Some Simple SPOOKY Data Analysis

Cindy Rush

January 22, 2018

Introduction

This files contains text mining analysis of the SPOOKY data. The goal is to remind ourselves of some of our basic tools for working with text data in `R` and also to practice reproducibility. You should be able to put this file in the `doc` folder of your `Project 1` repository and it should just run (provided you have `multiplot.R` in the `libs` folder and `spooky.csv` in the `data` folder).

Setup the libraries

First we want to install and load libraries we need along the way. Note that the following code is completely reproducible – you don't need to add any code on your own to make it run.

```
packages.used <- c("ggplot2", "dplyr", "tibble", "tidyr",  "stringr", "tidytext", "topicmodels", "wordcl

# check packages that need to be installed.
packages.needed <- setdiff(packages.used, intersect(installed.packages()[,1], packages.used))

# install additional packages
if(length(packages.needed) > 0) {
  install.packages(packages.needed, dependencies = TRUE, repos = 'http://cran.us.r-project.org')
}
```

```
##
## The downloaded binary packages are in
##  /var/folders/cq/ttfp57xn5bsb6v_g87q46nzr0000gn/T//RtmpqKBcXv/downloaded_packages
```

```
library(ggplot2)
library(dplyr)
library(tibble)
library(tidyr)
library(stringr)
library(tidytext)
library(topicmodels)
library(wordcloud)
library(ggridges)

source("../libs/multiplot.R")
```

Read in the data

The following code assumes that the dataset `spooky.csv` lives in a `data` folder (and that we are inside a `docs` folder).

```
spooky <- read.csv('../data/spooky.csv', as.is = TRUE)
```

An overview of the data structure and content

Let's first remind ourselves of the structure of the data.

```
head(spooky)
```

```
##        id
## 1 id26305
## 2 id17569
## 3 id11008
## 4 id27763
## 5 id12958
## 6 id22965
##
## 1
## 2
## 3
## 4
## 5
## 6 A youth passed in solitude, my best years spent under your gentle and feminine fosterage, has so r
##   author
## 1    EAP
## 2    HPL
## 3    EAP
## 4    MWS
## 5    HPL
## 6    MWS
```

```
summary(spooky)
```

```
##       id                 text              author
##  Length:19579       Length:19579       Length:19579
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

We see from the above that each row of our data contains a unique ID, a single sentence text excerpt, and an abbreviated author name. `HPL` is Lovecraft, `MWS` is Shelly, and `EAP` is Poe. We finally note that there are no missing values, and we change author name to be a factor variable, which will help us later on.

```
sum(is.na(spooky))
```

```
## [1] 0
```

```
spooky$author <- as.factor(spooky$author)
```

Data Cleaning

We first use the `unnest_tokens()` function to drop all punctuation and transform all words into lower case. At least for now, the punctuation isn't really important to our analysis – we want to study the words. In addition, `tidytext` contains a dictionary of stop words, like "and" or "next", that we will get rid of for our analysis, the idea being that the non-common words (…maybe the SPOOKY words) that the authors use will be more interesting. If this is new to you, here's a textbook that can help: *Text Mining with R; A Tidy Approach*. It teaches the basic handling of natural language data in `R` using tools from the "tidyverse". The tidy text format is a table with one token per row, where a token is a word.

```
spooky_wrd <- unnest_tokens(spooky, word, text)
spooky_wrd <- anti_join(spooky_wrd, stop_words, by = "word")
```

Last Time and some More

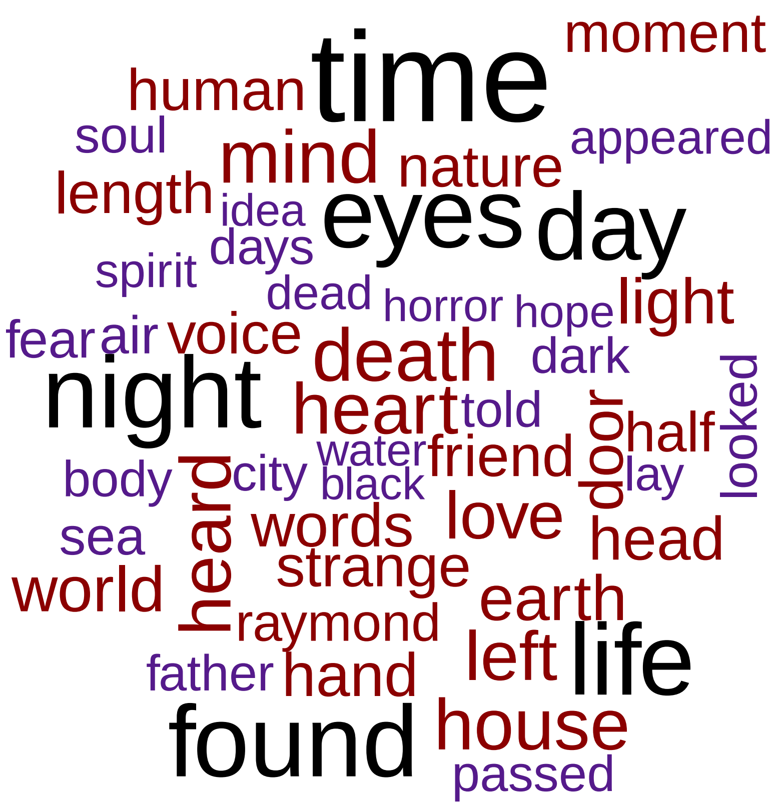
Word Frequency

Now we study some of the most common words in the entire data set. With the below code we plot the fifty most common words in the entire datset. We see that "time", "life", and "night" all appear frequently.

```
words <- count(group_by(spooky_wrd, word))$word
freqs <- count(group_by(spooky_wrd, word))$n
head(sort(freqs, decreasing = TRUE))
```

```
## [1] 729 563 559 559 540 516
```

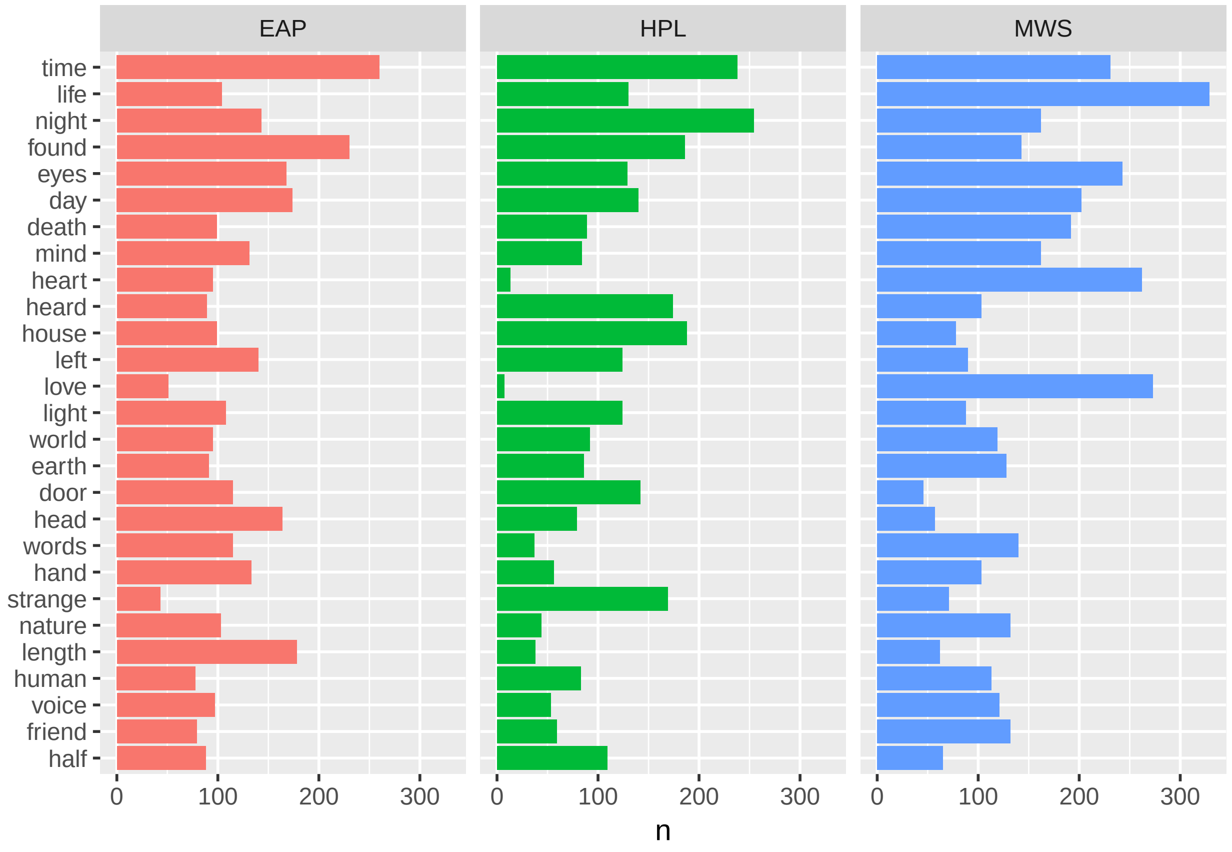
```
wordcloud(words, freqs, max.words = 50, color = c("purple4", "red4", "black"))
```



We can compare the way the authors use the most frequent words too.

```
author_words <- count(group_by(spooky_wrd, word, author))
all_words    <- rename(count(group_by(spooky_wrd, word)), all = n)

author_words <- left_join(author_words, all_words, by = "word")
author_words <- arrange(author_words, desc(all))
author_words <- ungroup(head(author_words, 81))

ggplot(author_words) +
```

```r
  geom_col(aes(reorder(word, all, FUN = min), n, fill = author)) +
  xlab(NULL) +
  coord_flip() +
  facet_wrap(~ author) +
  theme(legend.position = "none")
```



Data Visualization

We'll do some simple numerical summaries of the data to provide some nice visualizations.

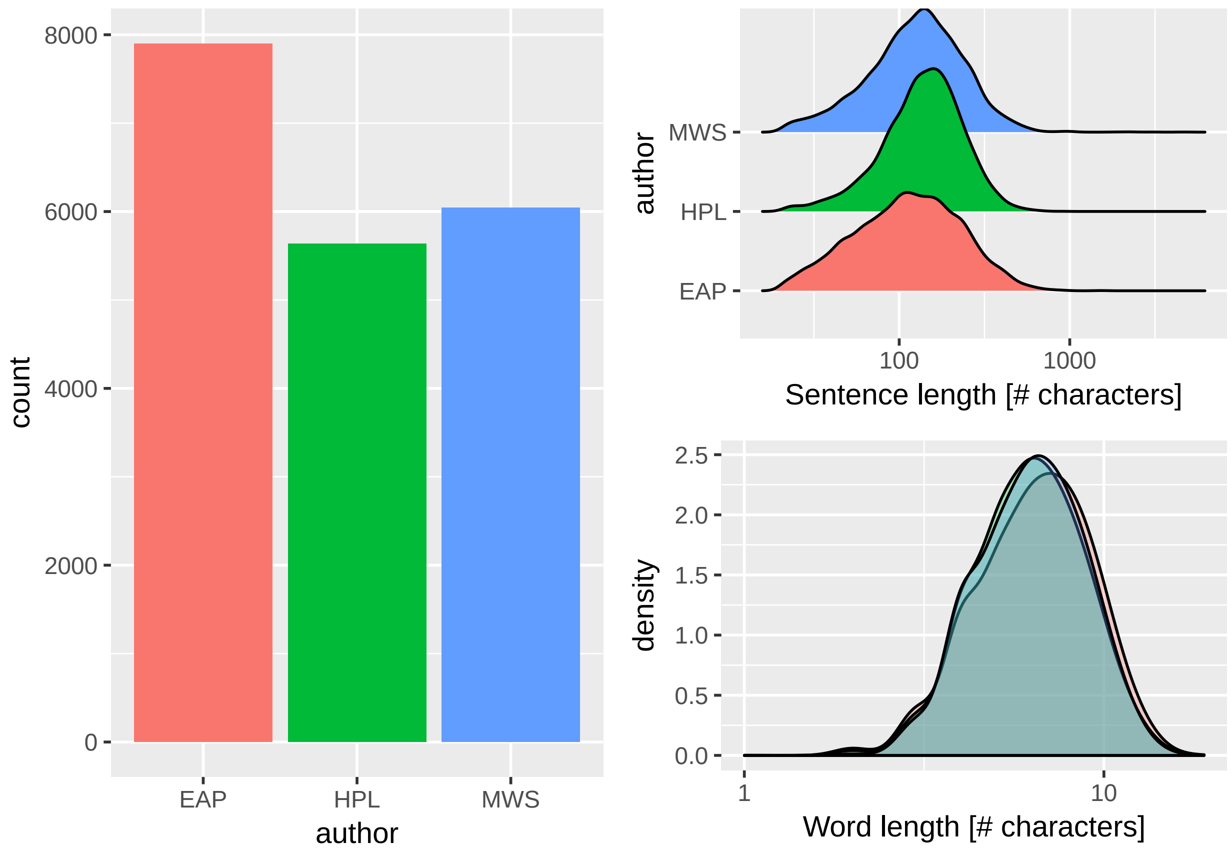
```r
p1 <- ggplot(spooky) +
      geom_bar(aes(author, fill = author)) +
      theme(legend.position = "none")

spooky$sen_length <- str_length(spooky$text)
head(spooky$sen_length)
```

```
## [1] 231  71 200 206 174 468
```

```r
p2 <- ggplot(spooky) +
      geom_density_ridges(aes(sen_length, author, fill = author)) +
      scale_x_log10() +
      theme(legend.position = "none") +
      labs(x = "Sentence length [# characters]")

spooky_wrd$word_length <- str_length(spooky_wrd$word)
head(spooky_wrd$word_length)
```

```
## [1]  7  8  5 12 10  7
```

```
p3 <- ggplot(spooky_wrd) +
      geom_density(aes(word_length, fill = author), bw = 0.05, alpha = 0.3) +
      scale_x_log10() +
      theme(legend.position = "none") +
      labs(x = "Word length [# characters]")

layout <- matrix(c(1, 2, 1, 3), 2, 2, byrow = TRUE)
multiplot(p1, p2, p3, layout = layout)
```



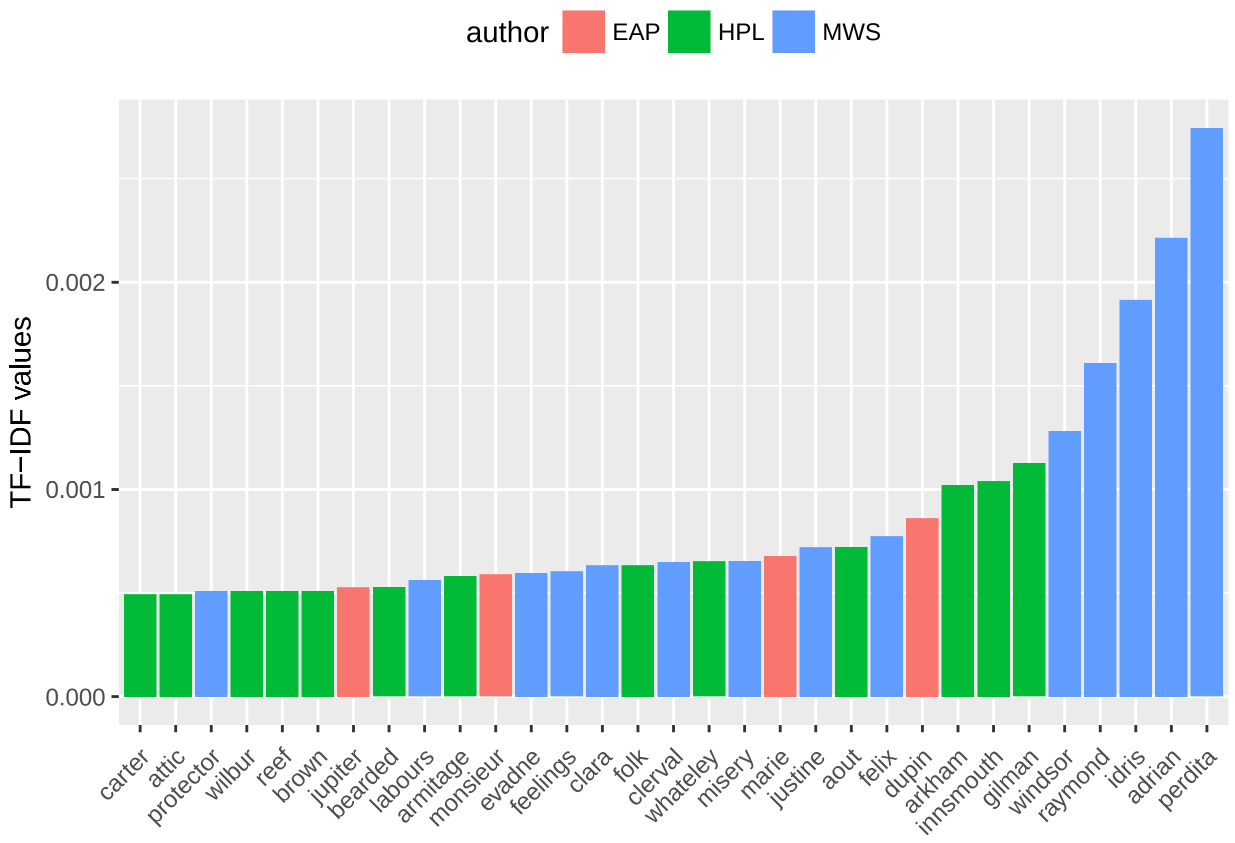
From the above plots we find:

-
-
-

TF-IDF

TF stands for term frequency or how often a word appears in a text and it is what is studied above in the word cloud. IDF stands for inverse document frequncy, and it is a way to pay more attention to words that are rare within the entire set of text data that is more sophisticated than simply removing stop words. Multiplying these two values together calculates a term's tf-idf, which is the frequency of a term adjusted for how rarely it is used. We'll use tf-idf as a heuristic index to indicate how frequently a certain author uses a word relative to the frequency that ll the authors use the word. Therefore we will find words that are characteristic for a specific author, a good thing to have if we are interested in solving the author identification problem.

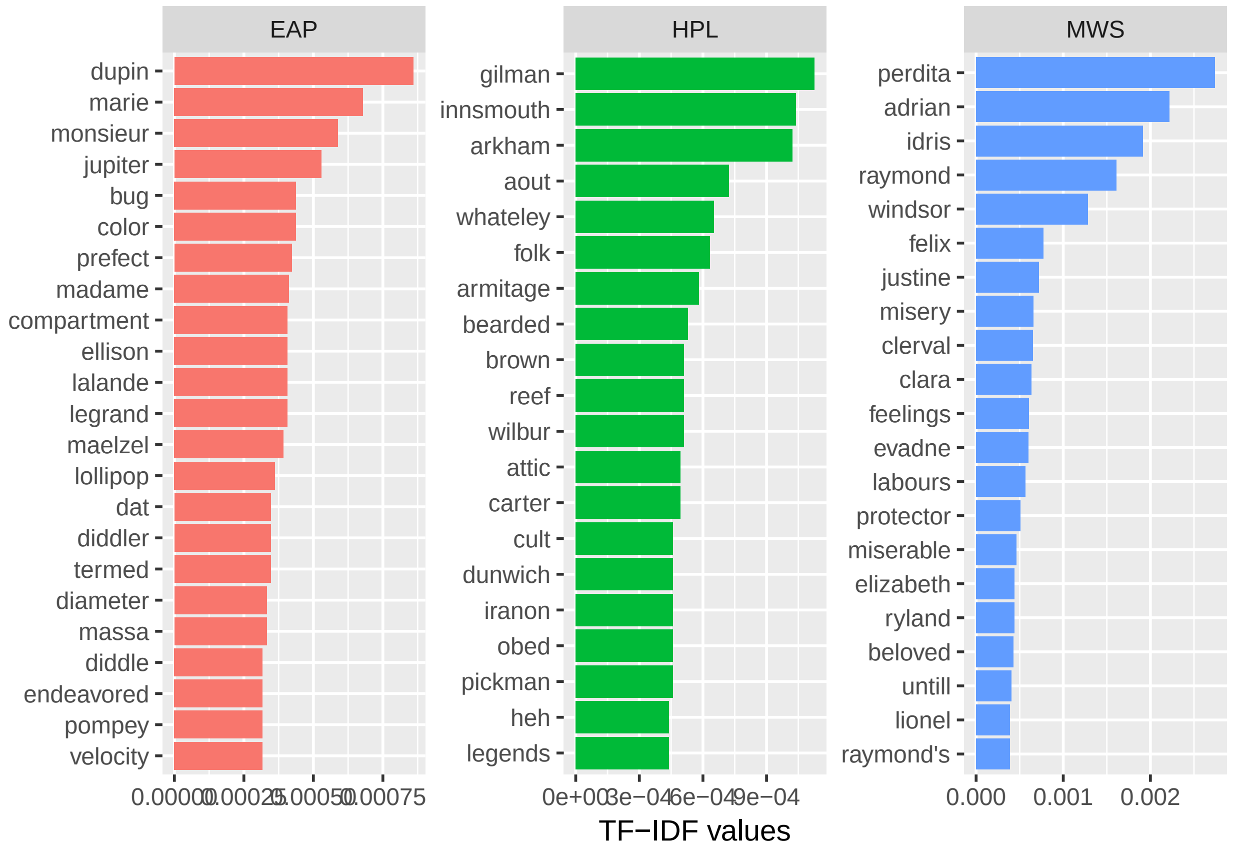
```
frequency <- count(spooky_wrd, author, word)
tf_idf    <- bind_tf_idf(frequency, word, author, n)

tf_idf <- arrange(tf_idf, desc(tf_idf))
tf_idf <- mutate(tf_idf, word = factor(word, levels = rev(unique(word))))
tf_idf_30 <- top_n(tf_idf, 30, tf_idf)

ggplot(tf_idf_30) +
  geom_col(aes(word, tf_idf, fill = author)) +
  labs(x = NULL, y = "TF-IDF values") +
  theme(legend.position = "top", axis.text.x  = element_text(angle=45, hjust=1, vjust=0.9))
```



Note that in the above, many of the words recognized by their tf-idf scores are names. This makes sense – if we see text referencing Raymond, Idris, or Perdita, we know almost for sure that MWS is the author. But some non-names stand out. EAP often uses "monsieur" and "jupiter" while HPL uses the words "bearded" and "attic" more frequently than the others. We can also look at the most characteristic terms per author.

```
tf_idf <- ungroup(top_n(group_by(tf_idf, author), 20, tf_idf))

ggplot(tf_idf) +
  geom_col(aes(word, tf_idf, fill = author)) +
  labs(x = NULL, y = "tf-idf") +
  theme(legend.position = "none") +
  facet_wrap(~ author, ncol = 3, scales = "free") +
  coord_flip() +
```

```r
  labs(y = "TF-IDF values")
```



Sentiment Analysis

We will use sentences as units of analysis for this tutorial, as sentences are natural languge units for organizing thoughts and ideas. For each sentence, we apply sentiment analysis using NRC sentiment lexion. "The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing."

From *Text Mining with R; A Tidy Approach*, "When human readers approach text, we use our understanding of the emotional intent of words to infer whether a section of text is positive or negative, or perhaps characterized by some other more nuanced emotion like surprise or disgust. We can also use the tools of text mining to approach the emotional content of text programmatically." This is the goal of sentiment analysis.

```r
get_sentiments('nrc')
```

```
## # A tibble: 13,901 x 2
##           word sentiment
##          <chr>     <chr>
##  1      abacus     trust
##  2     abandon      fear
##  3     abandon  negative
##  4     abandon   sadness
##  5   abandoned     anger
##  6   abandoned      fear
##  7   abandoned  negative
##  8   abandoned   sadness
##  9 abandonment     anger
## 10 abandonment      fear
## # ... with 13,891 more rows
```

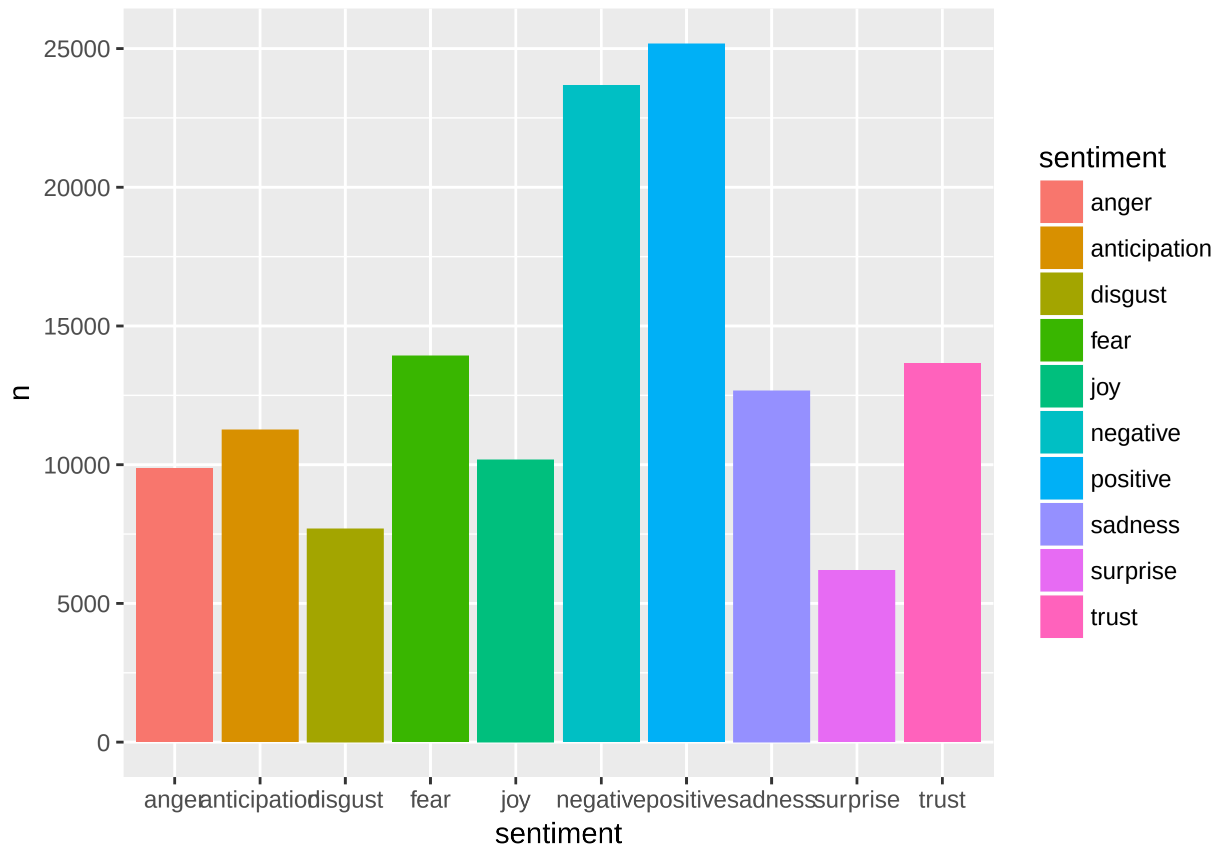
```r
sentiments <- inner_join(spooky_wrd, get_sentiments('nrc'), by = "word")

count(sentiments, sentiment)
```

```
## # A tibble: 10 x 2
##       sentiment     n
##           <chr> <int>
##  1        anger  9869
##  2 anticipation 11258
##  3      disgust  7697
##  4         fear 13927
##  5          joy 10190
##  6     negative 23674
##  7     positive 25175
##  8      sadness 12674
##  9     surprise  6199
## 10        trust 13655
```

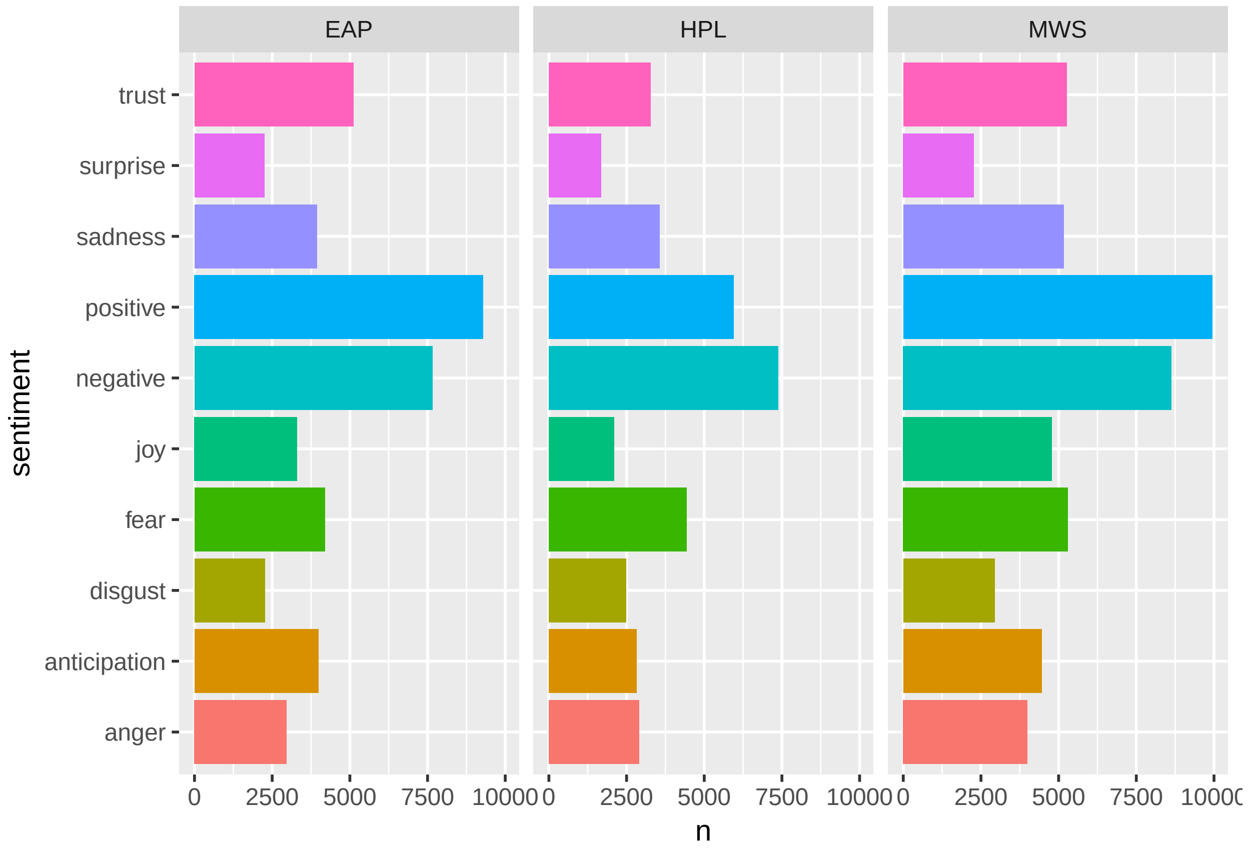
```r
count(sentiments, author, sentiment)
```

```
## # A tibble: 30 x 3
##    author    sentiment     n
##    <fctr>        <chr> <int>
##  1    EAP        anger  2962
##  2    EAP anticipation  3982
##  3    EAP      disgust  2261
##  4    EAP         fear  4194
##  5    EAP          joy  3302
##  6    EAP     negative  7659
##  7    EAP     positive  9291
##  8    EAP      sadness  3938
##  9    EAP     surprise  2244
## 10    EAP        trust  5116
## # ... with 20 more rows
```

```r
ggplot(count(sentiments, sentiment)) +
  geom_col(aes(sentiment, n, fill = sentiment))
```



```r
ggplot(count(sentiments, author, sentiment)) +
  geom_col(aes(sentiment, n, fill = sentiment)) +
  facet_wrap(~ author) +
  coord_flip() +
  theme(legend.position = "none")
```



Comparing Positivity

Let's only study the "positive" words. Note that the amount of "postive" words attributed to each author varies greatly, and the relative frequency of "positive" words to the other sentiments also varies between authors.

```
nrc_pos    <- filter(get_sentiments('nrc'), sentiment == "positive")
nrc_pos
```

```
## # A tibble: 2,312 x 2
##              word sentiment
##             <chr>     <chr>
##  1           abba  positive
##  2        ability  positive
##  3 abovementioned  positive
##  4       absolute  positive
##  5     absolution  positive
##  6       absorbed  positive
##  7      abundance  positive
##  8       abundant  positive
##  9       academic  positive
## 10        academy  positive
## # ... with 2,302 more rows
```

```
positive <- inner_join(spooky_wrd, nrc_pos, by = "word")
head(positive)
```

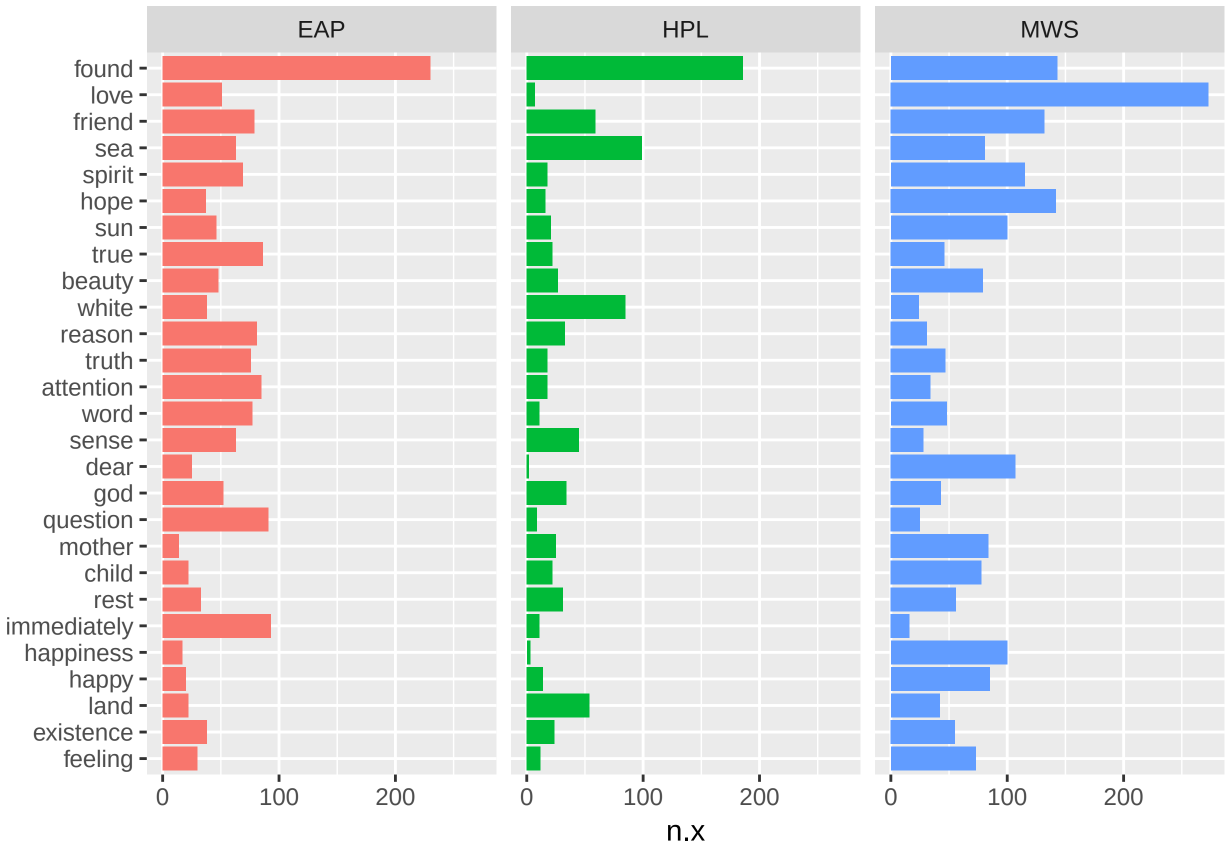
```
##        id author     word word_length sentiment
## 1 id11008    EAP     gold           4  positive
## 2 id27763    MWS   lovely           6  positive
## 3 id27763    MWS  fertile           7  positive
## 4 id27763    MWS    happy           5  positive
## 5 id27763    MWS cheering           8  positive
## 6 id27763    MWS     fair           4  positive
```

```
count(positive, word, sort = TRUE)
```

```
## # A tibble: 1,690 x 2
##      word     n
##     <chr> <int>
##  1  found   559
##  2   love   331
##  3 friend   270
##  4    sea   243
##  5 spirit   202
##  6   hope   195
##  7    sun   167
##  8 beauty   154
##  9   true   154
## 10  white   147
## # ... with 1,680 more rows
```

Now we plot a frequency comparison of these “positive” words. Namely, we show the frequencies of the overall most frequently used positive words split between the three authors.

```
pos_words     <- count(group_by(positive, word, author))
pos_words_all <- count(group_by(positive, word))

pos_words <- left_join(pos_words, pos_words_all, by = "word")
pos_words <- arrange(pos_words, desc(n.y))
pos_words <- ungroup(head(pos_words, 81))

# Note the above is the same as
# pos_words <- pos_words  %>%
#                  left_join(pos_words_all, by = "word") %>%
#                  arrange(desc(n.y)) %>%
#                  head(81) %>%
#                  ungroup()

ggplot(pos_words) +
  geom_col(aes(reorder(word, n.y, FUN = min), n.x, fill = author)) +
  xlab(NULL) +
  coord_flip() +
  facet_wrap(~ author) +
  theme(legend.position = "none")
```



Topic Models

Here's a good resource about topic modeling in R: https://eight2late.wordpress.com/2015/09/29/a-gentle-introduction-to-topic-modeling-using-r/. Generally, topic modeling is a method for unsupervised classification of documents by themes, similar to clustering on numeric data.

We're going to run LDA, or Latent Dirichlet Allocation, (note that LDA can also stand for Linear Discriminant Analysis, which we aren't studying here) which is one of the most popular algorithms for performing topic modelling. As noted in the tidytext texbook (*Text Mining with R; A Tidy Approach*), LDA makes two basic assumptions:

- 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.

Using these assumptions, 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.

We use the `topicmodels` package for this analysis. Since the `topicmodels` package doesn't use the `tidytext` framework, we first convert our `spooky_wrd` dataframe into a document term matrix (DTM) matrix using tidytext tools.

```
sent_wrd_freqs <- count(spooky_wrd, id, word)
head(sent_wrd_freqs)
```

```
## # A tibble: 6 x 3
##        id     word     n
##     <chr>    <chr> <int>
## 1 id00001  content     1
## 2 id00001    idris     1
## 3 id00001     mine     1
## 4 id00001  resolve     1
## 5 id00002 accursed     1
## 6 id00002     city     1
```

```
spooky_wrd_tm <- cast_dtm(sent_wrd_freqs, id, word, n)
spooky_wrd_tm
```

```
## <<DocumentTermMatrix (documents: 19467, terms: 24941)>>
## Non-/sparse entries: 193944/485332503
## Sparsity           : 100%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

```
length(unique(spooky_wrd$id))
```

```
## [1] 19467
```

```
length(unique(spooky_wrd$word))
```

```
## [1] 24941
```

The matrix spooky_wrd_tm is a sparse matrix with 19467 rows, corresponding to the 19467 ids (or originally, sentences) in the spooky_wrd dataframe, and 24941 columns corresponding to the total number of unique words in the spooky_wrd dataframe. So each row of spooky_wrd_tm corresponds to one of the original sentences. The value of the matrix at a certain position is then the number of occurences of that word (determined by the column) in this specific sentence (determined by the row). Since most sentence/word pairings don't occur, the matrix is sparse meaning there are many zeros.

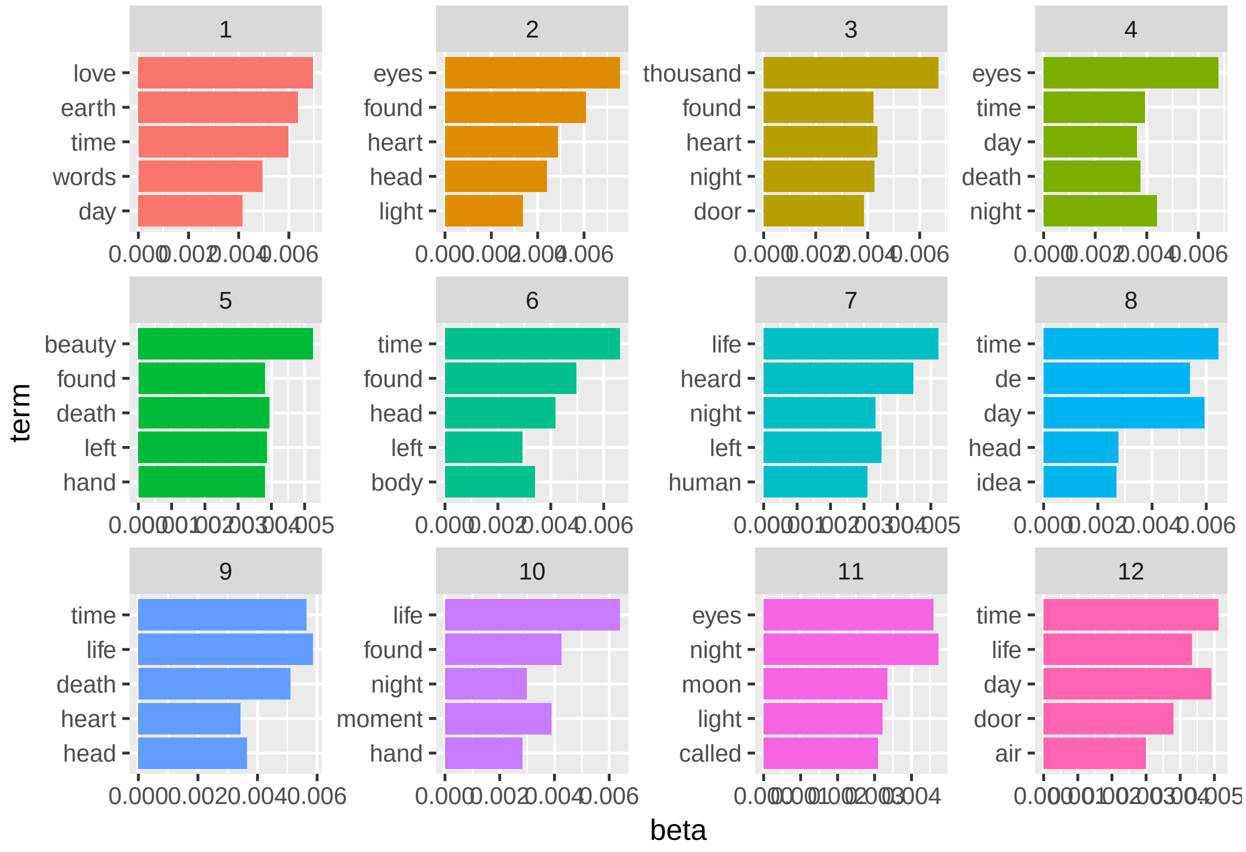
For LDA we must pick the number of possible topics. Let's try 12, though this selection is admittedly arbitrary.

```
spooky_wrd_lda    <- LDA(spooky_wrd_tm, k = 12, control = list(seed = 1234))
spooky_wrd_topics <- tidy(spooky_wrd_lda, matrix = "beta")
spooky_wrd_topics
```

```
## # A tibble: 299,292 x 3
##    topic    term         beta
##    <int>   <chr>        <dbl>
##  1     1 content 1.438913e-04
##  2     2 content 2.303686e-04
##  3     3 content 2.849487e-04
##  4     4 content 1.619628e-05
##  5     5 content 2.129616e-04
##  6     6 content 7.581265e-05
##  7     7 content 7.358757e-05
##  8     8 content 3.233463e-04
##  9     9 content 8.450580e-05
## 10    10 content 2.006677e-05
## # ... with 299,282 more rows
```

We note that in the above we use the `tidy` function to extract the per-topic-per-word probabilities, called "beta" or β, for the model. The final output has a one-topic-per-term-per-row format. For each combination, the model computes the probability of that term being generated from that topic. For example, the term "content" has a 1.619628×10^{-5} probability of being generated from topic 4. We visualizae the top terms (meaning the most likely terms associated with each topic) in the following.

```
spooky_wrd_topics_5 <- ungroup(top_n(group_by(spooky_wrd_topics, topic), 5, beta))
spooky_wrd_topics_5 <- arrange(spooky_wrd_topics_5, topic, -beta)
spooky_wrd_topics_5 <- mutate(spooky_wrd_topics_5, term = reorder(term, beta))

ggplot(spooky_wrd_topics_5) +
  geom_col(aes(term, beta, fill = factor(topic)), show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free", ncol = 4) +
  coord_flip()
```

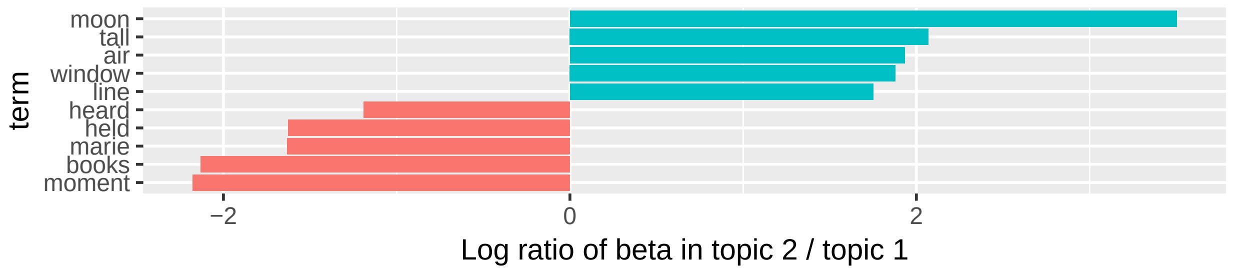


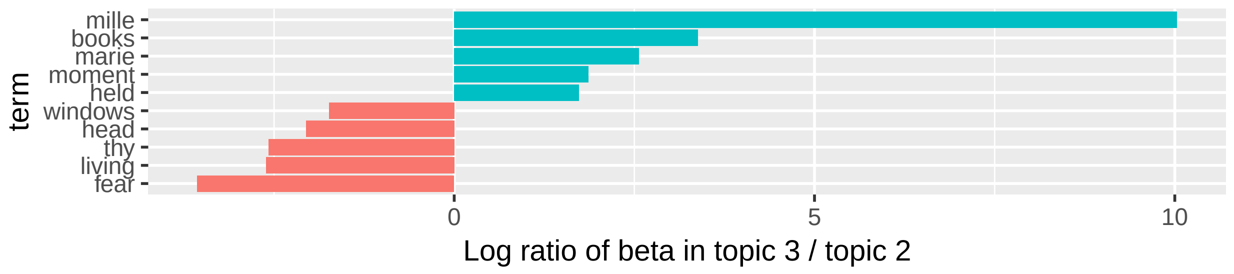
In the above, we see that the first topic is characterized by words like "love", "earth", and "words" while the third topic includes the word "thousand", and the fifth topic the word "beauty". Note that the words "eyes" and "time" appear in many topics. This is the advantage to topic modelling as opposed to clustering when using natural language – often a word may be likely to appear in documents characterized by multiple topics.

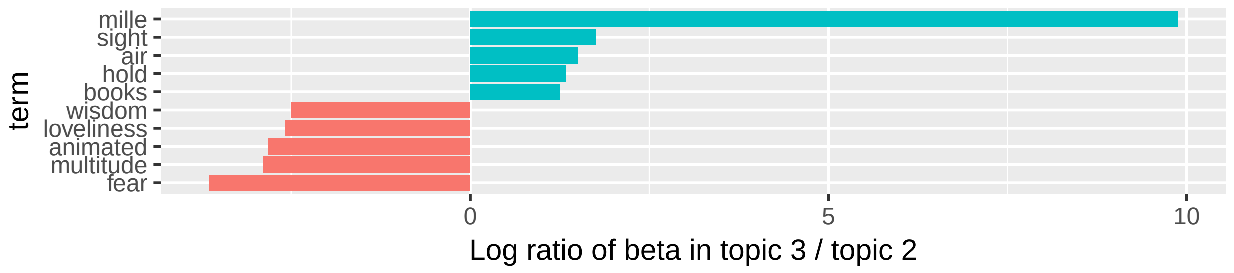
We can also study terms that have the greatest difference in probabilities between the topics, ignoring the words that are shared with similar frequency between topics. We choose only the first 3 topics as example and visualise the differences by plotting log ratios: $log_{10}(\beta$ of topic x $/\beta$ of topic y). So if a word is 10 times more frequent in topic x the log ratio will be 1, whereas it will be -1 if the word is 10 times more frequent in topic y.

```
spooky_wrd_topics <- mutate(spooky_wrd_topics, topic = paste0("topic", topic))
spooky_wrd_topics <- spread(spooky_wrd_topics, topic, beta)
```

```r
spooky_wrd_topics_12 <- filter(spooky_wrd_topics, topic2 > .001 | topic3 > .001)
spooky_wrd_topics_12 <- mutate(spooky_wrd_topics_12, log_ratio = log10(topic2 / topic1))
spooky_wrd_topics_12 <- group_by(spooky_wrd_topics_12, direction = log_ratio > 0)
spooky_wrd_topics_12 <- ungroup(top_n(spooky_wrd_topics_12, 5, abs(log_ratio)))
spooky_wrd_topics_12 <- mutate(spooky_wrd_topics_12, term = reorder(term, log_ratio))

p1 <- ggplot(spooky_wrd_topics_12) +
      geom_col(aes(term, log_ratio, fill = log_ratio > 0)) +
      theme(legend.position = "none") +
      labs(y = "Log ratio of beta in topic 2 / topic 1") +
      coord_flip()


spooky_wrd_topics_23 <- filter(spooky_wrd_topics, topic2 > .001 | topic3 > .001)
spooky_wrd_topics_23 <- mutate(spooky_wrd_topics_23, log_ratio = log10(topic3 / topic2))
spooky_wrd_topics_23 <- group_by(spooky_wrd_topics_23, direction = log_ratio > 0)
spooky_wrd_topics_23 <- ungroup(top_n(spooky_wrd_topics_23, 5, abs(log_ratio)))
spooky_wrd_topics_23 <- mutate(spooky_wrd_topics_23, term = reorder(term, log_ratio))

p2 <- ggplot(spooky_wrd_topics_23) +
      geom_col(aes(term, log_ratio, fill = log_ratio > 0)) +
      theme(legend.position = "none") +
      labs(y = "Log ratio of beta in topic 3 / topic 2") +
      coord_flip()

spooky_wrd_topics_13 <- filter(spooky_wrd_topics, topic3 > .001 | topic1 > .001)
spooky_wrd_topics_13 <- mutate(spooky_wrd_topics_13, log_ratio = log10(topic3 / topic1))
spooky_wrd_topics_13 <- group_by(spooky_wrd_topics_13, direction = log_ratio > 0)
spooky_wrd_topics_13 <- ungroup(top_n(spooky_wrd_topics_13, 5, abs(log_ratio)))
spooky_wrd_topics_13 <- mutate(spooky_wrd_topics_13, term = reorder(term, log_ratio))

p3 <- ggplot(spooky_wrd_topics_13) +
      geom_col(aes(term, log_ratio, fill = log_ratio > 0)) +
      theme(legend.position = "none") +
      labs(y = "Log ratio of beta in topic 3 / topic 2") +
      coord_flip()


layout <- matrix(c(1,2,3), 3, 1, byrow = TRUE)
multiplot(p1, p2, p3, layout = layout)
```







Much of the above work was adapted from this post: https://www.kaggle.com/headsortails/treemap-house-of-horror-spooky-eda-lda-features. It also has some other ideas about how to analyze the spooky data that we didn't have time to cover in the tutorial.

Readings for NLP with Python

- Natural Language Processing with Python
- A shorter tutorial
- Sentiment analysis
- Topic modeling