    0.08734867    0.08730496
     emailing    assistance         calls        adding
   0.08099781    0.07593254    0.05976378    0.05342765
   cancelling discrepancies
   0.05342765    0.05022206

sort(sk2_results[, 2], decreasing = TRUE)[1:10]
procedures          status          product
     0.10791536      0.09871495      0.09467755
   transactions            team          quality
     0.08815788      0.08602342      0.08442937
          goals           sales         inquires
     0.08302194      0.08265372      0.07951655
```

- We can find the words in each cluster that have the highest "prototype" scores.

CHALLENGE!!

YOUR TURN PART 1

5 minutes

Can you import and combine, the 10 articles for the 10 authors in the data/news_articles folder? The result should look something like:

```
# A tibble: 100 x 2
      id text
   <int> <chr>
 1     1 "The Internet may be overflowing with new technology but crime in cyberspace is still of…
 2     2 "The U.S. Postal Service announced Wednesday a plan to boost online commerce by enhancin…
 3     3 "Elementary school students with access to the Internet learned more than kids who lacke…
 4     4 "An influential Internet organisation has backed away from a proposal to dramatically ex…
 5     5 "An influential Internet organisation has backed away from a proposal to dramatically ex…
 6     6 "A group of leading trademark specialists plans to release recommendations aimed at mini…
 7     7 "When a company in California sells a book to a consumer in Canada from a Web site hoste…
```

YOUR TURN PART 2

5 minutes

Can you now tidy this data set and prepare for cluster analysis? The result should look something like:

```
    Terms
Docs bogus   business commission   consumer  consumers    federal fortuna      fraud   internet
   1   9.9  1.3888483  3.1433921  2.9570146  8.8459214  2.5176840     9.9  8.0698831  4.4343509
   2  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  2.3177156
   3  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915 -0.3280785
   4  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  3.3760332
   5  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  3.9051920
   6  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  0.7302392
   7  -0.1 -0.3803852  8.1329033  2.9570146 -0.1347095  9.1782237    -0.1 -0.1646915  1.7885568
```

YOUR TURN PART 3

5 minutes

Choose any of the cluster analysis approaches and apply. How many clusters do you think are best?

TOPIC MODELING

What are you talking about?

ADDITIONAL PACKAGE PREREQUISITE

```
library(topicmodels)        # topic modeling
library(ldatuning)          # topic modeling
```

LDA

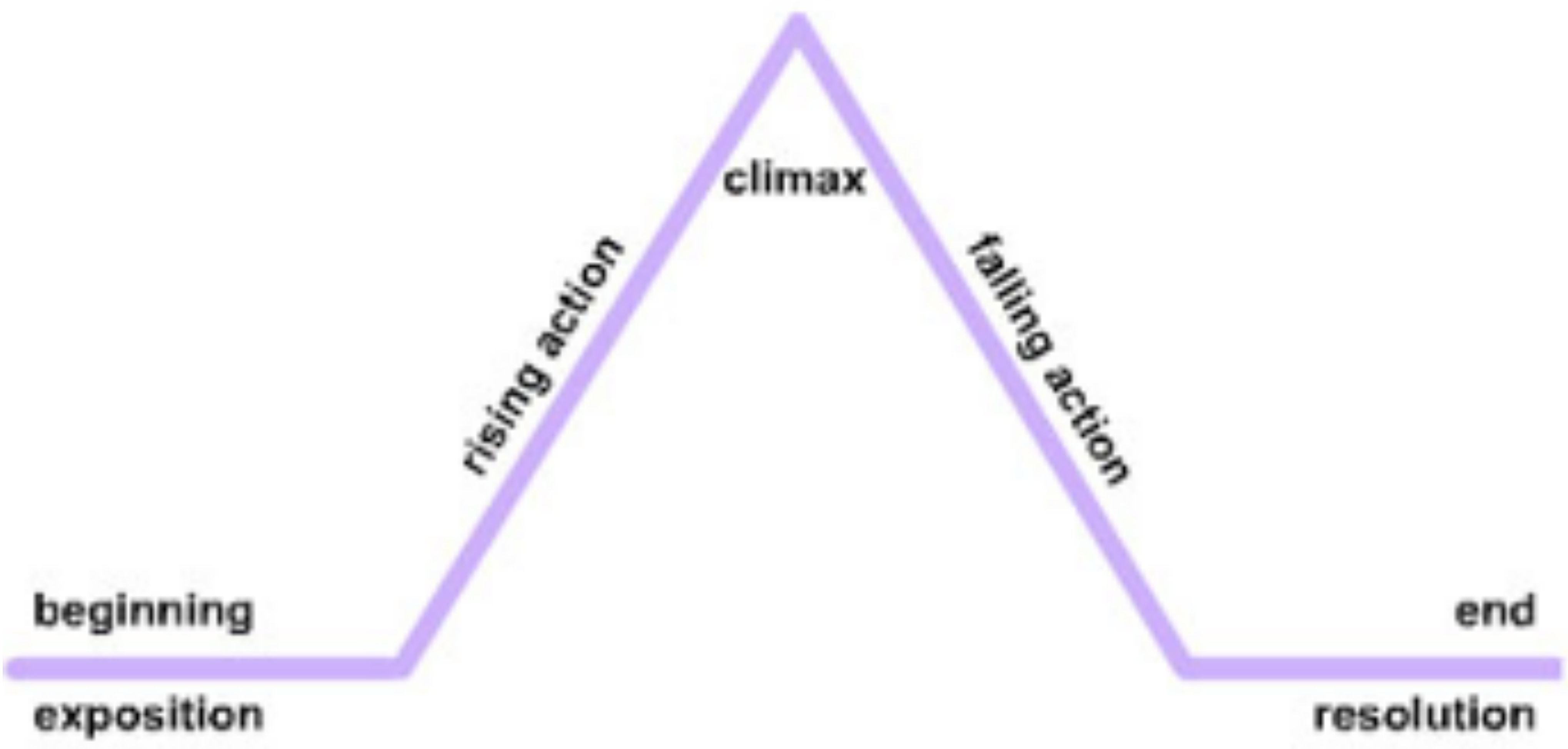
- We often have collections of documents, say store reviews, that we'd like to divide into natural groups so that we can understand them separately.
- Topic modeling is a method for unsupervised classification of such documents, similar to clustering on numeric data, which finds natural groups of items even when we're not sure what we're looking for.
- Latent Dirichlet Allocation (LDA) is a popular method for fitting a topic model.
- We will use the `topicmodels` package to perform LDA

LDA

- Latent Dirichlet Allocation (LDA) is a popular method for fitting a topic model.
  - ***Probability-based*** approach to finding clusters within documents
  - Latent because the identified ***topics are concealed*** and defined by the user
  - ***Dirichlet distributions*** are used in stats to understand multivariate (or in this case multi-word) probability distributions
  - LDA seeks to answer two questions:
    1. Probability of a word being attributed to a particular topic
    2. Probability of a document being attributed to a particular topic

HARRY POTTER TOPIC MODELING

- With the Harry Potter series, each book has its own plot
- However, the different plot points in each book may overlap or be unique
- Topic modeling allows us to look at the entire Harry Potter series and identify unique and/or common themes (topics) that occur.



CREATE THE HARRY POTTER SERIES

```
titles <- c("Philosopher's Stone", "Chamber of Secrets", "Prisoner of Azkaban",
            "Goblet of Fire", "Order of the Phoenix", "Half-Blood Prince",
            "Deathly Hallows")

books <- list(philosophers_stone, chamber_of_secrets, prisoner_of_azkaban,
              goblet_of_fire, order_of_the_phoenix, half_blood_prince,
              deathly_hallows)

series <- tibble()

for(i in seq_along(titles)) {

  clean <- tibble(chapter = seq_along(books[[i]]),
                  text = books[[i]]) %>%
    unnest_tokens(word, text) %>%
    mutate(book = titles[i]) %>%
    select(book, everything())

  series <- rbind(series, clean)
}

series$book <- factor(series$book, levels = rev(titles))
```

This chunk of code creates a data frame that captures every word by chapter by book…

CREATE THE HARRY POTTER SERIES

```
series
# A tibble: 1,089,386 x 3
                  book chapter    word
                <fctr>   <int>   <chr>
 1 Philosopher's Stone       1     the
 2 Philosopher's Stone       1     boy
 3 Philosopher's Stone       1     who
 4 Philosopher's Stone       1   lived
 5 Philosopher's Stone       1      mr
 6 Philosopher's Stone       1     and
 7 Philosopher's Stone       1     mrs
 8 Philosopher's Stone       1 dursley
 9 Philosopher's Stone       1      of
10 Philosopher's Stone       1  number
# ... with 1,089,376 more rows
```

PERFORM LDA

```
series %>%
  anti_join(stop_words) %>%
  unite(document, book, chapter) %>%
  count(document, word)
# A tibble: 215,433 x 3
               document        word     n
                  <chr>       <chr> <int>
 1 Chamber of Secrets_1           1     1
 2 Chamber of Secrets_1 abnormality     1
 3 Chamber of Secrets_1      absent     1
 4 Chamber of Secrets_1      aching     1
 5 Chamber of Secrets_1         age     1
 6 Chamber of Secrets_1         ago     1
 7 Chamber of Secrets_1         aim     1
 8 Chamber of Secrets_1       aimed     1
 9 Chamber of Secrets_1     allowed     1
10 Chamber of Secrets_1    announce     1
# ... with 215,423 more rows
```

Here, we:

- remove stop words
- create a document variable for each book/chapter
- compute term frequency

PERFORM LDA

```
# first we turn into a document term matrix
df_dtm <- series %>%
  anti_join(stop_words) %>%
  unite(document, book, chapter) %>%
  count(document, word) %>%
  cast_dtm(document, word, n)

# LDA across each chapter in the Harry Potter series
levels_lda <- LDA(df_dtm, k = 7, control = list(seed = 1234))
```

Here, we:

- turn this information into a document term matrix
- use `LDA()` to perform an LDA model with k specified topics

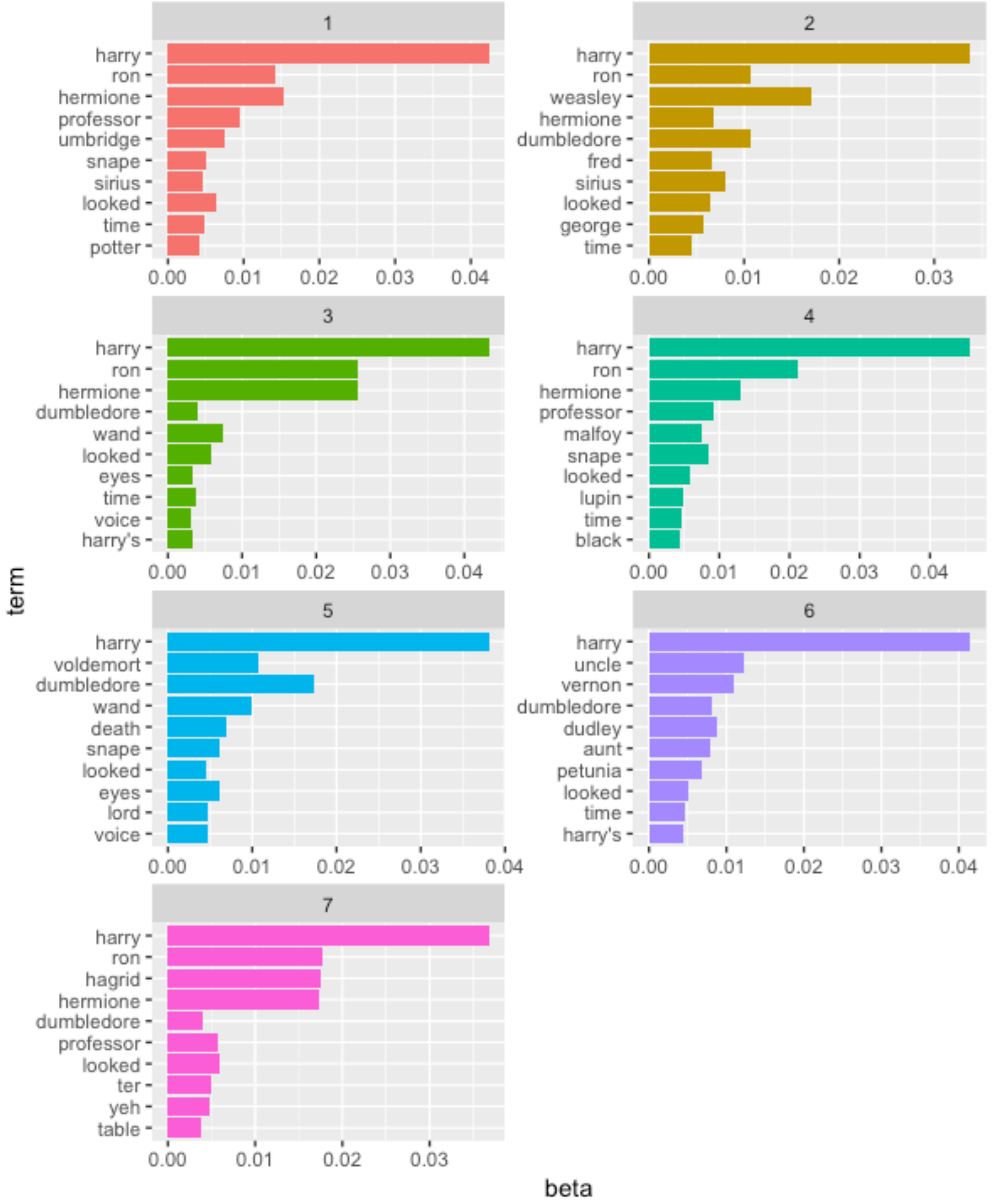
PER TOPIC PER WORD PROBABILITIES

```
levels_topics <- tidy(levels_lda, matrix = "beta")
levels_topics %>%
  arrange(desc(beta))
# A tibble: 166,565 x 3
   topic     term        beta
   <int>    <chr>       <dbl>
 1     4    harry 0.04571602
 2     3    harry 0.04332584
 3     1    harry 0.04240264
 4     6    harry 0.04145501
 5     5    harry 0.03810293
 6     7    harry 0.03681298
 7     2    harry 0.03377472
 8     3      ron 0.02561400
 9     3 hermione 0.02558098
10     4      ron 0.02116254
# ... with 166,555 more rows
```

- There is a .046 (4.6%) probability of "Harry" being generated from topic 1

PER TOPIC PER WORD PROBABILITIES

```
# top 10 terms within each topic
levels_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



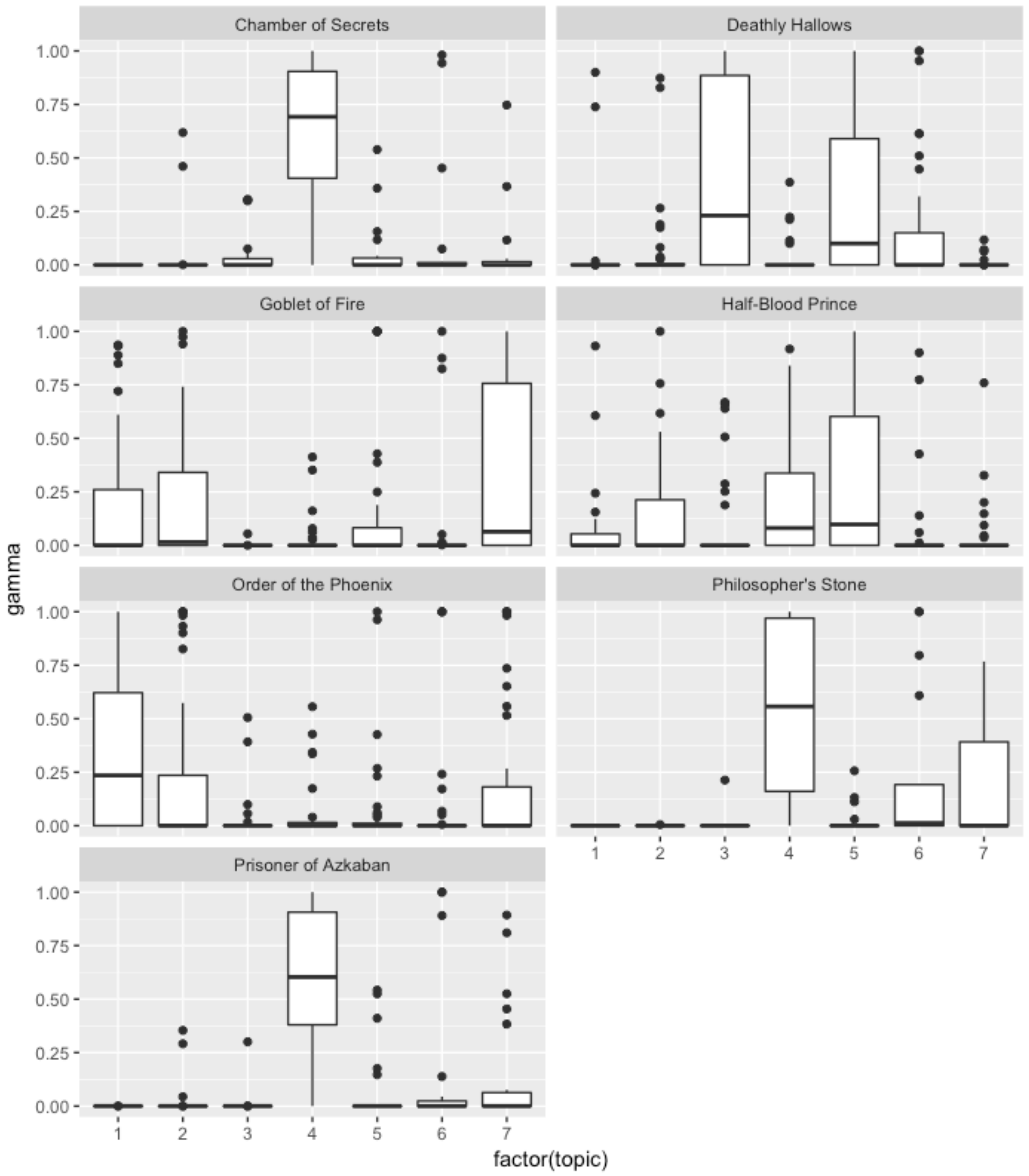
PER DOCUMENT PER TOPIC PROBABILITIES

```
levels_gamma <- tidy(levels_lda, matrix = "gamma")
levels_gamma
# A tibble: 1,400 x 3
                document topic        gamma
                   <chr> <int>        <dbl>
 1  Chamber of Secrets_1     1 3.389664e-05
 2 Chamber of Secrets_10     1 1.664132e-05
 3 Chamber of Secrets_11     1 1.504487e-05
 4 Chamber of Secrets_12     1 1.983007e-05
 5 Chamber of Secrets_13     1 2.985947e-05
 6 Chamber of Secrets_14     1 3.790253e-05
 7 Chamber of Secrets_15     1 2.101221e-05
 8 Chamber of Secrets_16     1 3.609628e-05
 9 Chamber of Secrets_17     1 1.797750e-05
10 Chamber of Secrets_18     1 2.886677e-05
# ... with 1,390 more rows
```

- There is a .00003 probability of chapter 1 of Chamber of Secrets being generated from topic 1

PER DOCUMENT PER TOPIC PROBABILITIES

```
# top 10 terms within each topic
levels_gamma %>%
  separate(document, into = c("book", "chapter"), sep = "_") %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ book, ncol = 2)
```



OPTIMAL NUMBER OF TOPICS

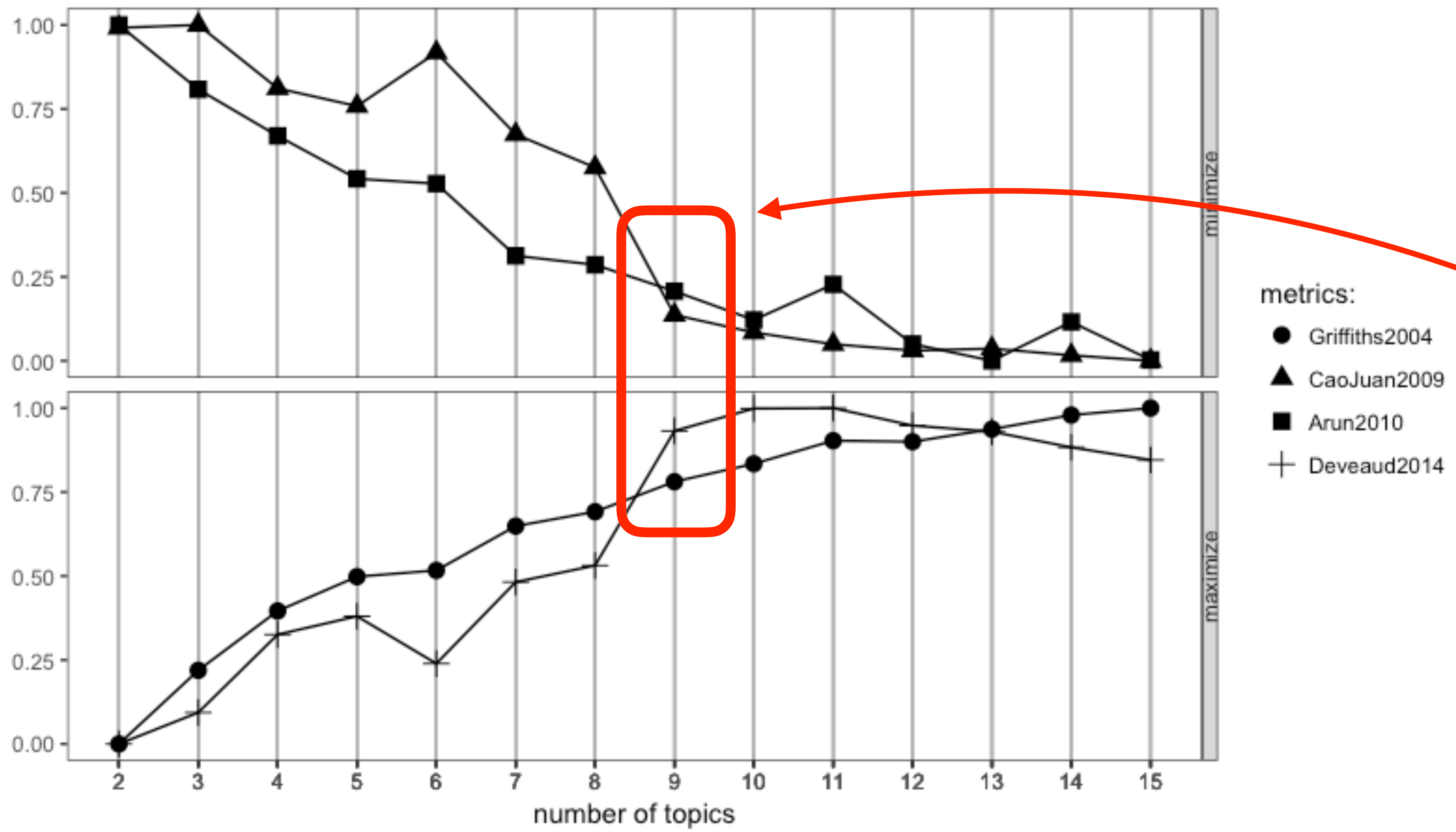
```
# DO NOT RUN IN CLASS ----> takes about 15 min
# find optimal number of topics
install.packages("ldatuning")
library(ldatuning)

result <- FindTopicsNumber(
  df_dtm,
  topics = seq(from = 2, to = 15, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2L,
  verbose = TRUE
)
```

Much like cluster analysis, we can use different metrics to identify preferred number of topics

- `ldatuning`
- Provide a range of possible k values
- Provides the 4 primary metrics used in literature
- Becomes computationally expensive for large data sets

OPTIMAL NUMBER OF TOPICS

```
FindTopicsNumber_plot(result)
```



The goal:

- Convergence across metrics
- A single k that optimizes all/ most metrics or…
- The knee in the curve where we have diminishing returns

CHALLENGE!!

YOUR TURN PART 1

5 minutes

Can you import and combine, the 10 articles for the 10 authors in the data/news_articles folder? The result should look something like:

```
# A tibble: 100 x 2
      id text
   <int> <chr>
 1     1 "The Internet may be overflowing with new technology but crime in cyberspace is still of…
 2     2 "The U.S. Postal Service announced Wednesday a plan to boost online commerce by enhancin…
 3     3 "Elementary school students with access to the Internet learned more than kids who lacke…
 4     4 "An influential Internet organisation has backed away from a proposal to dramatically ex…
 5     5 "An influential Internet organisation has backed away from a proposal to dramatically ex…
 6     6 "A group of leading trademark specialists plans to release recommendations aimed at mini…
 7     7 "When a company in California sells a book to a consumer in Canada from a Web site hoste
```

YOUR TURN PART 2

5 minutes

Can you now tidy this data set and prepare for topic modeling? The result should look something like:

```
    Terms
Docs bogus   business commission   consumer  consumers    federal fortuna      fraud   internet
   1   9.9  1.3888483  3.1433921  2.9570146  8.8459214  2.5176840     9.9  8.0698831  4.4343509
   2  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  2.3177156
   3  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915 -0.3280785
   4  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  3.3760332
   5  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  3.9051920
   6  -0.1 -0.3803852 -0.1829487 -0.2397579 -0.1347095 -0.1465319    -0.1 -0.1646915  0.7302392
   7  -0.1 -0.3803852  8.1329033  2.9570146 -0.1347095  9.1782237    -0.1 -0.1646915  1.7885568
```

YOUR TURN PART 3

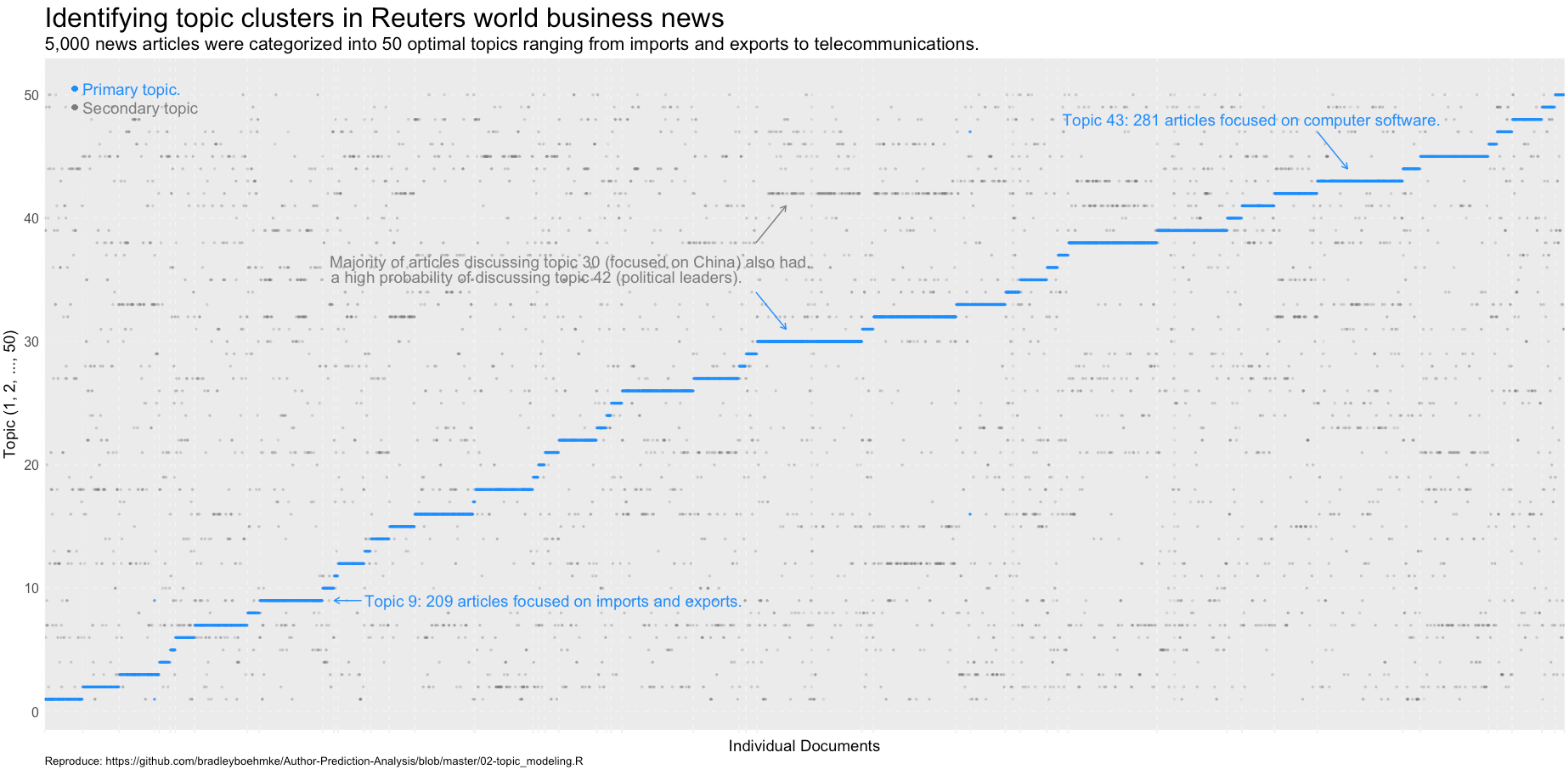
5 minutes

Identify the optimal number of topics. Hint: start with a small search space then expand.

YOUR TURN PART 4

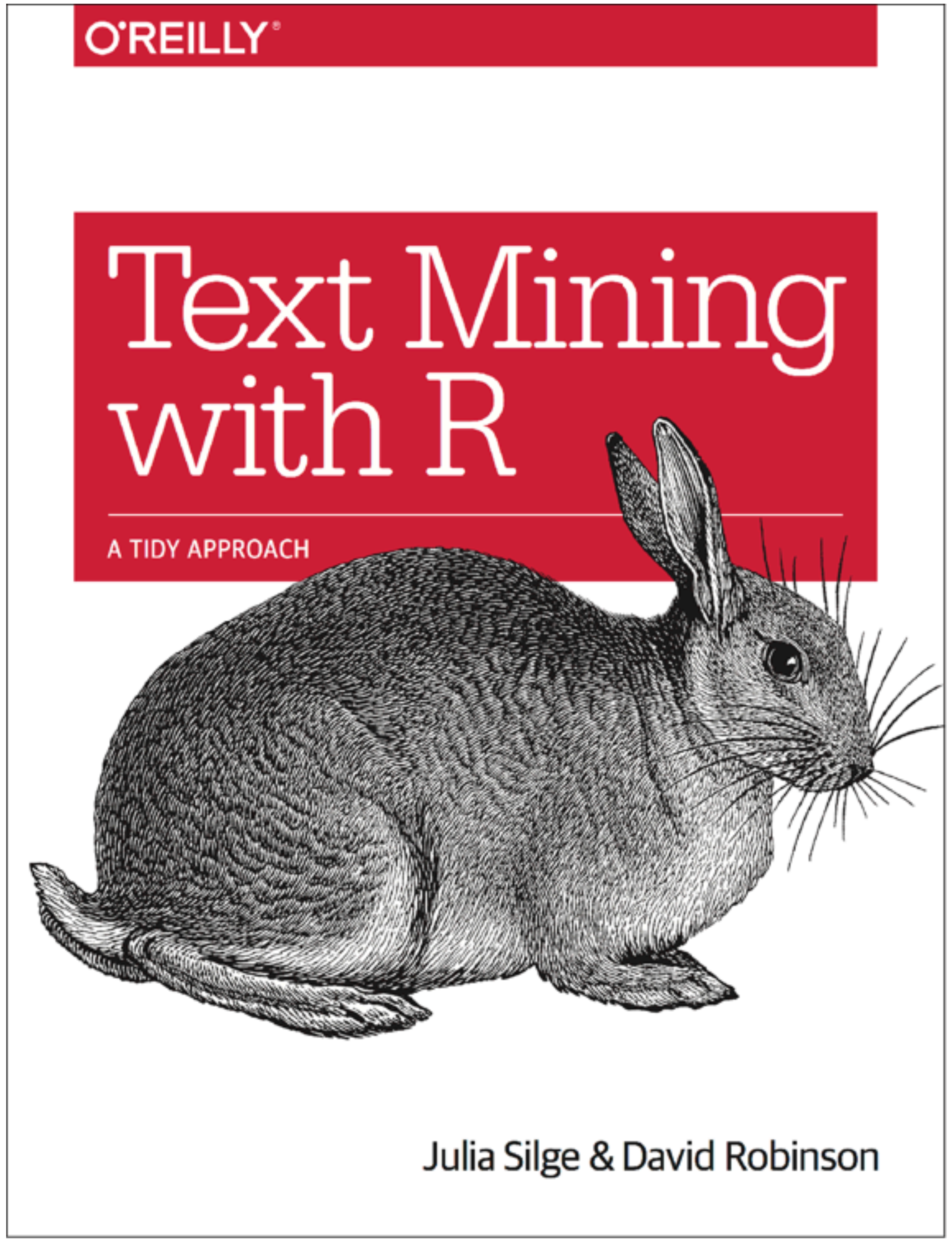
5 minutes

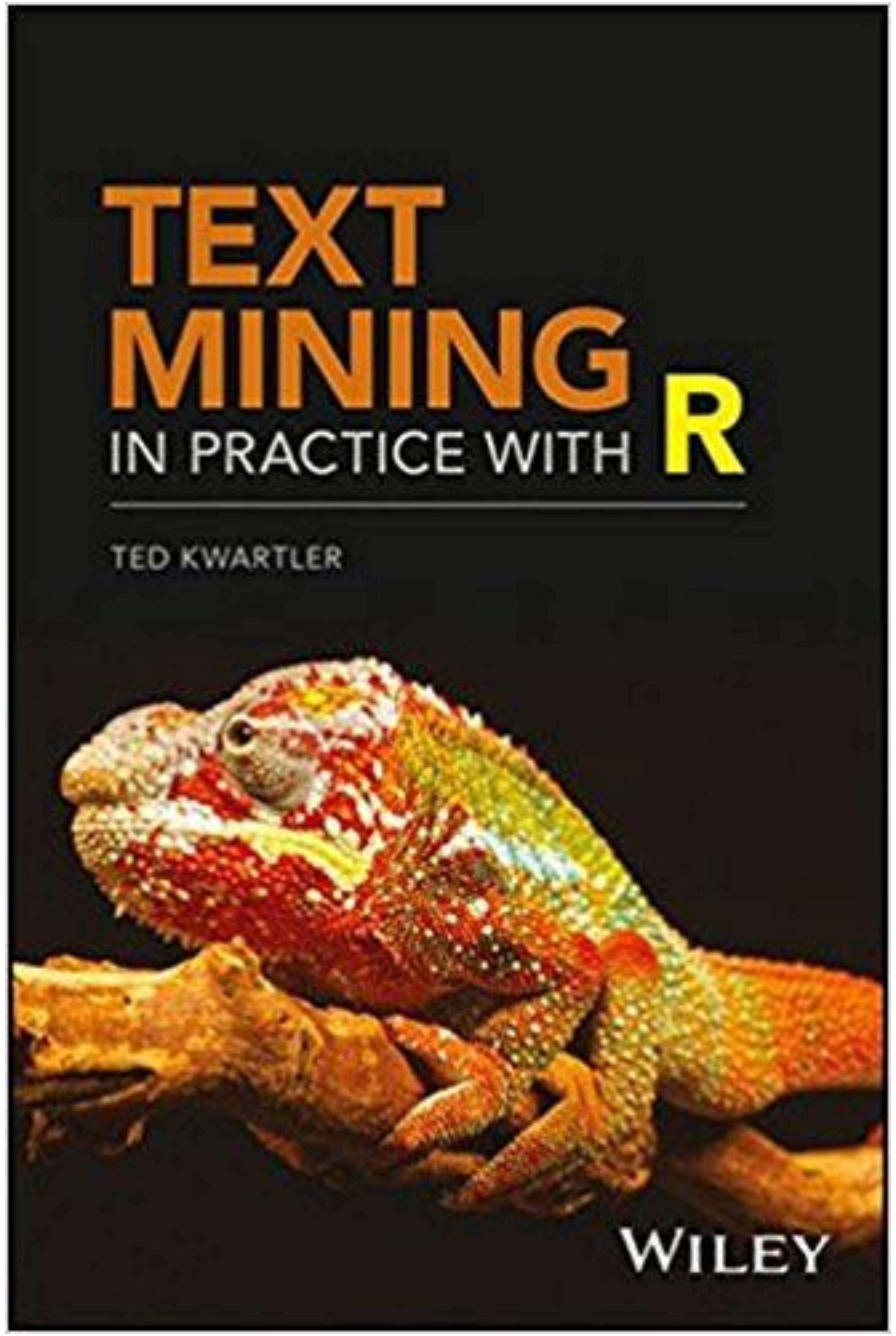
Apply a topic model with the preferred k and identify the words that best explain each topic.


Identifying topic clusters in Reuters world business news
5,000 news articles were categorized into 50 optimal topics ranging from imports and exports to telecommunications.
Primary topic.
Secondary topic
Topic 43: 281 articles focused on computer software.
Majority of articles discussing topic 30 (focused on China) also had a high probability of discussing topic 42 (political leaders).
Topic 9: 209 articles focused on imports and exports.
Topic (1, 2, ..., 50)
0
10
20
30
40
50
Individual Documents
Reproduce: https://github.com/bradleyboehmke/Author-Prediction-Analysis/blob/master/02-topic_modeling.R

SO LITTLE TIME!

LEARN MORE


O'REILLY
Text Mining with R
A TIDY APPROACH
Julia Silge & David Robinson


TEXT MINING IN PRACTICE WITH R
TED KWARTLER
WILEY