Topic modeling demo

Load necessary libraries

```
library(tidyverse)
library(tidymodels)
library(tidylo)
library(stm)
library(purrr)
library(RColorBrewer)
library(ggwordcloud)
library(ggrepel)
library(patchwork)

#devtools::install_github("juliasilge/tidytext", force = TRUE)
library(tidytext)

theme_set(theme_bw())
```

Load data: publicly available text datasets for the Office dialogue across all seasons:

```
library(schrute)
df_office <- schrute::theoffice
speaking_characters <- df_office %>%
  count(character, sort = TRUE) %>%

  # select only characters with at least 500 lines across all seasons
  filter(n > 500) %>%
  pull(character)
```

Unnest tokens: restructure as one-token-per-row (each word is a separate row in a tibble)

```
df_tokens <- df_office %>%

  # filter for our speaking characters
  filter(character %in% speaking_characters) %>%

  group_by(character) %>%
  unnest_tokens(word, text) %>%

  # remove stop words (uninteresting words like of, from, and, the, etc.)
  anti_join(get_stopwords(), by = join_by(word))
```

```r
head(df_tokens)
```

```
## # A tibble: 6 x 2
## # Groups:   character [1]
##   character word
##   <chr>     <chr>
## 1 Michael   right
## 2 Michael   jim
## 3 Michael   quarterlies
## 4 Michael   look
## 5 Michael   good
## 6 Michael   things
```

Create a sparse matrix: each row is a corpus (book) and each column is a word

```r
sparse_mat <- df_tokens %>%

  # count number of times a word is used by each character
  dplyr::count(character, word) %>%

  # filter for words that appear at least 3 times
  filter(n > 3) %>%

  # create sparse matrix
  cast_sparse(character, word, n)

dim(sparse_mat)
```

```
## [1]   19 3016
```

```r
sparse_mat[1:10, 1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
## Andy   4 5 4 6  5  8 6 4 4  50
## Angela . . . .  .  . . . .   7
## Darryl . . . .  .  . . . .   8
## Dwight . . . 5 12  8 . 6 8  24
## Erin   . . . .  .  . . . .   5
## Holly  . . . .  .  . . . .   4
## Jan    . . . .  .  . . . .   7
## Jim    . 6 7 6  . 13 . . . 111
## Kelly  . . 4 .  .  . . . .  18
## Kevin  . . . .  .  . . . .   7
```

Fit topic model using Structural Topic Models (http://www.structuraltopicmodel.com/)

```r
set.seed(0306)
topic_model <- stm(sparse_mat,
                   K = 4, # K is the number of 'topics' (clusters)
                   verbose = FALSE)

summary(topic_model)
```

```
## A topic model with 4 topics, 19 documents and a 3016 word dictionary.

## Topic 1 Top Words:
##       Highest Prob: just, oh, know, yeah, like, michael, get
##       FREX: michael, andy, um, mean, sorry, hi, kevin
##       Lift: channel, foster, cuz, executive, ruff, bloody, blake
##       Score: ethics, senator, cece, mural, pens, pum, confused
## Topic 2 Top Words:
##       Highest Prob: just, know, oh, yeah, right, like, hey
##       FREX: tuna, whoa, na, mike, gotta, pretty, nope
##       Lift: agent, america's, andrew, anger, audition, ba, banjo
##       Score: tuna, nard, jakey, wimowheh, philly, na, andrew
## Topic 3 Top Words:
##       Highest Prob: oh, know, just, okay, like, get, jim
##       FREX: schrute, mose, hay, farm, sensei, beet, male
##       Lift: hay, immediately, knife, wake, weak, 107th, 6
##       Score: spell, hay, mose, sensei, male, belsnickel, battle
## Topic 4 Top Words:
##       Highest Prob: know, just, oh, okay, like, well, right
##       FREX: beep, award, trouble, somebody, everybody, 300, rabies
##       Lift: 200, 4, 500, 645, a.j, ability, accountant
##       Score: packer, anxiety, beep, award, boom, trouble, anybody
```

lift = frequency divided by frequency in other topics **FREX** weights words by frequency and exclusivity to the topic

Ta-da!!

But. . . . how can we determine the optimal number of topics for our dataset?

```r
df_many_models <- tibble(K = seq(3, 10, by = 1)) %>%
  mutate(topic_model = map (

    .x = K,
    .f = ~ stm(sparse_mat,
               K = .x,
               verbose = FALSE)

  ))

head(df_many_models)
```

```
## # A tibble: 6 x 2
##       K topic_model
```

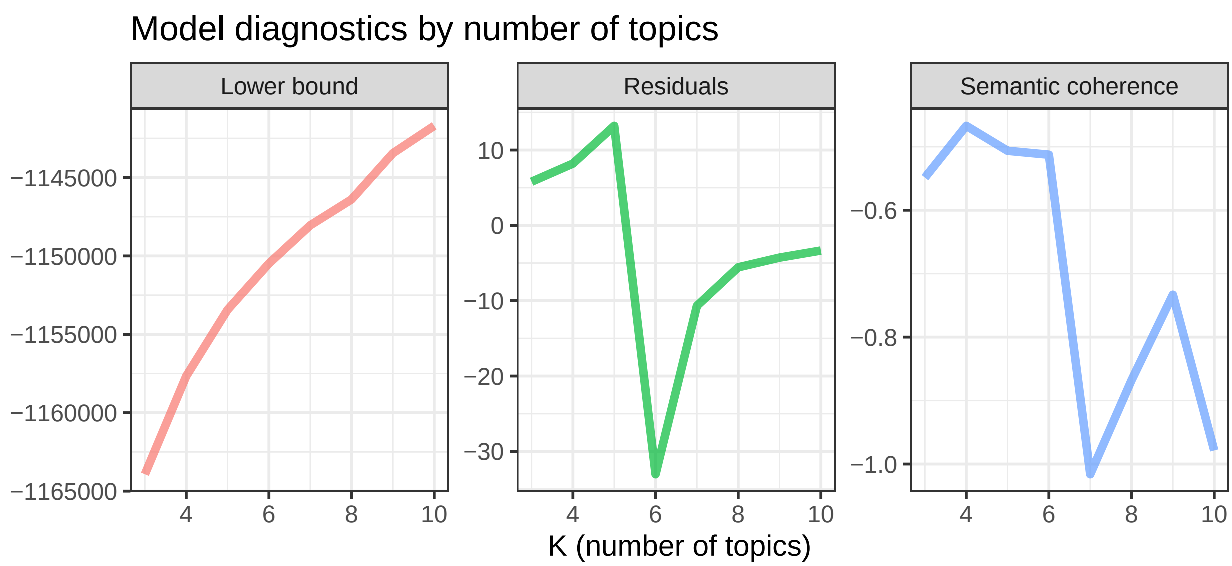
```
##   <dbl> <list>
## 1     3 <STM>
## 2     4 <STM>
## 3     5 <STM>
## 4     6 <STM>
## 5     7 <STM>
## 6     8 <STM>
```

Now that we've fit all these topic models with different numbers of topics, we can explore how many topics are appropriate/good/"best"

```
df_diagnostics <- df_many_models %>%
  mutate(exclusivity = map(topic_model, exclusivity),
         semantic_coherence = map(topic_model, semanticCoherence, sparse_mat),
         residual = map(topic_model, checkResiduals, sparse_mat),
         bound =  map_dbl(topic_model, function(x) max(x$convergence$bound)),
         lfact = map_dbl(topic_model, function(x) lfactorial(x$settings$dim$K)),
         lbound = bound + lfact,
         iterations = map_dbl(topic_model, function(x) length(x$convergence$bound)))

head(df_diagnostics)
```

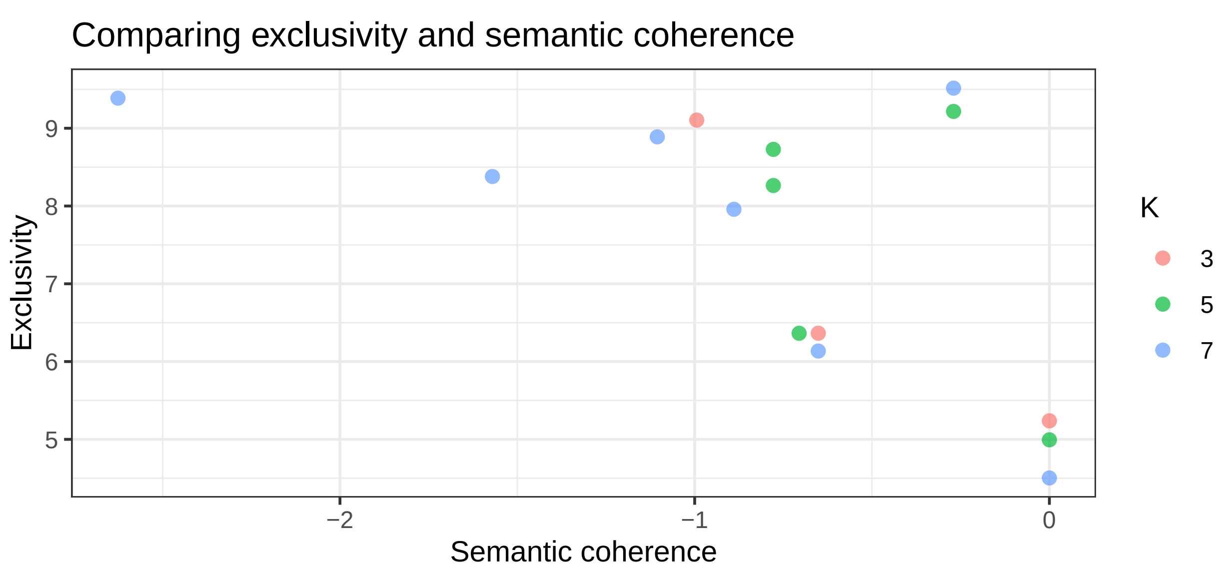
```
## # A tibble: 6 x 9
##       K topic_model exclusivity semantic_coherence residual         bound lfact
##   <dbl> <list>      <list>      <list>             <list>           <dbl> <dbl>
## 1     3 <STM>       <dbl [3]>   <dbl [3]>          <named list> -1163925.  1.79
## 2     4 <STM>       <dbl [4]>   <dbl [4]>          <named list> -1157667.  3.18
## 3     5 <STM>       <dbl [5]>   <dbl [5]>          <named list> -1153438.  4.79
## 4     6 <STM>       <dbl [6]>   <dbl [6]>          <named list> -1150492.  6.58
## 5     7 <STM>       <dbl [7]>   <dbl [7]>          <named list> -1148057.  8.53
## 6     8 <STM>       <dbl [8]>   <dbl [8]>          <named list> -1146407. 10.6
## # i 2 more variables: lbound <dbl>, iterations <dbl>
```

```
df_diagnostics %>%
  transmute(K,
            `Lower bound` = lbound,
            Residuals = map_dbl(residual, "dispersion"),
            `Semantic coherence` = map_dbl(semantic_coherence, mean)) %>%
  gather(Metric, Value, -K) %>%
  ggplot(aes(K, Value, color = Metric)) +
  geom_line(linewidth = 1.5, alpha = 0.7, show.legend = FALSE) +
  facet_wrap(~Metric, scales = "free_y") +
  labs(x = "K (number of topics)",
       y = NULL,
       title = "Model diagnostics by number of topics")
```



Semantic coherence is maximized when the most probable words in a given topic frequently co-occur together, and it's a metric that correlates well with human judgment of topic quality. Having high semantic coherence is relatively easy, though, if you only have a few topics dominated by very common words, so you want to look at both semantic coherence and exclusivity of words to topics. It's a tradeoff. Read more about semantic coherence in the original paper about it (https://dl.acm.org/citation.cfm?id=2145462.

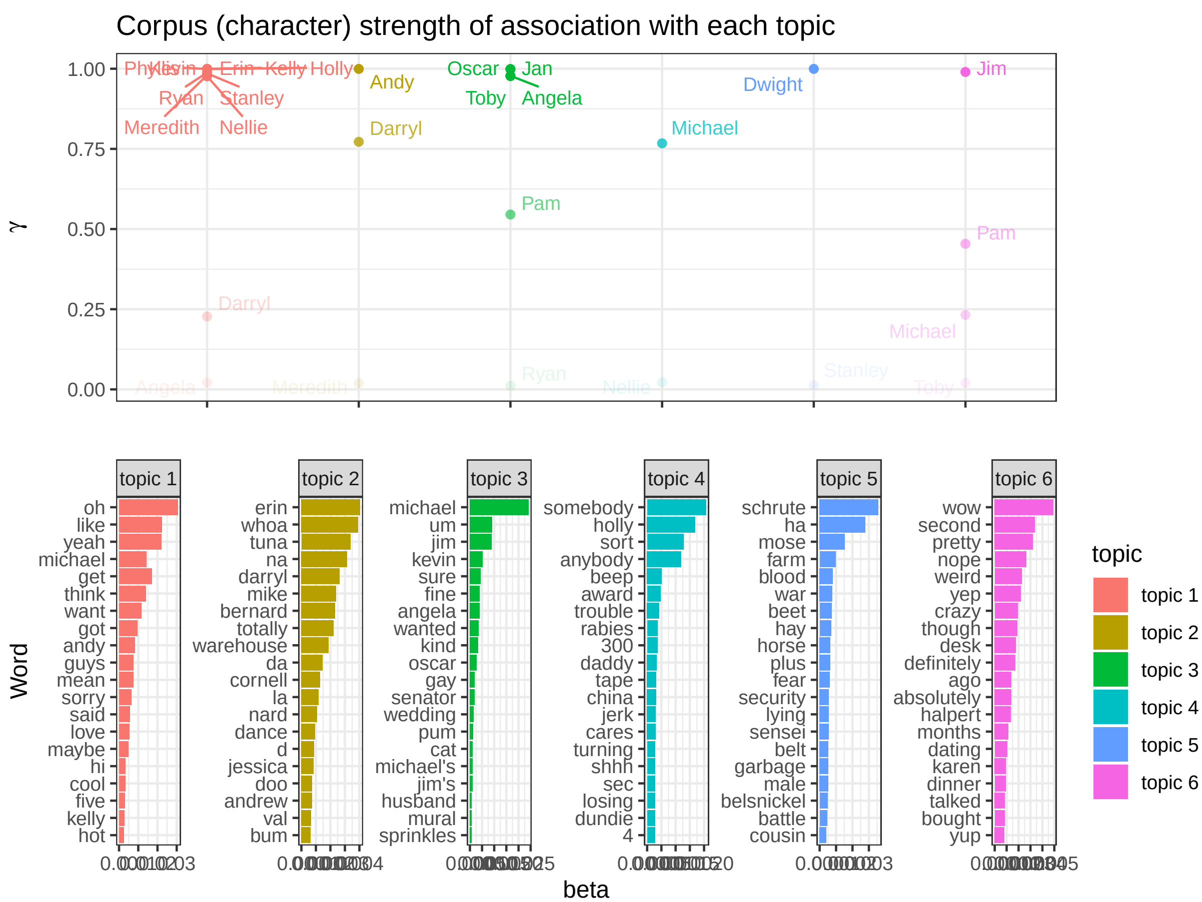
```
df_diagnostics %>%
  select(K, exclusivity, semantic_coherence) %>%
  unnest(cols = c(exclusivity, semantic_coherence)) %>%
  filter(K %in% c(3, 5, 7)) %>%
  mutate(K = as.factor(K)) %>%
  ggplot(aes(semantic_coherence, exclusivity, color = K)) +
  geom_point(size = 2, alpha = 0.7) +
  labs(x = "Semantic coherence",
       y = "Exclusivity",
       title = "Comparing exclusivity and semantic coherence")
```



In choosing our final topic model, we should go with a K value that is statistically sound, but also contextually (in this case, biologically) meaningful and informative

Finalize model

```
topic_model <- stm(sparse_mat,
                   K = 6, # K is the number of 'topics' (clusters)
                   verbose = FALSE)

p_beta <- tidy(topic_model, matrix = "frex") %>%
  left_join(tidy(topic_model)) %>%
  group_by(topic) %>%
  slice_head(n = 20) %>%
  mutate(topic = paste0("topic ", topic)) %>%

  ggplot(aes(x = beta, y = reorder(term, beta), fill = topic)) +
  geom_col() +
  labs(y = "Word") +
  facet_wrap(vars(topic), scales = "free", nrow = 1)

# Topic probabilities
group_gamma <- tidy(
  topic_model,
  matrix = "gamma",
  document_names = rownames(sparse_mat)) %>%
  mutate(chapter = factor(document)) %>%
  dplyr::select(chapter, topic, gamma) %>%
  mutate(topic = factor(topic)
  )

p_gamma <- group_gamma %>%
  filter(gamma > 0.01 ) %>%
  ggplot(aes(x = topic, y = gamma, color = topic)) +
  geom_point(aes(alpha = gamma)) +
  geom_text_repel(aes(alpha = gamma, label = chapter), size = 3,
                  max.overlaps = 50) +

  labs(y = expression(gamma), x = "") +
  ggtitle("Corpus (character) strength of association with each topic") +
  theme(legend.position = "none",
        axis.text.x = element_blank())

(p_gamma / p_beta)
```



Corpus (character) strength of association with each topic
γ
1.00
0.75
0.50
0.25
0.00
Phyllis
Kevin
Erin
Kelly
Holly
Ryan
Stanley
Meredith
Nellie
Andy
Darryl
Oscar
Jan
Toby
Angela
Pam
Michael
Dwight
Jim
Pam
Michael
Word
topic 1
oh
like
yeah
michael
get
think
want
got
andy
guys
mean
sorry
said
love
maybe
hi
cool
five
kelly
hot
topic 2
erin
whoa
tuna
na
darryl
mike
bernard
totally
warehouse
da
cornell
la
nard
dance
d
jessica
doo
andrew
val
bum
topic 3
michael
um
jim
kevin
sure
fine
angela
wanted
kind
oscar
gay
senator
wedding
pum
cat
michael's
jim's
husband
mural
sprinkles
topic 4
somebody
holly
sort
anybody
beep
award
trouble
rabies
300
daddy
tape
china
jerk
cares
turning
shhh
sec
losing
dundie
4
topic 5
schrute
ha
mose
farm
blood
war
beet
hay
horse
plus
fear
security
lying
sensei
belt
garbage
male
belsnickel
battle
cousin
topic 6
wow
second
pretty
nope
weird
yep
crazy
though
desk
definitely
ago
absolutely
halpert
months
dating
karen
dinner
talked
bought
yup
beta
topic
topic 1
topic 2
topic 3
topic 4
topic 5
topic 6