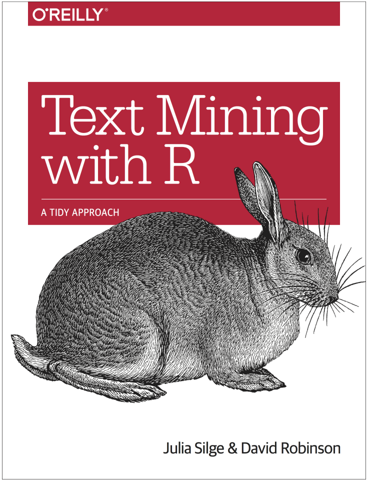
Socio-Informatics 348

Text Analysis
Analysing Word and Document Frequency

Dr Lisa Martin

Department of Information Science
Stellenbosch University

Today's Reading



Text Mining with R, Chapter 3

The Central Question

How do we quantify what a document is about?

One approach: Term Frequency (tf)

- How frequently a word occurs in a document
- Problem: Common words like "the", "is", "of" occur frequently but aren't very meaningful

Better approach: TF-IDF

- Combines term frequency (tf) with inverse document frequency (idf)
- Inverse Document Frequency (idf) helps address this: Decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents

Inverse Document Frequency (IDF)

Key idea: Decrease weight for commonly used words, increase weight for rare words

Formula:

$$\mathrm{idf}(\mathrm{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

TF-IDF:

$$\text{tf-idf} = \mathrm{tf} \times \mathrm{idf}$$

Measures how important a word is to a document in a collection (corpus) of documents

Example: Jane Austen's Novels

Computing Term Frequency

```r
library(dplyr)
library(janeaustenr)
library(tidytext)

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

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

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

Creates one row per word-book combination with counts

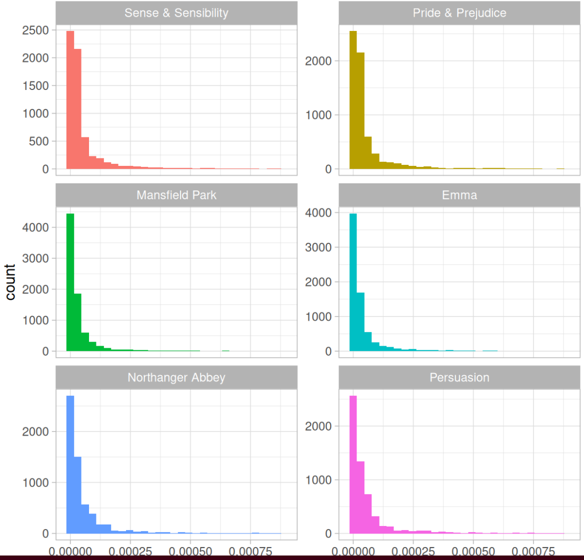
Most Common Words in Austen

```
book_words
#> # A tibble: 40,378 × 4
#>    book              word      n  total
#>    <fct>             <chr> <int>  <int>
#>  1 Mansfield Park    the    6206 160465
#>  2 Mansfield Park    to     5475 160465
#>  3 Mansfield Park    and    5438 160465
#>  4 Emma              to     5239 160996
#>  5 Emma              the    5201 160996
#>  6 Emma              and    4896 160996
#>  7 Mansfield Park    of     4778 160465
#>  8 Pride & Prejudice the    4331 122204
#>  9 Emma              of     4291 160996
#> 10 Pride & Prejudice to     4162 122204
#> # ℹ 40,368 more rows
```

The usual suspects: "the", "and", "to"

Term Frequency Distribution

Pattern observed: Term frequency $= n/\text{total}$



Term Frequency Distribution

Pattern observed:

- Many words occur rarely
- Few words occur frequently
- Long tails to the right (extremely common words)
- Similar distributions across all novels

This is typical in language—leads us to Zipf's law

Zipf's Law

Named after: George Zipf (20th century American linguist)

Statement: The frequency that a word appears is inversely proportional to its rank

$$\text{frequency} \propto \frac{1}{\text{rank}}$$

Common in natural language corpora (books, websites, speech)

Testing Zipf's Law on Austen

```
freq_by_rank <- book_words %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         term_frequency = n/total) %>%
  ungroup()

freq_by_rank
#> # A tibble: 40,378 × 6
#>    book              word      n  total  rank term_frequency
#>    <fct>             <chr> <int>  <int> <int>          <dbl>
#>  1 Mansfield Park    the    6206 160465     1         0.0387
#>  2 Mansfield Park    to     5475 160465     2         0.0341
#>  3 Mansfield Park    and    5438 160465     3         0.0339
#>  4 Emma              to     5239 160996     1         0.0325
#>  5 Emma              the    5201 160996     2         0.0323
#>  6 Emma              and    4896 160996     3         0.0304
#>  7 Mansfield Park    of     4778 160465     4         0.0298
#>  8 Pride & Prejudice the    4331 122204     1         0.0354
#>  9 Emma              of     4291 160996     4         0.0267
#> 10 Pride & Prejudice to     4162 122204     2         0.0341
```

Testing Zipf's Law on Austen

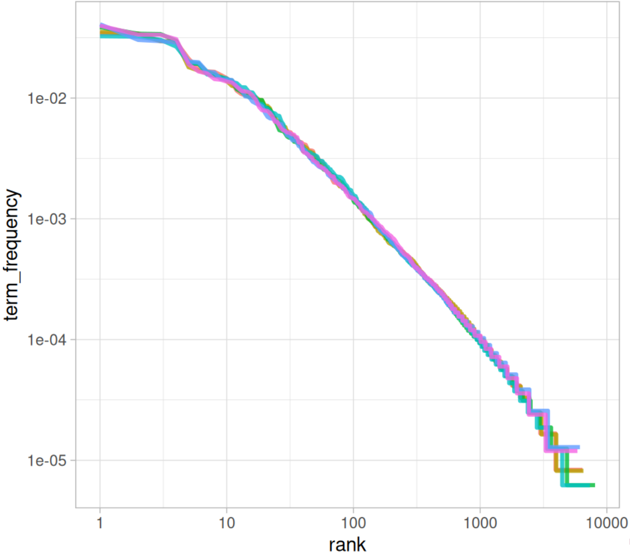
Visualisation: Plot rank vs. term frequency on log-log scale

- Inversely proportional relationship shows constant negative slope
- Austen novels show slope close to -1

```r
freq_by_rank %>%
  ggplot(aes(rank, term_frequency, color = book)) +
  geom_line(linewidth = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```

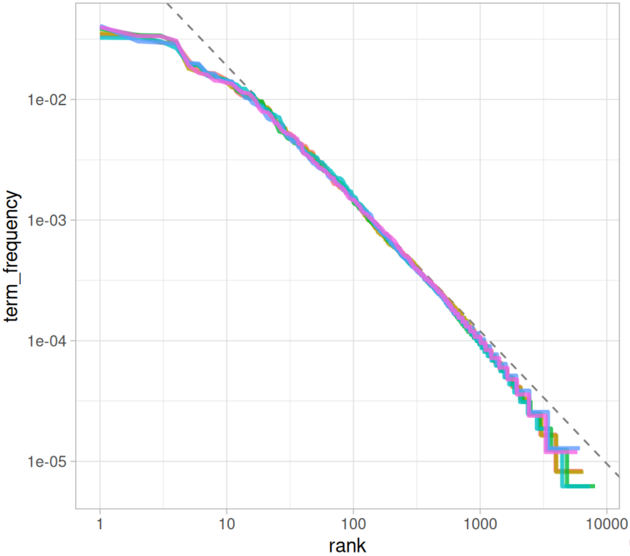
Testing Zipf's Law on Austen

Visualisation: Plot rank vs. term frequency on log-log scale



Fitting Zipf's Law

Visualisation: Slope very close to -1



Computing TF-IDF with tidytext

The `bind_tf_idf()` function requires:

- One row per token (term) per document
- Column with terms/tokens
- Column with documents
- Column with counts
- Adds three columns: `tf`, `idf`, `tf_idf`

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


book_tf_idf
#> # A tibble: 40,378 × 7
#>    book              word      n  total     tf   idf tf_idf
#>    <fct>             <chr> <int>  <int>  <dbl> <dbl>  <dbl>
#>  1 Mansfield Park    the    6206 160465 0.0387     0      0
#>  2 Mansfield Park    to     5475 160465 0.0341     0      0
#>  3 Mansfield Park    and    5438 160465 0.0339     0      0
#>  4 Emma              to     5239 160996 0.0325     0      0
```

Interpreting TF-IDF

Common words (appear in all documents):

- $\text{IDF} = \ln(6/6) = \ln(1) = 0$
- $\text{TF-IDF} = 0$
- Examples: "the", "to", "and", "of"

Rare words (appear in few documents):

- Higher IDF values
- Higher TF-IDF scores
- These are the *important* words that distinguish documents

Highest TF-IDF Words: Jane Austen

All proper nouns—character names that distinguish each novel:

```
book_tf_idf %>%
  select(-total) %>%
  arrange(desc(tf_idf))
#> # A tibble: 40,378 × 6
#>    book                word           n      tf   idf  tf_idf
#>    <fct>               <chr>      <int>   <dbl> <dbl>   <dbl>
#>  1 Sense & Sensibility elinor       623 0.00519  1.79 0.00931
#>  2 Sense & Sensibility marianne     492 0.00410  1.79 0.00735
#>  3 Mansfield Park      crawford     493 0.00307  1.79 0.00550
#>  4 Pride & Prejudice   darcy        373 0.00305  1.79 0.00547
#>  5 Persuasion          elliot       254 0.00304  1.79 0.00544
#>  6 Emma                emma         786 0.00488  1.10 0.00536
#>  7 Northanger Abbey    tilney       196 0.00252  1.79 0.00452
#>  8 Emma                weston       389 0.00242  1.79 0.00433
#>  9 Pride & Prejudice   bennet       294 0.00241  1.79 0.00431
#> 10 Persuasion          wentworth    191 0.00228  1.79 0.00409
```

What TF-IDF Reveals About Austen

Finding: Jane Austen used similar language across all six novels

What distinguishes one novel from another:

- Proper nouns
- Names of people and places
- Character-specific vocabulary

Point of TF-IDF: Identifies words that are important to one document within a collection

A Corpus of Physics Texts

Four classic physics texts from Project Gutenberg:

1. Galileo Galilei: *Discourse on Floating Bodies*
2. Christiaan Huygens: *Treatise on Light*
3. Nikola Tesla: *Experiments with Alternate Currents*
4. Albert Einstein: *Relativity: Special and General Theory*

Diversity:

- 300-year timespan
- Different languages (translated to English)
- Different physics topics

```r
library(gutenbergr)
physics <- gutenberg_download(c(37729, 14725, 13476, 30155),
                              meta_fields = "author")
```

Processing Physics Texts

First, tokenize and count words per author:

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

physics_words
#> # A tibble: 12,667 × 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    2993
#>  5 Galilei, Galileo    of     2049
#>  6 Einstein, Albert    of     2028
#>  7 Tesla, Nikola       of     1737
#>  8 Huygens, Christiaan of     1708
```

Processing Physics Texts

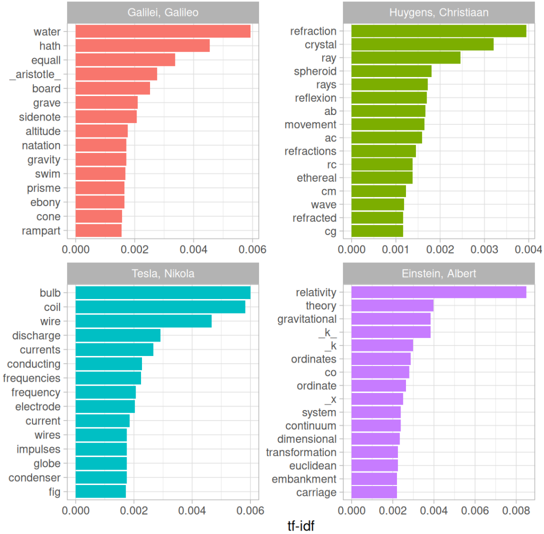
Then compute TF-IDF per author:

```
plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

plot_physics %>%
  group_by(author) %>%
  slice_max(tf_idf, n = 15) %>%
  ungroup() %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(tf_idf, word, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = "tf-idf", y = NULL) +
  facet_wrap(~author, ncol = 2, scales = "free")
```

Processing Physics Texts

Then visualise highest scoring words per author:



Interesting Findings in Physics Texts

High TF-IDF words reveal:

- Technical terms specific to each author's work
- Galileo: geometry terms, "water", "solid"
- Huygens: optics terms, "refraction", "waves"
- Tesla: electrical terms, "coil", "frequency"
- Einstein: relativity terms, "coordinate", "observer"

Data cleaning challenges:

- Diagram labels (AB, RC, etc.) appearing as high TF-IDF
- Mathematical notation artifacts (_k_)
- Need custom stop words for meaningful results

Removing Uninformative Terms

Flexible approach: create custom stop word lists for specific corpora

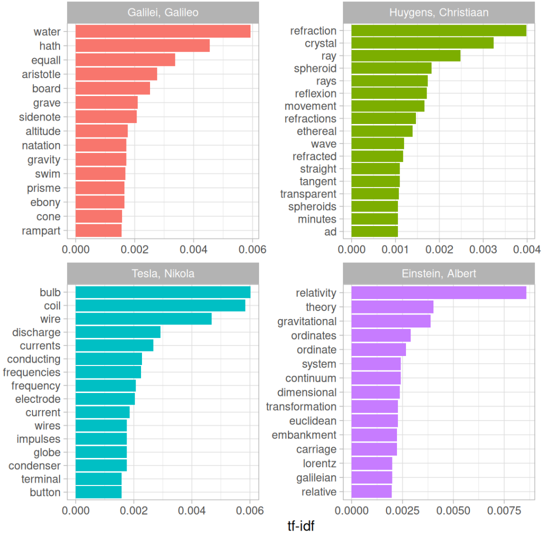
```
mystopwords <- tibble(word = c("eq", "co", "rc", "ac", "ak", "bn",
                                   "fig", "file", "cg", "cb", "cm",
                               "ab", "_k", "_k_", "_x"))

physics_words <- anti_join(physics_words, mystopwords,
                           by = "word")

plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(word = str_remove_all(word, "_")) %>%
  group_by(author) %>%
  slice_max(tf_idf, n = 15) %>%
  ungroup() %>%
  mutate(word = fct_reorder(word, tf_idf)) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))
```

Physics TF-IDF After Cleaning

More meaningful high TF-IDF terms:



Physics TF-IDF After Cleaning

More meaningful high TF-IDF terms:

- Domain-specific technical vocabulary
- Concepts central to each author's work
- Words that distinguish one physicist's writing from others