03 Analyzing Word and Document Frequency

As I mentioned earlier, there are words in a document, however, that occur many times but may not be important. In English, these are probably words like "the", "is", "of", and so forth. I might take the approach of adding words like these to a list of stop words and removing them before analysis, but it is possible that some of these words might be more important in some documents than others. A list of stop words is not a very sophisticated approach to adjusting term frequency for commonly used words.

My approach for this project is to look at a term's inverse document frequency (idf), which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents. This can be combined with term frequency to calculate a term's tf-idf (the two quantities multiplied together), the frequency of a term adjusted for how rarely it is used.

03_00 Background Information

The statistic tf-idf is intended to measure how important a word is to a document in a collection (or corpus) of documents, for example, to one novel in a collection of novels or to one website in a collection of websites.

It is a rule-of-thumb or heuristic quantity. While it has proved useful in text mining, search engines, etc., its theoretical foundations are considered less than firm by information theory experts. The inverse document frequency for any given term is defined as

idf(term) = ln((n of documents)*(n of documents containing term))

03_01 Term Frequency in Jane Austen's Novels

Let's start by looking at the published novels of Jane Austen and examine first term frequency, then tf-idf.

```
book_words <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word, sort = TRUE) %>%
  ungroup()

total_words <- book_words %>%
  group_by(book) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words)
```

```
## Joining, by = "book"
```

```
book_words
```

```
## # A tibble: 40,379 × 4
##                  book  word     n  total
##                <fctr> <chr> <int>  <int>
## 1      Mansfield Park   the  6206 160460
## 2      Mansfield Park    to  5475 160460
## 3      Mansfield Park   and  5438 160460
## 4                Emma    to  5239 160996
## 5                Emma   the  5201 160996
## 6                Emma   and  4896 160996
## 7      Mansfield Park    of  4778 160460
## 8   Pride & Prejudice   the  4331 122204
```

```
## 9                Emma     of   4291 160996
## 10 Pride & Prejudice     to   4162 122204
## # ... with 40,369 more rows
```

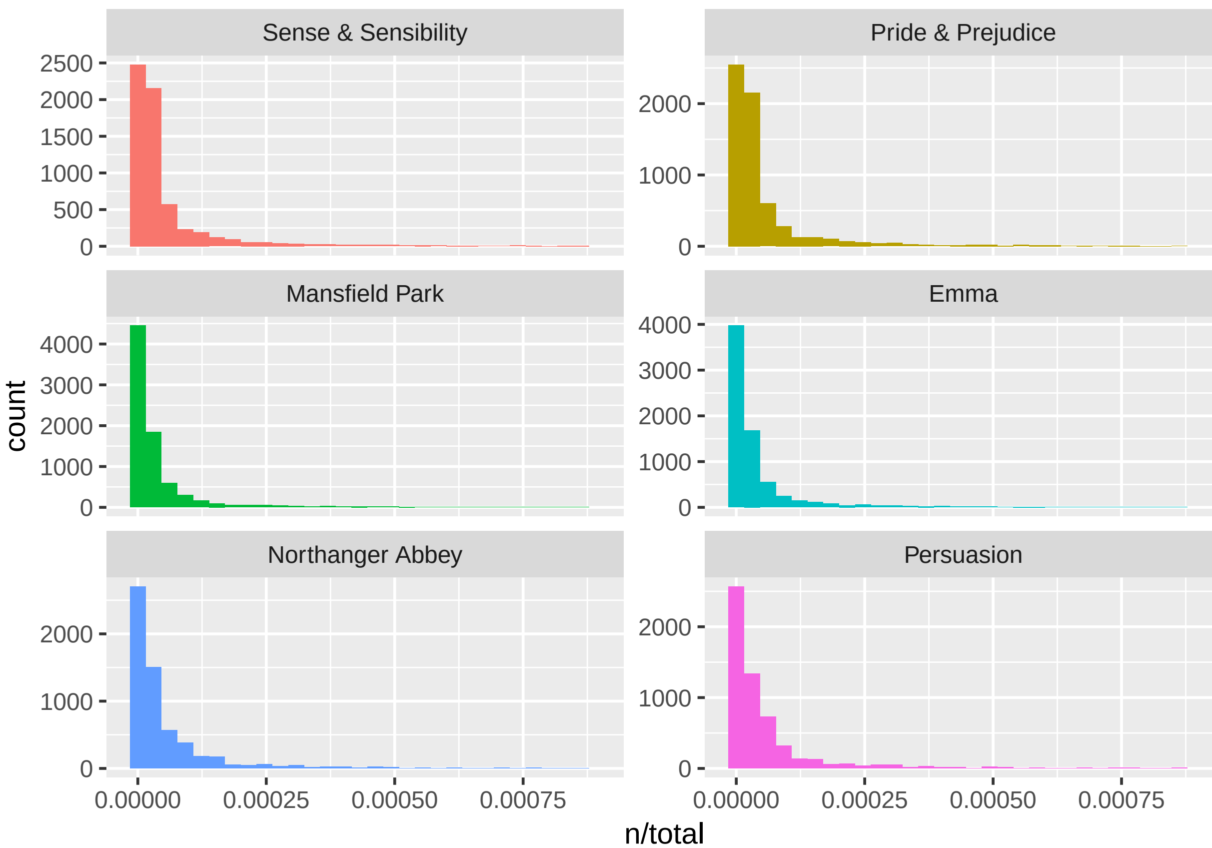
There is one row in this book_words data frame for each word-book combination; n is the number of times that word is used in that book and total is the total words in that book. The usual suspects are here with the highest n, "the", "and", "to", and so forth.

Next, I want to look at the distribution of n/total for each novel, the number of times a word appears in a novel divided by the total number of terms (words) in that novel. This is exactly what term frequency is.

```
ggplot(book_words, aes(n/total, fill = book)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~book, ncol = 2, scales = "free_y")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 896 rows containing non-finite values (stat_bin).
```



There are very long tails to the right for these novels (those extremely common words!) that I have not shown in these plots. These plots exhibit similar distributions for all the novels, with many words that occur rarely and fewer words that occur frequently.

03_02 Zipf's Law

Since I have the data frame I used to plot term frequency, I can examine Zipf's law for Jane Austen's novels with just a few lines of dplyr functions. Zipf's law states that the frequency that a word appears is inversely proportional to its rank.

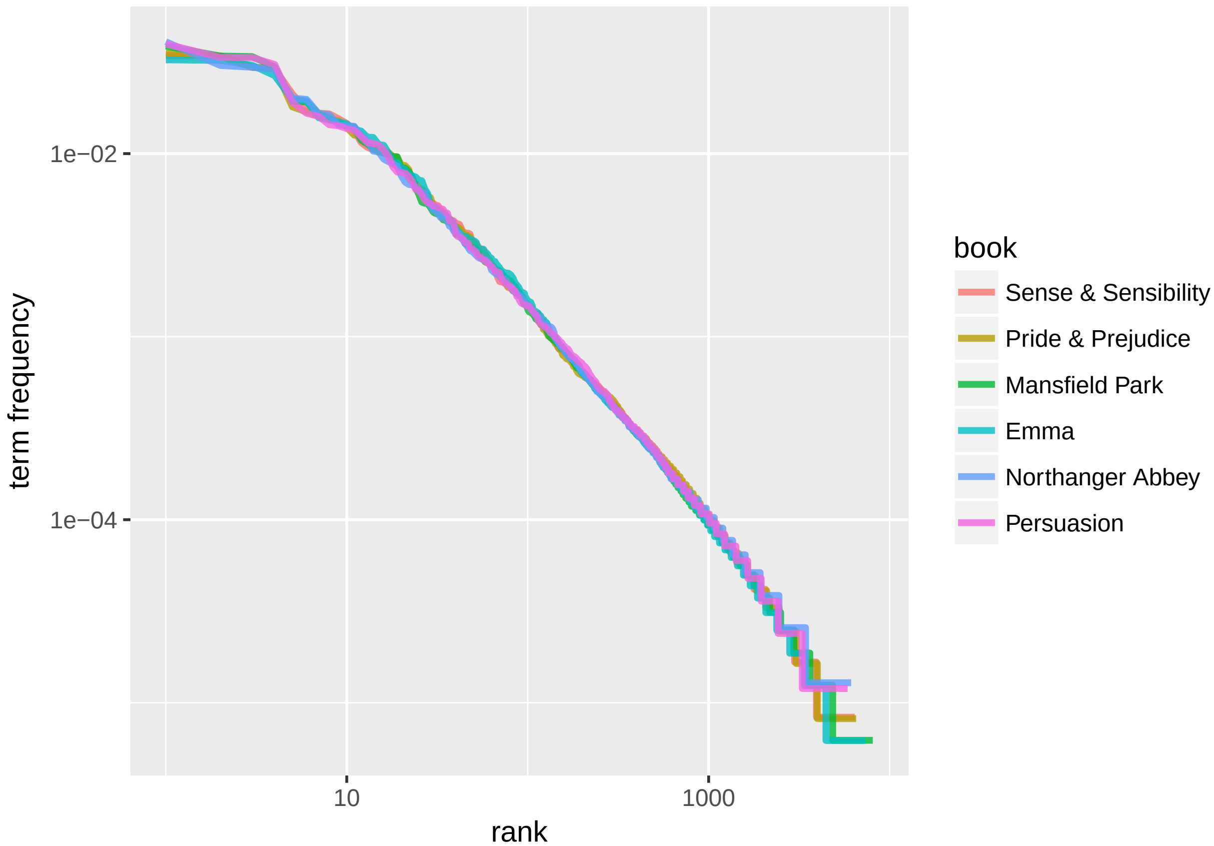
```r
freq_by_rank <- book_words %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)

freq_by_rank
```

```
## Source: local data frame [40,379 x 6]
## Groups: book [6]
##
##                  book  word     n  total  rank `term frequency`
##                <fctr> <chr> <int>  <int> <int>            <dbl>
## 1      Mansfield Park   the  6206 160460     1       0.03867631
## 2      Mansfield Park    to  5475 160460     2       0.03412065
## 3      Mansfield Park   and  5438 160460     3       0.03389007
## 4                Emma    to  5239 160996     1       0.03254118
## 5                Emma   the  5201 160996     2       0.03230515
## 6                Emma   and  4896 160996     3       0.03041069
## 7      Mansfield Park    of  4778 160460     4       0.02977689
## 8   Pride & Prejudice   the  4331 122204     1       0.03544074
## 9                Emma    of  4291 160996     4       0.02665284
## 10 Pride & Prejudice    to  4162 122204     2       0.03405780
## # ... with 40,369 more rows
```

The rank column here tells us the rank of each word within the frequency table; the table was already ordered by n so we could use row_number() to find the rank. Then, we can calculate the term frequency in the same way we did before.

The rank column here tells us the rank of each word within the frequency table; the table was already ordered by n so we could use row_number() to find the rank. Then, we can calculate the term frequency in the same way we did before.

```r
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = book)) +
  geom_line(size = 1.2, alpha = 0.8) +
  scale_x_log10() +
  scale_y_log10()
```



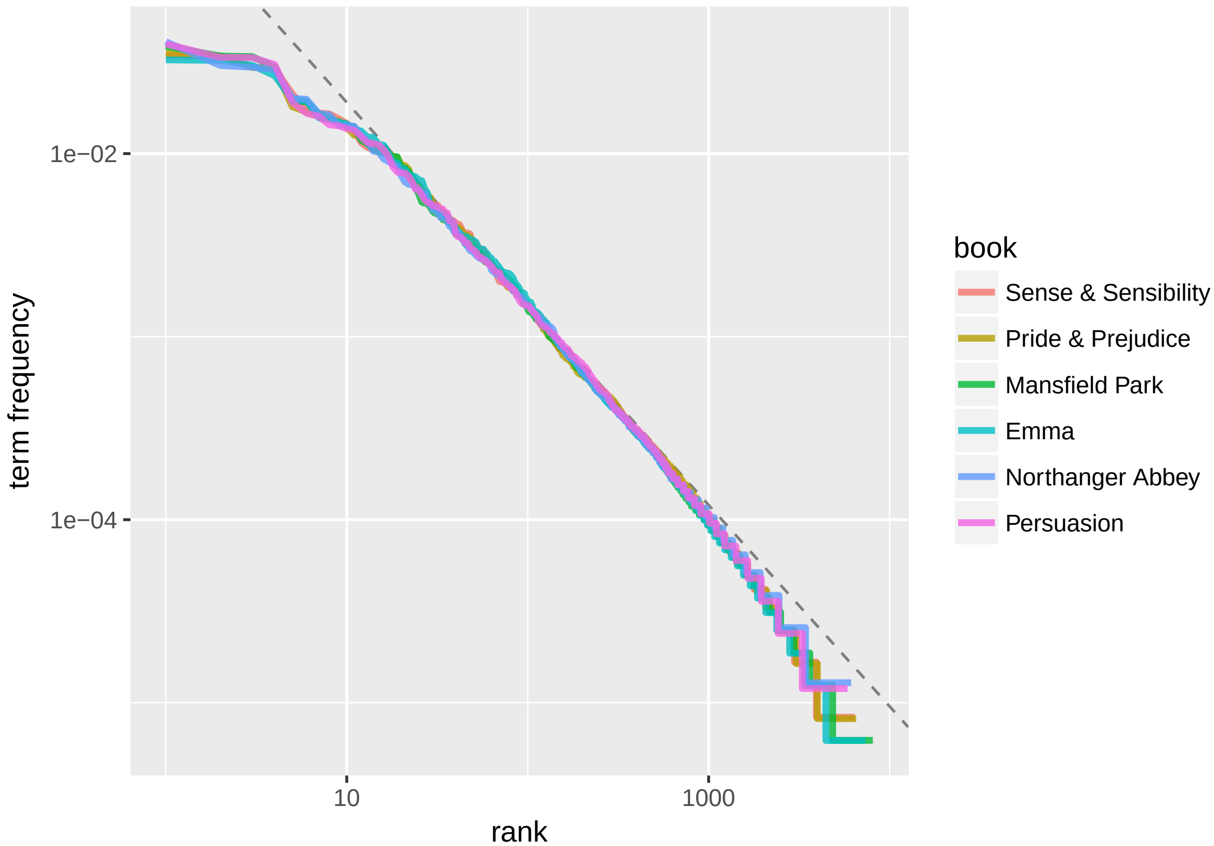
I see that all six of Jane Austen's novels are similar to each other, and that the relationship between rank and frequency does have negative slope. It is not quite constant, though. Perhaps I could view this as a broken power law with, say, three sections. Let's see what the exponent of the power law is for the middle section of the rank range.

```
rank_subset <- freq_by_rank %>%
  filter(rank < 500,
         rank > 10)

lm(log10(`term frequency`) ~ log10(rank), data = rank_subset)
```

```
##
## Call:
## lm(formula = log10(`term frequency`) ~ log10(rank), data = rank_subset)
##
## Coefficients:
## (Intercept)  log10(rank)
##     -0.6225      -1.1125
```

I want to plot this fitted power law with the data.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = book)) +
  geom_abline(intercept = -0.62, slope = -1.1, color = "gray50", linetype = 2) +
  geom_line(size = 1.2, alpha = 0.8) +
  scale_x_log10() +
  scale_y_log10()
```



I have found a result close to the classic version of Zipf's law for the corpus of Jane Austen's novels. The deviations I see here at high rank are not uncommon for many kinds of language. A corpus of language often contains fewer rare words than predicted by a single power law. The deviations at low rank are more unusual.

03_03 The bind_tf_idf Function

The idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents, in this case, the group of Jane Austen's novels as a whole. Calculating tf-idf attempts to find the words that are important (i.e., common) in a text, but not too common.

```
book_words <- book_words %>%
  bind_tf_idf(word, book, n)
book_words
```

```
## # A tibble: 40,379 × 7
##                  book  word     n  total         tf   idf tf_idf
##                <fctr> <chr> <int>  <int>      <dbl> <dbl>  <dbl>
## 1      Mansfield Park   the  6206 160460 0.03867631     0      0
## 2      Mansfield Park    to  5475 160460 0.03412065     0      0
## 3      Mansfield Park   and  5438 160460 0.03389007     0      0
## 4                Emma    to  5239 160996 0.03254118     0      0
## 5                Emma   the  5201 160996 0.03230515     0      0
## 6                Emma   and  4896 160996 0.03041069     0      0
## 7      Mansfield Park    of  4778 160460 0.02977689     0      0
## 8   Pride & Prejudice   the  4331 122204 0.03544074     0      0
## 9                Emma    of  4291 160996 0.02665284     0      0
```

```
## 10 Pride & Prejudice    to  4162 122204 0.03405780     0      0
## # ... with 40,369 more rows
```

Notice that idf and thus tf-idf are zero for these extremely common words. These are all words that appear in all six of Jane Austen's novels, so the idf term (which will then be the natural log of 1) is zero. The inverse document frequency (and thus tf-idf) is very low (near zero) for words that occur in many of the documents in a collection. This is how this approach decreases the weight for common words. The inverse document frequency will be a higher number for words that occur in fewer of the documents in the collection.

I look at terms with high tf-idf in Jane Austen's works.

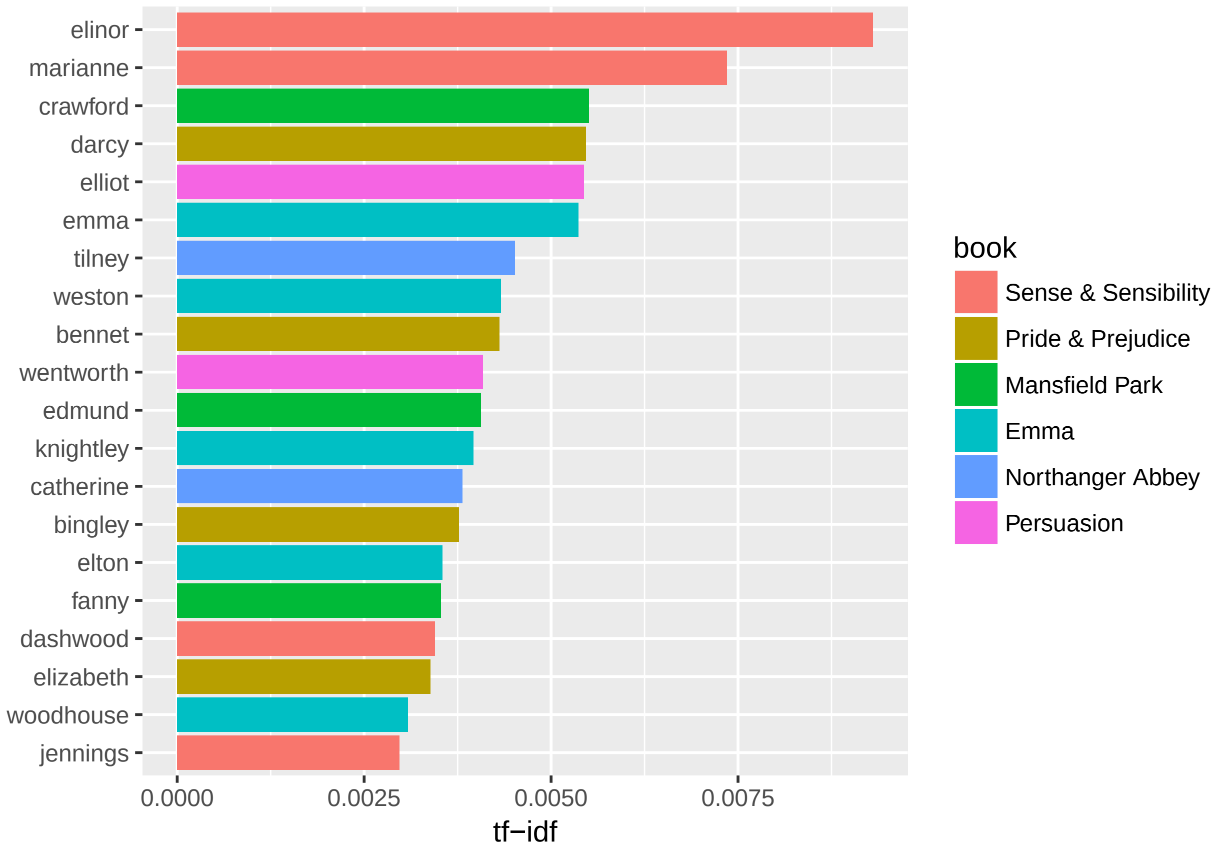
```
book_words %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 40,379 × 6
##                   book      word     n          tf      idf      tf_idf
##                 <fctr>     <chr> <int>       <dbl>    <dbl>       <dbl>
## 1  Sense & Sensibility    elinor   623 0.005193528 1.791759 0.009305552
## 2  Sense & Sensibility  marianne   492 0.004101470 1.791759 0.007348847
## 3       Mansfield Park  crawford   493 0.003072417 1.791759 0.005505032
## 4     Pride & Prejudice     darcy   373 0.003052273 1.791759 0.005468939
## 5           Persuasion    elliot   254 0.003036207 1.791759 0.005440153
## 6                 Emma      emma   786 0.004882109 1.098612 0.005363545
## 7      Northanger Abbey    tilney   196 0.002519928 1.791759 0.004515105
## 8                 Emma    weston   389 0.002416209 1.791759 0.004329266
## 9     Pride & Prejudice    bennet   294 0.002405813 1.791759 0.004310639
## 10          Persuasion wentworth   191 0.002283132 1.791759 0.004090824
## # ... with 40,369 more rows
```

I see all proper nouns, names that are in fact important in these novels. None of them occur in all of novels, and they are important, characteristic words for each text within the corpus of Jane Austen's novels.

Next, I want to look at a visualization for these high tf-idf words.

```
plot_austen <- book_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word))))

ggplot(plot_austen[1:20,], aes(word, tf_idf, fill = book)) +
  geom_col() +
  labs(x = NULL, y = "tf-idf") +
  coord_flip()
```

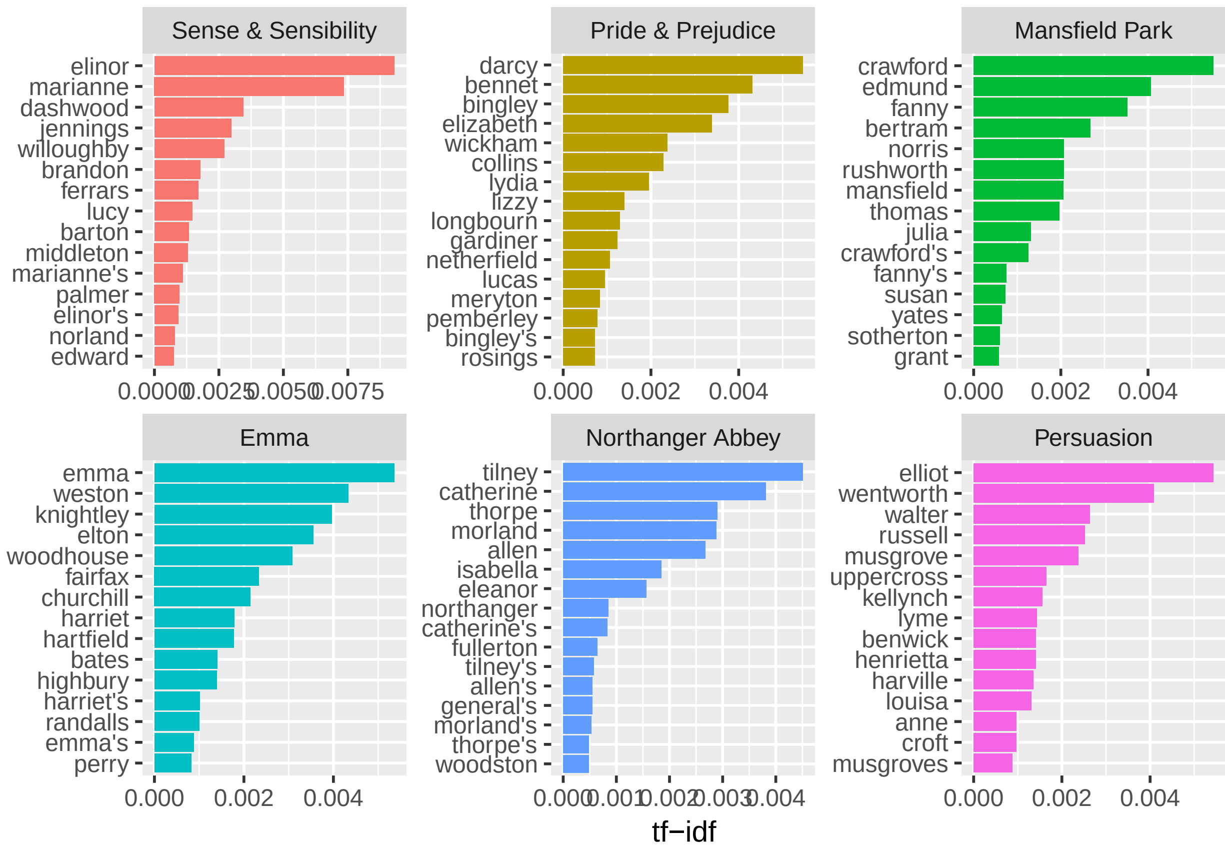


And I look at the novels individually.

```
plot_austen <- plot_austen %>%
  group_by(book) %>%
  top_n(15) %>%
  ungroup
```

```
## Selecting by tf_idf
```

```
ggplot(plot_austen, aes(word, tf_idf, fill = book)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~book, ncol = 3, scales = "free") +
  coord_flip()
```



These words are, as measured by tf-idf, the most important to each novel and most readers would likely agree. What measuring tf-idf has done here is show us that Jane Austen used similar language across her six novels, and what distinguishes one novel from the rest within the collection of her works are the proper nouns, the names of people and places. This is the point of tf-idf. It identifies words that are important to one document within a collection of documents.

03_04 A Carpus of Physics Texts

Further, I want to work with another corpus of documents, to see what terms are important in a different set of works. Leave the world of fiction and narrative entirely and download some classic physics texts from Project Gutenberg and see what terms are important in these works, as measured by tf-idf.

```
physics <- gutenberg_download(c(37729, 14725, 13476, 5001),
                              meta_fields = "author")
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

After I have the texts, I want to find out how many times each word was used in each text.

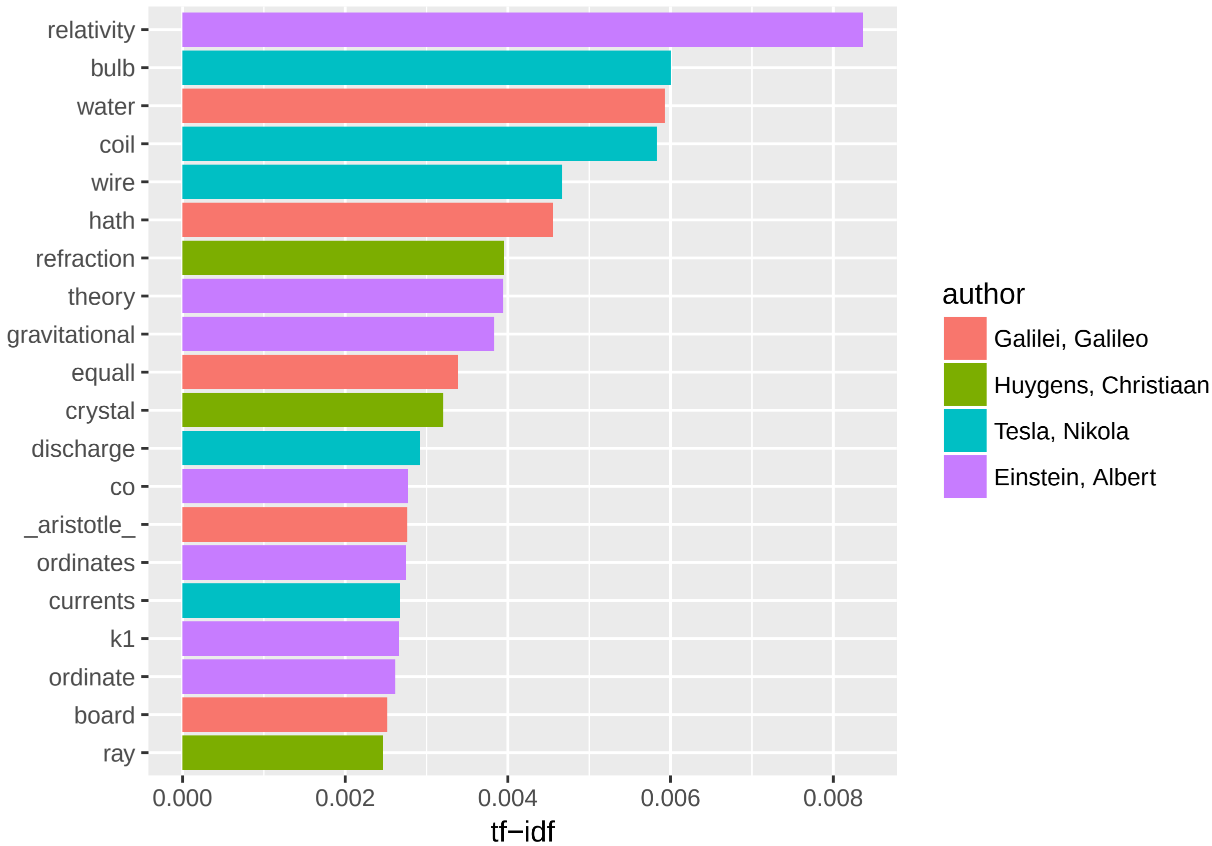
```
physics_words <- physics %>%
  unnest_tokens(word, text) %>%
  count(author, word, sort = TRUE) %>%
  ungroup()

physics_words
```

```
## # A tibble: 12,592 × 3
```

```
##                  author  word     n
##                   <chr> <chr> <int>
## 1     Galilei, Galileo   the  3760
## 2        Tesla, Nikola   the  3604
## 3  Huygens, Christiaan   the  3553
## 4     Einstein, Albert   the  2994
## 5     Galilei, Galileo    of  2049
## 6     Einstein, Albert    of  2030
## 7        Tesla, Nikola    of  1737
## 8  Huygens, Christiaan    of  1708
## 9  Huygens, Christiaan    to  1207
## 10       Tesla, Nikola     a  1176
## # ... with 12,582 more rows
```

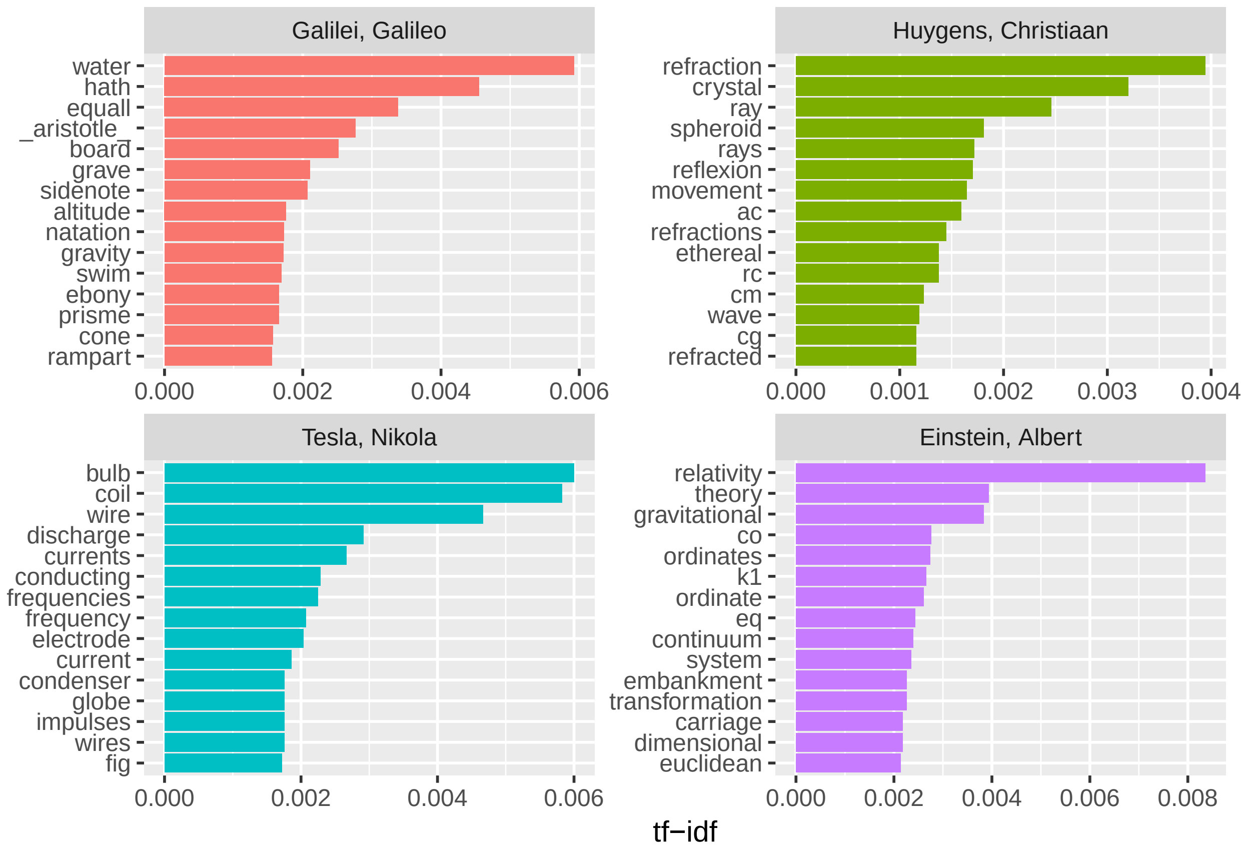
I see just the raw counts. I need to remember that these documents are all different lengths. Let's go ahead and calculate tf-idf, then visualize the high tf-id words.

```
physics_words <- physics_words %>%
  bind_tf_idf(word, author, n)

plot_physics <- physics_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

ggplot(plot_physics[1:20,], aes(word, tf_idf, fill = author)) +
  geom_col() +
  labs(x = NULL, y = "tf-idf") +
  coord_flip()
```



Next, I want to look at each text individually.

```
plot_physics <- plot_physics %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  mutate(word = reorder(word, tf_idf))

ggplot(plot_physics, aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~author, ncol = 2, scales = "free") +
  coord_flip()
```



I notice the word "eq" in the Einstein text, which is interesting. So I want to analyze this in detail.

```
physics %>%
  filter(str_detect(text, "eq\\.")) %>%
  select(text)
```

```
## # A tibble: 55 × 1
##                                                                       text
##                                                                      <chr>
## 1                                                  eq. 1: file eq01.gif
## 2                                                  eq. 2: file eq02.gif
## 3                                                  eq. 3: file eq03.gif
## 4                                                  eq. 4: file eq04.gif
## 5                                                eq. 05a: file eq05a.gif
## 6                                                eq. 05b: file eq05b.gif
## 7                        the distance between the points being eq. 06 .
## 8  direction of its length with a velocity v is eq. 06 of a metre.
## 9                                   velocity v=c we should have eq. 06a ,
## 10                  the rod as judged from K1 would have been eq. 06 ;
## # ... with 45 more rows
```

Some cleaning up of the text may be in order. "K1" is the name of a coordinate system for Einstein:

```
physics %>%
  filter(str_detect(text, "K1")) %>%
  select(text)
```

```
## # A tibble: 59 × 1
##                                                                              text
```

```
##                                                                      <chr>
## 1            to a second co-ordinate system K1 provided that the latter is
## 2           condition of uniform motion of translation. Relative to K1 the
## 3      tenet thus: If, relative to K, K1 is a uniformly moving co-ordinate
## 4    with respect to K1 according to exactly the same general laws as with
## 5  does not hold, then the Galileian co-ordinate systems K, K1, K2, etc.,
## 6   Relative to K1, the same event would be fixed in respect of space and
## 7   to K1, when the magnitudes x, y, z, t, of the same event with respect
## 8    of light (and of course for every ray) with respect to K and K1. For
## 9  reference-body K and for the reference-body K1. A light-signal is sent
## 10  immediately follows. If referred to the system K1, the propagation of
## # ... with 49 more rows
```

Maybe it makes sense to keep this one. Also notice that in this line I have "co-ordinate", which explains why there are separate "co" and "ordinate" items in the high tf-idf words for the Einstein text.

"AB", "RC", and so forth are names of rays, circles, angles, and so forth for Huygens.

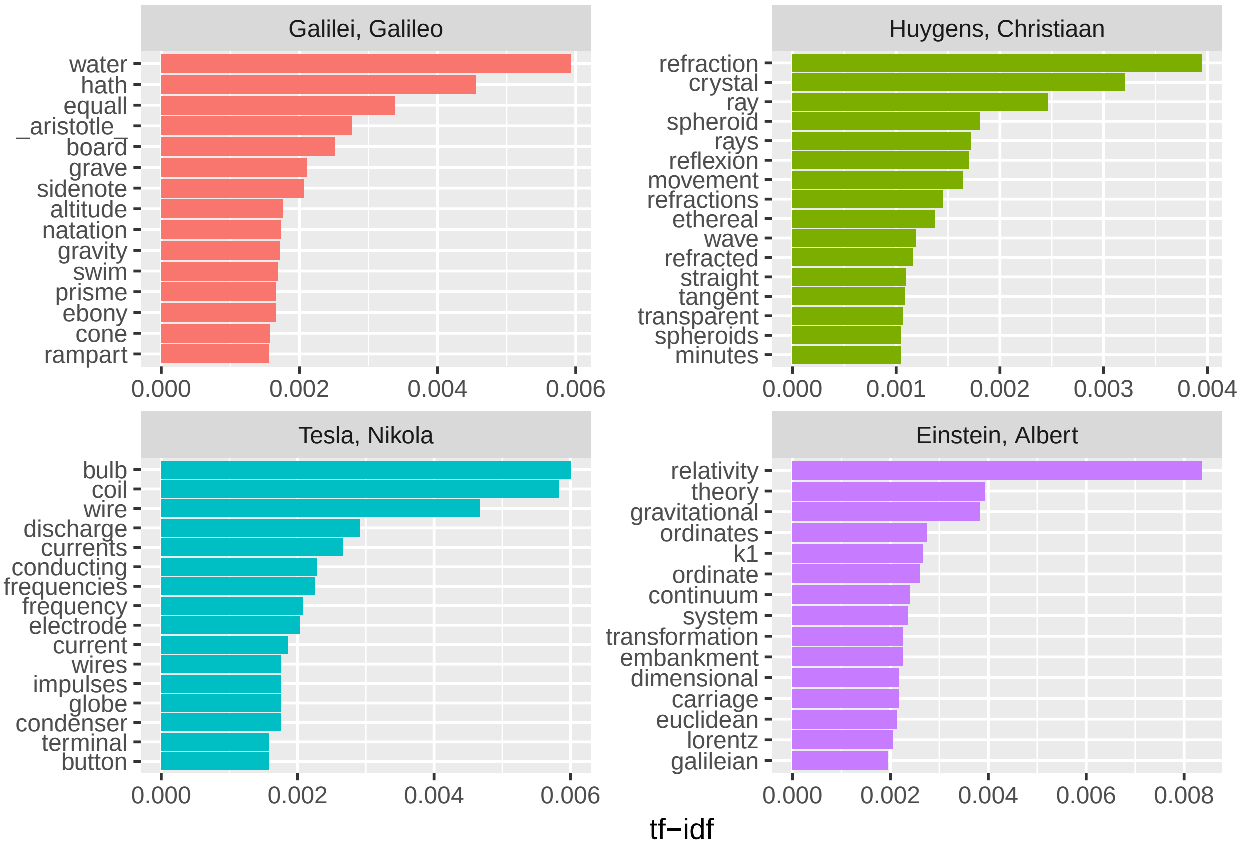
```
physics %>%
  filter(str_detect(text, "AK")) %>%
  select(text)
```

```
## # A tibble: 34 × 1
##                                                                       text
##                                                                      <chr>
## 1   Now let us assume that the ray has come from A to C along AK, KC; the
## 2    be equal to the time along KMN. But the time along AK is longer than
## 3  that along AL: hence the time along AKN is longer than that along ABC.
## 4      And KC being longer than KN, the time along AKC will exceed, by as
## 5      line which is comprised between the perpendiculars AK, BL. Then it
## 6  ordinary refraction. Now it appears that AK and BL dip down toward the
## 7  side where the air is less easy to penetrate: for AK being longer than
## 8    than do AK, BL. And this suffices to show that the ray will continue
## 9      surface AB at the points AK_k_B. Then instead of the hemispherical
## 10 along AL, LB, and along AK, KB, are always represented by the line AH,
## # ... with 24 more rows
```

I want to remove some of these less meaningful words to make a better, more meaningful plot. I will need to go back a few steps since I am removing words from the tidy data frame.

```
mystopwords <- data_frame(word = c("eq", "co", "rc", "ac", "ak", "bn",
                                   "fig", "file", "cg", "cb", "cm"))
physics_words <- anti_join(physics_words, mystopwords, by = "word")
plot_physics <- physics_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  ungroup %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

ggplot(plot_physics, aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
```

```
facet_wrap(~author, ncol = 2, scales = "free") +
coord_flip()
```



Summary

Using term frequency and inverse document frequency allowed me to find words that are characteristic for one document within a collection of documents, whether that document is a novel or physics text or webpage. Exploring term frequency on its own can give insight into how language is used in a collection of natural language, and dplyr verbs like count() and rank() give me tools to reason about term frequency. The tidytext package uses an implementation of tf-idf consistent with tidy data principles that enabled me to see how different words are important in documents within a collection or corpus of documents.