HW5_key

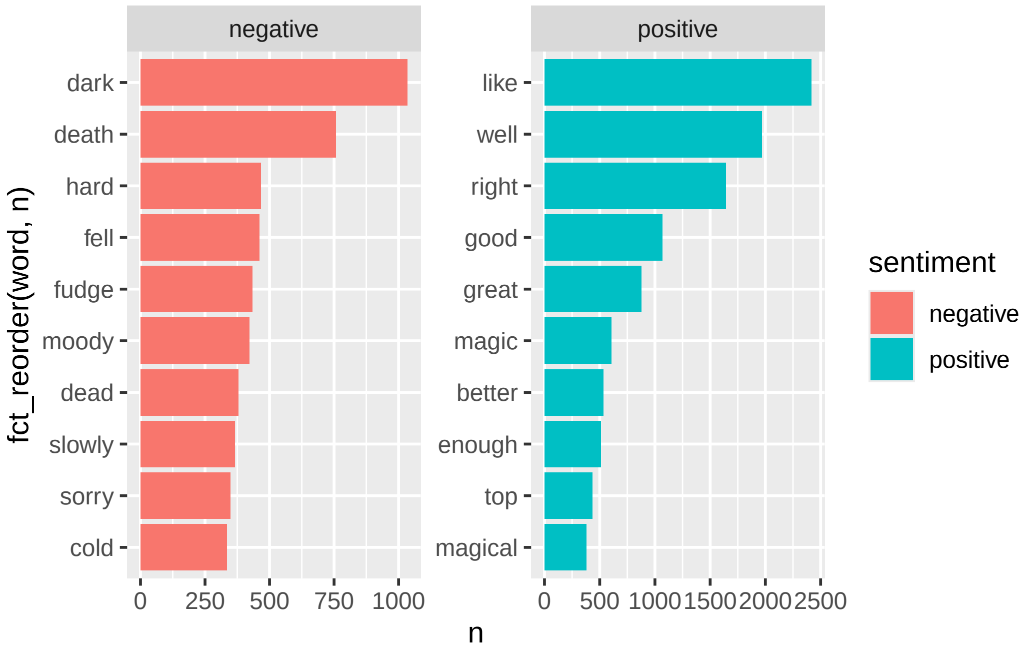
Harry Potter

The `potter_untidy` dataset includes the text of 7 books of the Harry Potter series by J.K. Rowling. For a brief overview of the books (or movies), see this quote from Wikipedia:

> Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).

1. What words contribute the most to negative and positive sentiment scores? Show a faceted bar plot of the top 10 negative and the top 10 positive words (according to the "bing" lexicon) across the entire series.

```
potter_tidy |>
  inner_join(get_sentiments("bing")) |>
  count(sentiment, word, sort = TRUE) |>
  group_by(sentiment) |>
  top_n(10) |>
  ungroup() |>
  ggplot(aes(x = fct_reorder(word, n), y = n, fill = sentiment)) +
    geom_col() +   # makes bar plot where heights = values in the data set
    coord_flip() +
    facet_wrap(~ sentiment, scales = "free")
```




```
# note in warning that enviously appears as both positive and negative
```

2. Find a list of the top 10 words associated with "fear" and with "trust" (according to the "nrc" lexicon) across the entire series.

```
# Check out which words are associated with which sentiment
get_sentiments("nrc") |>
  count(sentiment)
```

```
# A tibble: 10 x 2
   sentiment        n
   <chr>        <int>
 1 anger         1245
 2 anticipation   837
 3 disgust       1056
 4 fear          1474
 5 joy            687
 6 negative      3316
 7 positive      2308
 8 sadness       1187
 9 surprise       532
10 trust         1230
```

```r
get_sentiments("nrc") |>
  filter(sentiment == "fear") |>
  inner_join(potter_tidy) |>
  count(word, sort = TRUE)
```

```
Joining with `by = join_by(word)`
```

```
# A tibble: 887 x 2
   word         n
   <chr>    <int>
 1 death      757
 2 feeling    391
 3 fire       388
 4 crouch     297
 5 shaking    277
 6 scar       276
 7 mad        269
 8 kill       267
 9 elf        259
10 watch      256
# i 877 more rows
```

```r
get_sentiments("nrc") |>
  filter(sentiment == "trust") |>
  inner_join(potter_tidy) |>
  count(word, sort = TRUE)
```

```
Joining with `by = join_by(word)`
```

```
# A tibble: 676 x 2
   word          n
   <chr>     <int>
 1 professor  2006
 2 good       1065
 3 school      634
 4 found       614
 5 ministry    576
 6 top         434
 7 sir         419
 8 feeling     391
```

```
 9 lord         391
10 ground       386
# i 666 more rows
```

3. Make a wordcloud for the entire series after removing stop words using the “smart” source.

```
# wordcloud wants a column with words and another column with counts
words <- potter_tidy |>
  anti_join(stop_words) |>
  anti_join(potter_names, join_by(word == firstname)) |>
  anti_join(potter_names, join_by(word == lastname)) |>
  count(word) |>
  arrange(desc(n))

# Note: this will look better in html than in the Plots window in RStudio
wordcloud(
  words = words$word,
  freq = words$n,
  max.words = 100,
  random.order = FALSE,
  rot.per = 0,
  colors = brewer.pal(6, "Dark2")
)
```

```
# See Z's R Tip of the Day for suggestions on options

# Or for even cooler looks, use wordcloud2 in html
#words_df <- words |>
#  slice_head(n = 80) |>
#  data.frame()

#wordcloud2(words_df, size = .35, shape = 'star')
```

4. Create a wordcloud with the top 20 negative words and the top 20 positive words in the Harry Potter series according to the bing lexicon. The words should be sized by their respective counts and colored based on whether their sentiment is positive or negative. (Feel free to be resourceful and creative to color words by a third variable!)

```
pos_neg <- potter_tidy |>
  inner_join(get_sentiments("bing")) |>
  count(sentiment, word, sort = TRUE) |>
  group_by(sentiment) |>
  top_n(20) |>
  ungroup()
```

```
Joining with `by = join_by(word)`
```

```
Warning in inner_join(potter_tidy, get_sentiments("bing")): Detected an unexpected many-to-ma
i Row 41432 of `x` matches multiple rows in `y`.
i Row 2698 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
Selecting by n
```

```
wordcloud(
  words = pos_neg$word,
  freq = pos_neg$n,
  random.order = FALSE,
  rot.per = 0,
  ordered.colors = TRUE,
  colors = brewer.pal(6, "Dark2")[factor(pos_neg$sentiment)]
)
```

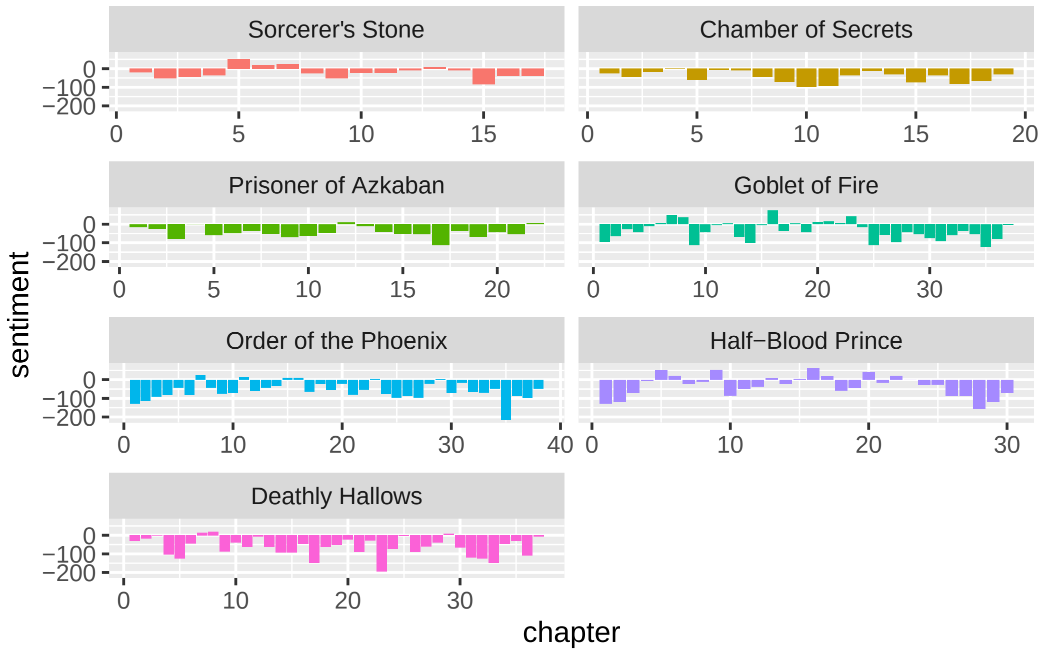

```
# Not sure why this doesn't work...
# pos_neg_df <- data.frame(pos_neg)
# wordcloud2(
#   pos_neg_df[,2:3],
#   color = ifelse(pos_neg_df[,1] == "positive", "blue", "red")
# )

# Can also look in wordcloud documentation under "comparisons and
#   commonality clouds"
```

5. Make a faceted bar chart to compare the positive/negative sentiment trajectory over the 7 Harry Potter books. You should have one bar per chapter (thus chapter becomes the index), and the bar should extend up from 0 if there are more positive than negative words in a chapter (according to the “bing” lexicon), and it will extend down from 0 if there are more negative than positive words.

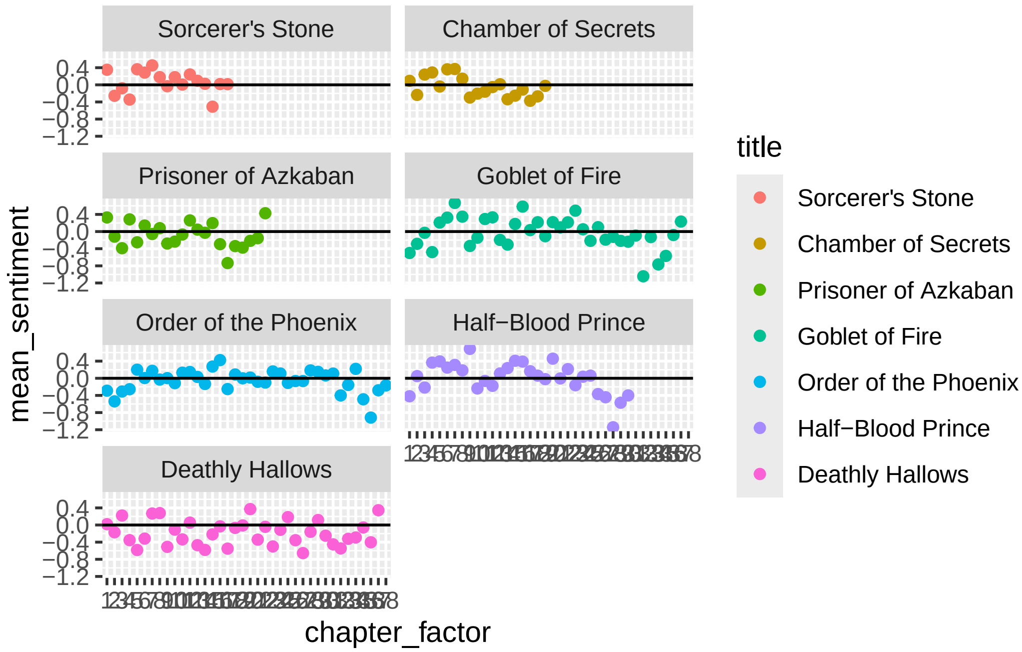
6. Repeat (5) using a faceted scatterplot to show the average sentiment score according to the “afinn” lexicon for each chapter. (Hint: use `mutate(chapter_factor = factor(chapter))` to treat chapter as a factor variable.)

```
potter_tidy |>
  inner_join(get_sentiments("bing")) |>
  count(title, chapter, sentiment) |>
```

```
  spread(key = sentiment, value = n, fill = 0) |>
  mutate(sentiment = positive - negative) |>
  ggplot(aes(x = chapter, y = sentiment, fill = title)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~title, ncol = 2, scales = "free_x")
```




```
potter_tidy |>
  mutate(chapter_factor = factor(chapter)) |>
  inner_join(get_sentiments("afinn")) |>
  group_by(title, chapter_factor) |>
  summarise(mean_sentiment = mean(value)) |>
  ggplot(aes(x = chapter_factor, y = mean_sentiment,
             fill = title, color = title)) +
    geom_point() +
    geom_hline(yintercept = 0) +
    facet_wrap(~title, ncol = 2)
```



7. Make a faceted bar plot showing the top 10 words that distinguish each book according to the tf-idf statistic.

```
book_word_count <- potter_tidy |>
  count(word, title, sort = TRUE)

book_tfidf <- book_word_count |>
  bind_tf_idf(word, title, n)

book_tfidf |>
  arrange(-tf_idf)
```

```
# A tibble: 67,845 x 6
   word        title                    n       tf   idf    tf_idf
   <chr>       <fct>                <int>    <dbl> <dbl>     <dbl>
 1 slughorn    Half-Blood Prince      335 0.00196  1.25  0.00245
 2 umbridge    Order of the Phoenix   496 0.00192  0.847 0.00162
 3 bagman      Goblet of Fire         208 0.00108  1.25  0.00136
 4 lockhart    Chamber of Secrets     197 0.00231  0.560 0.00129
 5 lupin       Prisoner of Azkaban    369 0.00351  0.336 0.00118
 6 winky       Goblet of Fire         145 0.000756 1.25  0.000947
 7 champions   Goblet of Fire          84 0.000438 1.95  0.000852
```

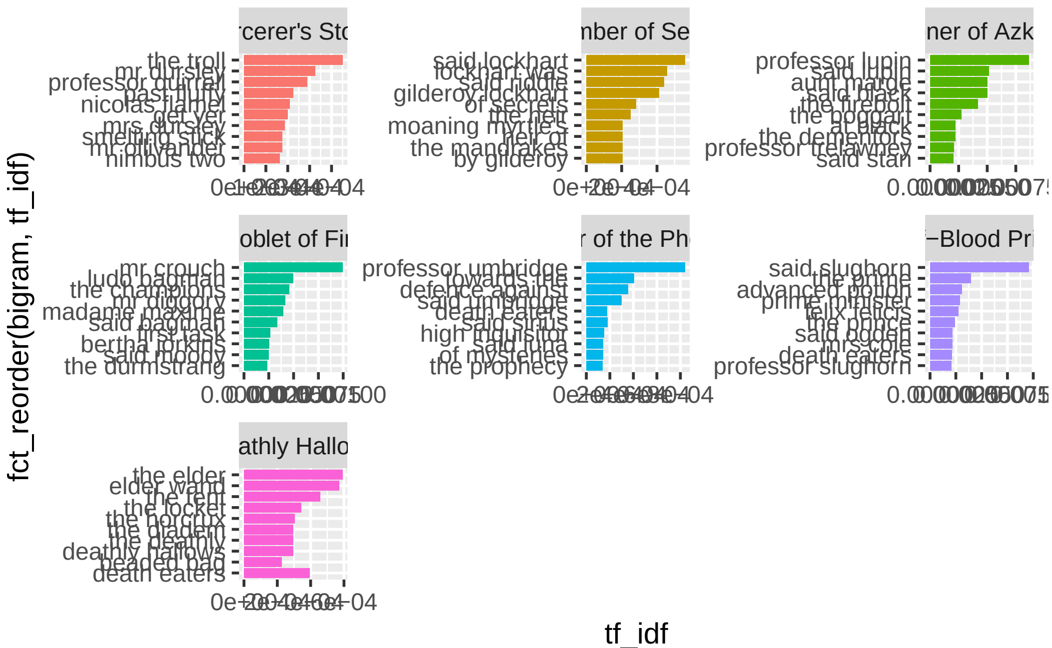
```
 8 xenophilius Deathly Hallows          79 0.000400 1.95  0.000778
 9 griphook    Deathly Hallows         117 0.000592 1.25  0.000742
10 mclaggen    Half-Blood Prince        65 0.000379 1.95  0.000738
# i 67,835 more rows
```

```r
book_tfidf |>
  group_by(title) |>
  arrange(desc(tf_idf)) |>
  top_n(10, wt = tf_idf) |>
  ungroup() |>
  ggplot(aes(x = fct_reorder(word, tf_idf), y = tf_idf, fill = title)) +
    geom_col(show.legend = FALSE) +
    coord_flip() +
    facet_wrap(~title, scales = "free")
```

8. Repeat (7) to show the top 10 2-word combinations that distinguish each book.

```r
tidy_ngram <- potter_untidy |>
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

bigram_tf_idf <- tidy_ngram |>
  count(title, bigram) |>
```

```r
  bind_tf_idf(bigram, title, n) |>
  arrange(desc(tf_idf)) |>
  filter(!is.na(bigram))

bigram_tf_idf |>
  group_by(title) |>
  arrange(desc(tf_idf)) |>
  top_n(10, wt = tf_idf) |>
  ungroup() |>
  ggplot(aes(x = fct_reorder(bigram, tf_idf), y = tf_idf, fill = title)) +
    geom_col(show.legend = FALSE) +
    coord_flip() +
    facet_wrap(~title, scales = "free")
```



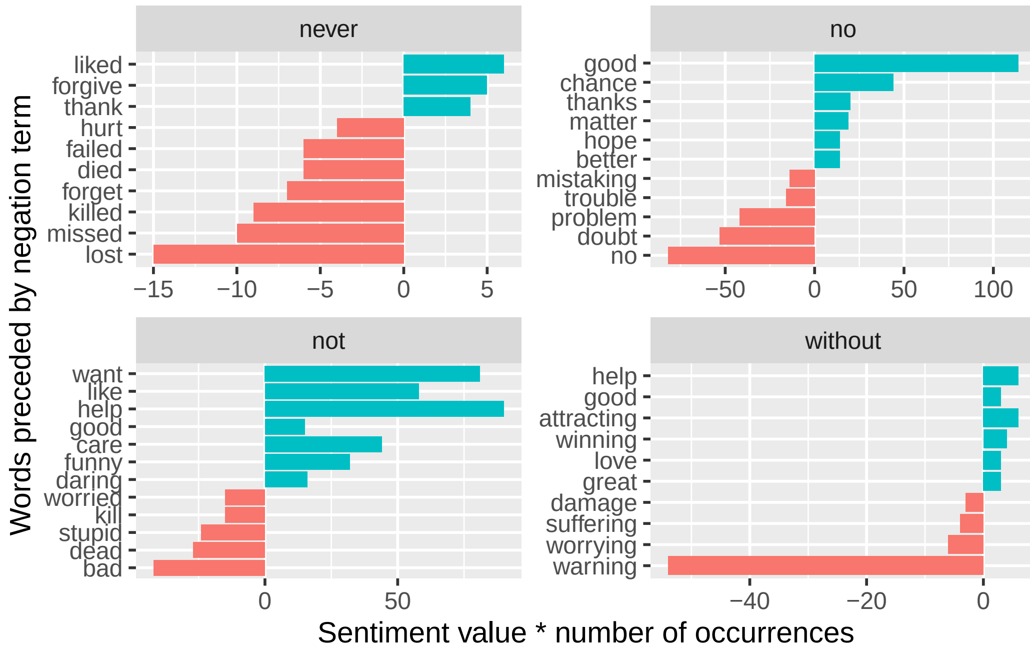
9. Find which words contributed most in the "wrong" direction using the afinn sentiment combined with how often a word appears among all 7 books. Come up with a list of 4 negation words, and for each negation word, illustrate the words associated with the largest "wrong" contributions in a faceted bar plot.

```r
afinn <- get_sentiments("afinn")

bigrams_separated <- tidy_ngram |>
```

```r
  separate(bigram, c("word1", "word2"), sep = " ") |>
  count(word1, word2, sort = TRUE) |>
  filter(!is.na(word1) & !is.na(word2))

negation_words <- c("not", "no", "never", "without")

negated_words <- bigrams_separated |>
  filter(word1 %in% negation_words) |>
  inner_join(afinn, by = c(word2 = "word")) |>
  arrange(desc(n))

negated_words
```

```
# A tibble: 379 x 4
   word1 word2       n value
   <chr> <chr>   <int> <dbl>
 1 no    no         82    -1
 2 not   want       81     1
 3 no    doubt      53    -1
 4 not   help       45     2
 5 no    good       38     3
 6 not   like       29     2
 7 no    chance     22     2
 8 not   care       22     2
 9 no    problem    21    -2
10 no    matter     19     1
# i 369 more rows
```

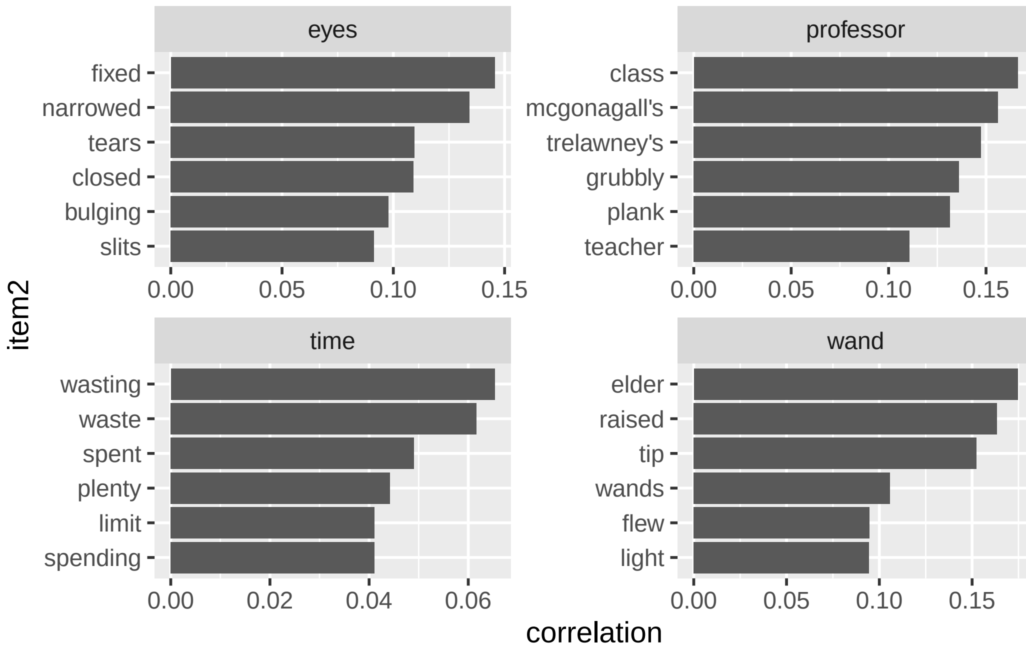
```r
negated_words |>
  mutate(contribution = n * value) |>
  arrange(desc(abs(contribution))) |>
  group_by(word1) |>
  slice_max(abs(contribution), n = 10) |>
  ungroup() |>
  mutate(word2 = reorder(word2, contribution)) |>
  ggplot(aes(n * value, word2, fill = n * value > 0)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ word1, scales = "free") +
    labs(x = "Sentiment value * number of occurrences",
         y = "Words preceded by negation term")
```



10. Select a set of 4 “interesting” terms and then use the Phi coefficient to find and plot the 6 words most correlated with each of your “interesting” words. Start by dividing `potter_tidy` into 80-word sections and then remove names and spells and stop words.

```
library(widyr)

potter_section_words <- potter_tidy |>
  mutate(section = 1 + row_number() %/% 80) |>
  anti_join(potter_names, join_by(word == firstname)) |>
  anti_join(potter_names, join_by(word == lastname)) |>
  anti_join(potter_spells, join_by(word == first_word)) |>
  anti_join(potter_spells, join_by(word == second_word)) |>
  filter(!word %in% stop_words$word,
         !is.na(word))

word_cors <- potter_section_words |>
  group_by(word) |>
  filter(n() >= 10) |>
  pairwise_cor(word, section, sort = TRUE)

# Plot words most associated with a set of interesting words:
word_cors |>
  filter(item1 %in% c("eyes", "professor", "wand", "time")) |>
```

```
  group_by(item1) |>
  slice_max(correlation, n = 6) |>
  ungroup() |>
  mutate(item2 = reorder(item2, correlation)) |>
  ggplot(aes(item2, correlation)) +
    geom_bar(stat = "identity") +
    facet_wrap(~ item1, scales = "free") +
    coord_flip()
```



11. Create a network graph to visualize the correlations and clusters of words that were found by the `widyr` package in (10).

```
library(igraph)
```

```
Attaching package: 'igraph'

The following objects are masked from 'package:lubridate':

    %--%, union
```

```
The following objects are masked from 'package:dplyr':

    as_data_frame, groups, union


The following objects are masked from 'package:purrr':

    compose, simplify


The following object is masked from 'package:tidyr':

    crossing


The following object is masked from 'package:tibble':

    as_data_frame


The following objects are masked from 'package:stats':

    decompose, spectrum


The following object is masked from 'package:base':

    union
```

```r
library(ggraph)
set.seed(1989)

word_cors |>
  filter(correlation > .5) |>
  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()
```

12. Use LDA to fit a 2-topic model to all 7 Harry Potter books. Be sure to remove names, spells, and stop words before running your topic models. (a) Make a plot to illustrate words with greatest difference between two topics, using log ratio. (b) Print a table with the gamma variable for each document and topic. Based on (a) and (b), can you interpret what the two topics represent?

```
# cast the collection of 3 works as a document-term matrix
library(tm)
```

```
Loading required package: NLP
```

```
Attaching package: 'NLP'
```

```
The following object is masked from 'package:ggplot2':

    annotate
```

```
book_word_count <- potter_tidy |>
  group_by(title, word) |>
  count() |>
```

```r
  arrange(desc(n))

seven_books_dtm <- book_word_count |>
  anti_join(potter_names, join_by(word == firstname)) |>
  anti_join(potter_names, join_by(word == lastname)) |>
  anti_join(potter_spells, join_by(word == first_word)) |>
  anti_join(potter_spells, join_by(word == second_word)) |>
  filter(!word %in% stop_words$word,
         !is.na(word)) |>
  cast_dtm(title, word, n)

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

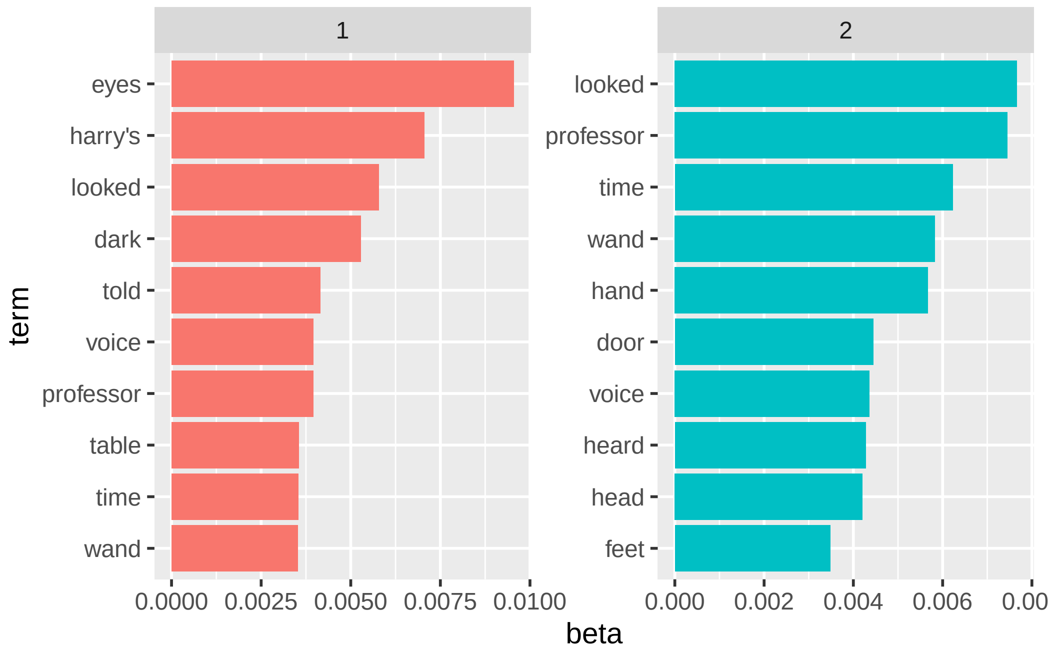
```
A LDA_VEM topic model with 2 topics.
```

```r
seven_books_topics <- tidy(seven_books_lda, matrix = "beta")
seven_books_topics
```

```
# A tibble: 46,830 x 3
   topic term         beta
   <int> <chr>       <dbl>
 1     1 professor 0.00395
 2     2 professor 0.00746
 3     1 wand      0.00352
 4     2 wand      0.00583
 5     1 looked    0.00578
 6     2 looked    0.00766
 7     1 voice     0.00395
 8     2 voice     0.00436
 9     1 time      0.00353
10     2 time      0.00623
# i 46,820 more rows
```

```r
# Find the most common words within each topic
seven_books_top_terms <- seven_books_topics |>
  group_by(topic) |>
  slice_max(beta, n = 10) |>
```

```
  ungroup() |>
  arrange(topic, -beta)

seven_books_top_terms |>
  mutate(term = reorder_within(term, beta, topic)) |>
  ggplot(aes(beta, term, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    scale_y_reordered()
```




```
# Find words with greatest difference between two topics, using log ratio
beta_wide <- seven_books_topics |>
  mutate(topic = paste0("topic", topic)) |>
  pivot_wider(names_from = topic, values_from = beta) |>
  filter(topic1 > .001 | topic2 > .001) |>
  mutate(log_ratio = log2(topic2 / topic1))

beta_wide
```

```
# A tibble: 201 x 4
   term        topic1    topic2 log_ratio
   <chr>        <dbl>     <dbl>     <dbl>
```

```
 1 professor 0.00395 0.00746      0.916
 2 wand      0.00352 0.00583      0.729
 3 looked    0.00578 0.00766      0.407
 4 voice     0.00395 0.00436      0.142
 5 time      0.00353 0.00623      0.817
 6 door      0.00300 0.00445      0.568
 7 head      0.00342 0.00420      0.296
 8 harry's   0.00705 0.00120     -2.56
 9 eyes      0.00954 0.000179    -5.74
10 death     0.00251 0.00189     -0.405
# i 191 more rows
```

```
beta_wide |>
  arrange(desc(abs(log_ratio))) |>
  slice_max(abs(log_ratio), n = 20) |>
  mutate(term = reorder(term, log_ratio)) |>
  ggplot(aes(log_ratio, term, fill = log_ratio > 0)) +
    geom_col(show.legend = FALSE) +
    labs(x = "Log ratio of Beta values",
         y = "Words in seven works")
```

```r
# find the gamma variable for each document and topic
seven_books_documents <- tidy(seven_books_lda, matrix = "gamma")
seven_books_documents
```

```
# A tibble: 14 x 3
   document             topic gamma
   <chr>                <int> <dbl>
 1 Sorcerer's Stone         1 0.489
 2 Chamber of Secrets       1 0.524
 3 Prisoner of Azkaban      1 0.436
 4 Goblet of Fire           1 0.501
 5 Order of the Phoenix     1 0.466
 6 Half-Blood Prince        1 0.457
 7 Deathly Hallows          1 0.482
 8 Sorcerer's Stone         2 0.511
 9 Chamber of Secrets       2 0.476
10 Prisoner of Azkaban      2 0.564
11 Goblet of Fire           2 0.499
12 Order of the Phoenix     2 0.534
13 Half-Blood Prince        2 0.543
14 Deathly Hallows          2 0.518
```

Note: this code could be even better by converting repeated sections into functions!