UPPSALA
UNIVERSITET

Text analysis III: Building a tidytext toolbox

Introduction to R for Social Sciences

Josef Ginnerskov
Department of Sociology
2022-08-23

Today's outline

1. A conceptual background to (computational) text analysis in R

2. Basic text analysis tasks performed on vector strings following the logic of the tm package

3. **More advance text analysis tasks conducted on digitized books based on the tidytext package**

4. Individually solving the problem set by building your own Gutenberg corpus

Loading books from Gutenberg (1/4)

GutenbergR is a package that lets us harvest the many fruits of the digital archive www.gutenberg.org as full texts with metadata.

```
library(tidyverse) # load the familiar tidyverse package
library(gutenbergr) # loading the Gutenberg R package

gutenberg_metadata # a tibble of the 51 k works with 8 metadata
gutenberg_authors # a tibble of the 16 k authors with 7 metadata
gutenberg_works() # a tibble of the 40 k English works with 8 metadata

gutenberg_works(author == "Dostoyevsky, Fyodor") # filter for only English translations of Dostoyevsky
```

```
## # A tibble: 12 × 8
##    gutenberg_id title  author  gutenberg_autho… language gutenberg_books… rights
##           <int> <chr>  <chr>              <int> <chr>    <chr>            <chr>
##  1          600 "Note… Dostoy…              314 en       <NA>             Publi…
##  2         2197 "The … Dostoy…              314 en       <NA>             Publi…
##  3         2302 "Poor… Dostoy…              314 en       <NA>             Publi…
##  4         2554 "Crim… Dostoy…              314 en       Best Books Ever… Publi…
##  5         2638 "The … Dostoy…              314 en       Best Books Ever… Publi…
##  6         8117 "The … Dostoy…              314 en       Best Books Ever… Publi…
##  7         8578 "The … Dostoy…              314 en       Racism           Publi…
##  8        28054 "The … Dostoy…              314 en       Best Books Ever… Publi…
##  9        36034 "Whit… Dostoy…              314 en       <NA>             Publi…
## 10        37536 "The … Dostoy…              314 en       <NA>             Publi…
## 11        38241 "Uncl… Dostoy…              314 en       <NA>             Publi…
## 12        40745 "Shor… Dostoy…              314 en       <NA>             Publi…
## # … with 1 more variable: has_text <lgl>
```

Loading books from Gutenberg (2/4)

To engage in text analysis, we need to build ourselves a test corpus. The code below is set to download six sociology books from www.gutenberg.org.

```
Socbooks.raw <- gutenberg_download(c(41360, 46423, 13205, 30610, 6568, 21609), # downloading books by id (Durkheim,
   Marx, Geddes, Blackmar, Ellwood and Rowe)
                                   mirror = "http://mirrors.xmission.com/gutenberg/",# when loading several books at
   once you might need to state a proper mirror...
                                   meta_fields = c("author", "title"), strip = TRUE) # include author and title
   metadata; "strip" entails keeping only the main book text

head(Socbooks.raw) # let's see what we got
```

```
## # A tibble: 6 × 4
##   gutenberg_id text                                    author        title
##          <int> <chr>                                   <chr>         <chr>
## 1         6568 "SOCIOLOGY AND MODERN SOCIAL PROBLEMS" Ellwood, Cha… Sociology a…
## 2         6568 ""                                     Ellwood, Cha… Sociology a…
## 3         6568 "BY"                                   Ellwood, Cha… Sociology a…
## 4         6568 ""                                     Ellwood, Cha… Sociology a…
## 5         6568 "CHARLES A. ELLWOOD, PH. D."           Ellwood, Cha… Sociology a…
## 6         6568 ""                                     Ellwood, Cha… Sociology a…
```

Loading books from Gutenberg (3/4)

A first feeling for the volume of the books before proceeding with the tidytext package.

```
Socbooks.raw %>% # yay, we can pipe it
  count(title) # counting how many rows that are in each book
```

```
## # A tibble: 6 × 2
##   title                                                    n
##   <chr>                                                <int>
## 1 A Contribution to The Critique Of The Political Economy  8467
## 2 Civics: as Applied Sociology                             4252
## 3 History of Human Society                                18682
## 4 Society: Its Origin and Development                     13686
## 5 Sociology and Modern Social Problems                     8928
## 6 The Elementary Forms of the Religious Life              22685
```

Loading books from Gutenberg (4/4)

Unfortunately, there are often a lot of unwanted signs in old books that will mess up the text analysis and thus need to be removed pronto.

```
Socbooks.sub <- Socbooks.raw # to keep things in order we create an object for substituting odd signs

Socbooks.sub$text <- gsub("_","",as.character(Socbooks.sub$text)) # substitute _ for nothing in the text column
Socbooks.sub$text <- gsub("--"," ",as.character(Socbooks.sub$text)) # substitute -- for a blank space in the text
   column
```

Turning books into a tidy tibble

In this analysis, we are goind to work with tidytext, which is a package set out to make text analysis friendly for the tidyverse universe. This is great since we can then add on other tidyverse packages like ggplot2 (you can later return to Pablo Lillo Cea's lecture on visualization to refresh your ggplot skills).

```
library(tidytext) # load tidytext
Socbooks.tidy <- Socbooks.sub %>% # we work with the substituted corpus
    mutate(line = row_number()) %>% # by mutating, we can keep information on what line a word resides
    unnest_tokens(word, text) # turning the book into a tall tibble where each word or token makes up a row (also
   preprocesses)
                              # can be modified to split by n-grams, sentences, lines, paragraphs, tweets...
head(Socbooks.tidy) # so, what happend?
```

```
## # A tibble: 6 × 5
##   gutenberg_id author                              title              line word
##          <int> <chr>                               <chr>             <int> <chr>
## 1         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     1 socio…
## 2         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     1 and
## 3         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     1 modern
## 4         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     1 social
## 5         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     1 probl…
## 6         6568 Ellwood, Charles A. (Charles Abram) Sociology and M…     3 by
```

Removing stop words

For most text analysis tasks, words that are too common do not only hold no meaning but are disruptive and cause an obstacle for performing the methods.

```
Socbooks.tidy.stop <- Socbooks.tidy %>% # tidytext comes with its own list "stop_words" that in most cases works great
  anti_join(stop_words) # anti_join from dplyr can be used to remove these words

Socbooks.stop.words <- tibble(word = # as expected, some academic slang ought to be removed as well
                                c("p", "cit", "tr", "pp", "ff", "nat", "ibid", "geddes", "prof", "per"))

Socbooks.tidy.stop <- Socbooks.tidy.stop %>% # remove your own set of stop words from the filtered tibble
  anti_join(Socbooks.stop.words)
```

Basic global/corpus count

With our tidy corpus at hand we can begin the analysis. Let's look at the top words for the corpus.

```
Socbooks.tidy.stop %>%
    count(word, sort = TRUE) %>% # we count the words and sort the counts from high to low
    top_n(10) # since this will be overwhelming we only look at the top 10
```

```
## # A tibble: 10 × 2
##    word           n
##    <chr>      <int>
##  1 social      2471
##  2 life        1981
##  3 time        1166
##  4 society     1164
##  5 people      1075
##  6 family       992
##  7 religious    970
##  8 individual   942
##  9 form         915
## 10 money        912
```

Generating a word cloud (1/2)

In the previous lecture, we saw that word clouds can bring some visual insights into the data, so why don't we generate a word cloud for the whole corpus?

```
library(wordcloud) # there is a specific package for word clouds - load it!

Socbooks.tidy.stop %>% # piping the clean tibble
  count(word) %>% # count all words
  with(wordcloud(word, n, # adding on the base R command "with"
                 max.words = 100, # limit the cloud to 100 words
                 colors = brewer.pal(8, "Dark2"))) # add our favorite color palette
```

Generating a word cloud (2/2)

Basic local/document count

Often you would quickly want to move from the global corpus level to the local document level. We can do the same count as previously but filter for a specific book or author and so on.

```
Socbooks.tidy.stop %>%
    filter(author == "Marx, Karl") %>% # filter by author, specifically Marx
    count(word, sort = TRUE) %>% # again, count words and sort from high to low
    top_n(10) # only the top 10
```

```
## # A tibble: 10 × 2
##    word            n
##    <chr>       <int>
##  1 money         834
##  2 gold          649
##  3 commodities   621
##  4 circulation   562
##  5 exchange      545
##  6 labor         523
##  7 commodity     434
##  8 production    414
##  9 form          335
## 10 time          328
```

Comparing word occurences per book

The next natural step is to compare the books. With the code below we can check out the most frequently used words book by book.

```
Socbooks.words <- Socbooks.tidy.stop %>% # we will need to store a separate object with the counts
  count(author, word, sort = TRUE) # including all words and the number of times each author have used them

Socbooks.word.count <- Socbooks.words %>% # also a separate object with the total words for each work
  group_by(author) %>% # first group by author
  summarize(total = sum(n)) # second summarize the total word could

Socbooks.word.count <- left_join(Socbooks.word.count, Socbooks.words) # finally join the words by author with the total
   words of each author
```

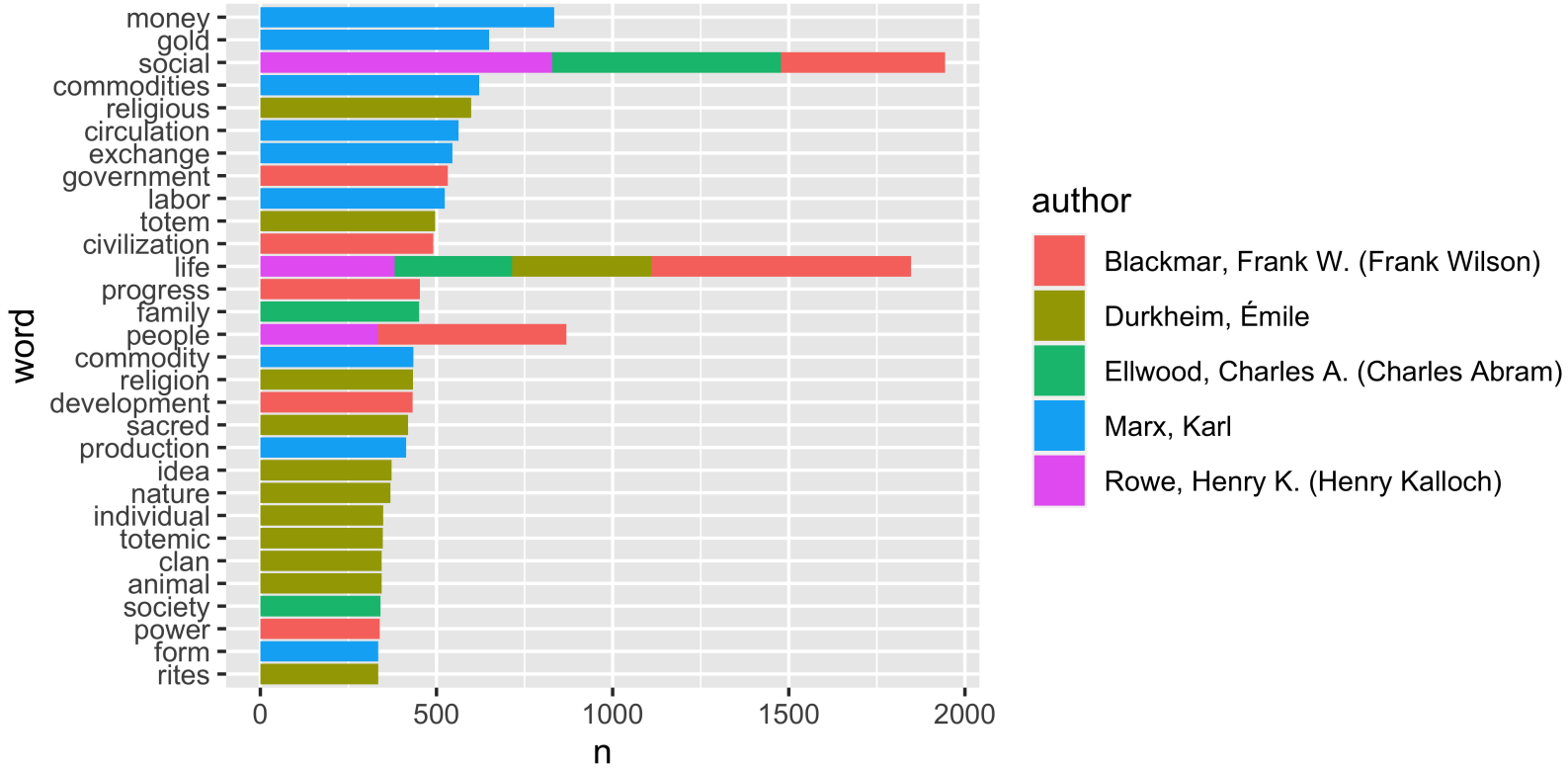
```
## # A tibble: 46,818 × 4
##    author                              total word            n
##    <chr>                               <int> <chr>       <int>
##  1 Blackmar, Frank W. (Frank Wilson) 71571 life            737
##  2 Blackmar, Frank W. (Frank Wilson) 71571 people          536
##  3 Blackmar, Frank W. (Frank Wilson) 71571 government      532
##  4 Blackmar, Frank W. (Frank Wilson) 71571 civilization    491
##  5 Blackmar, Frank W. (Frank Wilson) 71571 social          466
##  6 Blackmar, Frank W. (Frank Wilson) 71571 progress        453
##  7 Blackmar, Frank W. (Frank Wilson) 71571 development     432
##  8 Blackmar, Frank W. (Frank Wilson) 71571 power           339
##  9 Blackmar, Frank W. (Frank Wilson) 71571 time            295
## 10 Blackmar, Frank W. (Frank Wilson) 71571 human           275
## # … with 46,808 more rows
```

Viz word occurences per book (1/2)

Let us visualize the scores with the help of the ggplot2 package.

```
Socbooks.word.count %>%
   filter (n >= 333) %>% # filter so that only the 333 most used words are shown
   mutate(word = reorder(word, n)) %>% # reorder by word after the highest count "n"
   ggplot(aes(x = word, y = n, fill = author)) + # plot with the word in x, count in y and fill by author
   geom_col() + # bar chart where the heights of the bars represent the data values
   coord_flip() # since y is conditional on x, we can flip them for the sake of interpretation
```

Viz word occurences per book (2/2)



Comparing books via tf-idf

To get a better idea of which words are the most distinct for each book in relation to the overall corpus, term frequency-inverse document frequency always comes in handy.

```
Socbooks.tfidf <- Socbooks.tidy.stop %>% # new object storing the tf-idf statistics
    count(author, word, sort = TRUE) %>% # count words by author and sort the results
    bind_tf_idf(word, author, n) %>% # this tidytext function runs the tf-idf by word, author and counts (n)
    arrange(-tf_idf) %>% # orders the rows of by the values of the tf-idf score
    group_by(author) %>% # group the results by author
    top_n(15) %>% # keep only the top 15 words for each document
    ungroup() # ungroup the author grouping
```

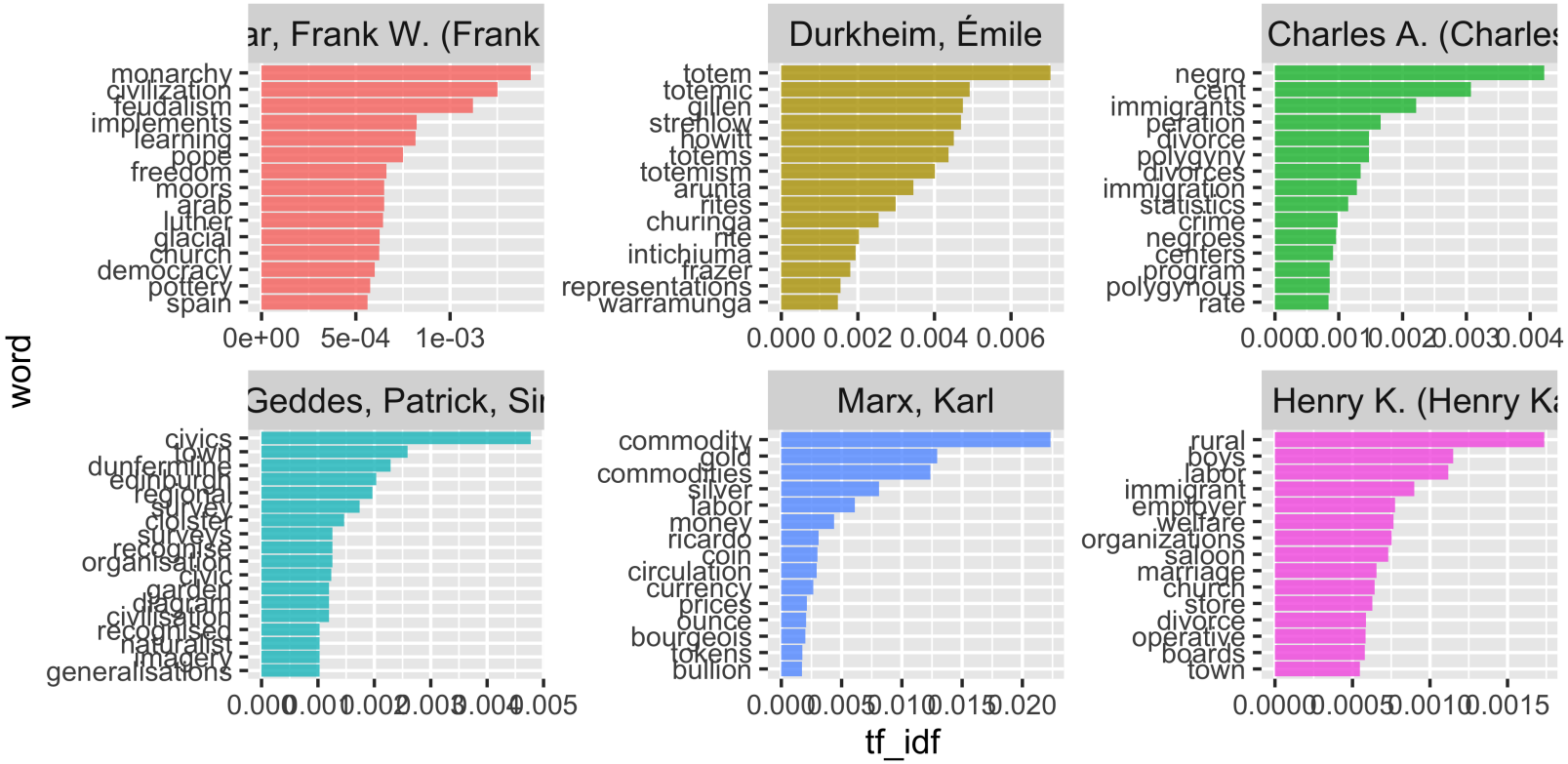
```
## # A tibble: 93 × 6
##    author               word            n      tf   idf  tf_idf
##    <chr>                <chr>       <int>   <dbl> <dbl>   <dbl>
##  1 Marx, Karl           commodity     434 0.0125  1.79  0.0223
##  2 Marx, Karl           gold          649 0.0186  0.693 0.0129
##  3 Marx, Karl           commodities   621 0.0178  0.693 0.0124
##  4 Marx, Karl           silver        257 0.00738 1.10  0.00811
##  5 Durkheim, Émile      totem         496 0.00640 1.10  0.00703
##  6 Marx, Karl           labor         523 0.0150  0.405 0.00609
##  7 Durkheim, Émile      totemic       347 0.00448 1.10  0.00492
##  8 Geddes, Patrick, Sir civics         68 0.00434 1.10  0.00477
##  9 Durkheim, Émile      gillen        205 0.00265 1.79  0.00474
## 10 Durkheim, Émile      strehlow      203 0.00262 1.79  0.00469
## # … with 83 more rows
```

Viz books via tf-idf (1/2)

To facilitate our ability to interpret the results, it is a good idea to visualize the tf-idf scores.

```
Socbooks.tfidf %>%
    mutate(word = reorder_within(word, tf_idf, author)) %>% # mutate and reorder by the words and their tf-idf scores
   by author
    ggplot(aes(word, tf_idf, fill = author)) + # the words as x and tf-idf socre as y, fill by author
    geom_col(alpha = 0.8, show.legend = FALSE) + # a bar chart; let's set the opacity (alpha) to 80% and drop the
   legend since we do not need it
    facet_wrap(~ author, scales = "free", ncol = 3) + # to fit the plot within the screen space, we wrap by author in
   three free scaled columns
    scale_x_reordered() + # we order the x axis along the facets defined in the facet_wrap
    coord_flip() + # again, flip x and y for the sake of interpretation
    theme(strip.text=element_text(size=11)) # add a nicer sized font
```

Viz books via tf-idf (2/2)



Calculating word correlations

To get some insights into how the most distinct words in the corpus are interrelated, we can calculate word correlations. Correlations were covered earlier in the course by Anna Soloveva.

```
library(widyr) # load package for computing correlations, co-occurrences etc on tidy data
Socbooks.word.corr <- Socbooks.word.count %>% # reuse the object for author word counts
  group_by(word) %>% # group by the variable "word"
  filter(n >= 250) %>% # only keep words occurring at least 250 times
  pairwise_cor(word, author, method = "pearson", sort = TRUE, upper = FALSE) # calculating the Pearson correlation
   coefficient; sort; avoid duplicates
```

```
## # A tibble: 861 × 3
##    item1        item2        correlation
##    <chr>        <chr>              <dbl>
##  1 government   civilization           1
##  2 life         social                 1
##  3 government   progress               1
##  4 civilization progress               1
##  5 government   development            1
##  6 civilization development            1
##  7 progress     development            1
##  8 government   power                  1
##  9 civilization power                  1
## 10 progress     power                  1
## # … with 851 more rows
```

Generate a word correlation graph (1/3)

Now let us try to visualize the results with a word graph. The first step is to turn our word correlations into edges.

```
library(igraph) # load package for constructing graphs
set.seed(1234) # pick a seed number to get the same results for randomization
Socbooks.word.corr.graph <- Socbooks.word.corr %>%
  filter(correlation > .3) %>% # filter to avoid overflowing the graph with insignificant words
  graph_from_data_frame() # turning our word correlations into a graph
```

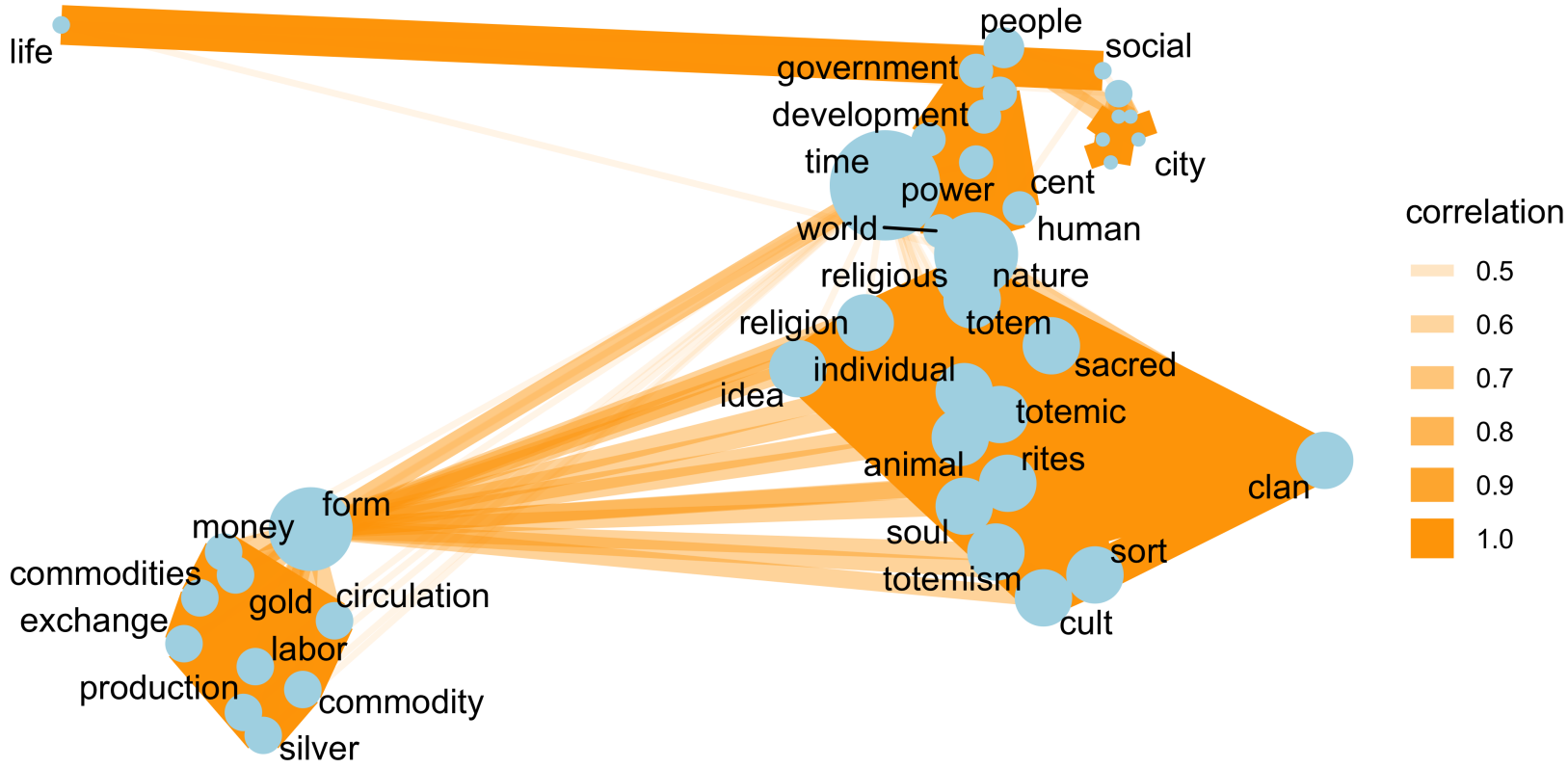
```
## IGRAPH 53fcc17 DN-- 42 249 --
## + attr: name (v/c), correlation (e/n)
## + edges from 53fcc17 (vertex names):
##  [1] government  ->civilization life        ->social
##  [3] government  ->progress     civilization->progress
##  [5] government  ->development  civilization->development
##  [7] progress    ->development  government  ->power
##  [9] civilization->power        progress    ->power
## [11] development ->power        government  ->human
## [13] civilization->human        progress    ->human
## [15] development ->human        power       ->human
## + ... omitted several edges
```

Generate a word correlation graph (2/3)

The second step is to create a nice visualization based on the word correlation graph that is easy to interpret.

```
library(ggraph) # an extension of ggplot2 aimed at supporting relational data structures
Socbooks.word.corr.graph %>%
  ggraph() + # function for generating a graph; you can try several different layouts with argument layout = "fr" [alt.
   "kk" or "drl"...)
  geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation), edge_colour = "orange") + # the edges/links
   are taken from the correlation variable; choose a cool color
  geom_node_point(size = 0.5*igraph::degree(Socbooks.word.corr.graph), colour = "lightblue") + # size the nodes/points
   based on half the number of its adjacent edges; color it
  geom_node_text(aes(label = name), repel = TRUE) + # display the word associated with each node/point; repel to enable
   interpretation
  theme_void() # add a haunting theme to let the graph elevate
```

Generate a word correlation graph (3/3)



Preparing for topic modeling

You might also be interested in modeling the most prevalent topics that run through your corpus, which can be made with topic modeling. One way to define topic modeling is to say that it is able to discover abstract topics within a corpus with the help of a probabilistic unspervised machine learning model. Step one for topic modeling is to create a document-term matrix (dtm).

```
Socbooks.dtm <- Socbooks.word.count %>% # we reuse the word counts in a dtm
  cast_dtm(author, word, n) # tidytext provides several fine cast functions to enable fast conversions
```

```
## <<DocumentTermMatrix (documents: 6, terms: 23308)>>
## Non-/sparse entries: 46818/93030
## Sparsity           : 67%
## Maximal term length: 23
## Weighting          : term frequency (tf)
```

Run a topic model

Now we are ready for topic modeling. This time we will go for the Latent Dirichlet Allocation (LDA), a classic and still most popular topic model.

```
library(topicmodels) # a package for LDA and correlated topic models (CTM) by its creators
Socbooks.lda <- LDA(Socbooks.dtm, # function for generating a LDA
                    k = 4, # in topic modeling you have to select the number of topics (k); this can be tricky but for
   now let's go for 4
                    control = list(seed = 1234)) # again, to get the same results for randomization
```

```
## A LDA_VEM topic model with 4 topics.
```

Inspecting the beta

In a topic model, the beta tells us how probable it for words to show up in a topic.

```
Socbooks.lda.beta <- tidy(Socbooks.lda, matrix = "beta") # the tidy function can easily generate a beta matrix
```

```
## # A tibble: 93,232 × 3
##    topic term            beta
##    <int> <chr>          <dbl>
##  1     1 life        0.000632
##  2     2 life        0.00514
##  3     3 life        0.0102
##  4     4 life        0.00723
##  5     1 people      0.000488
##  6     2 people      0.000607
##  7     3 people      0.00610
##  8     4 people      0.00544
##  9     1 government  0.000432
## 10     2 government  0.0000356
## # … with 93,222 more rows
```

Top terms in topic model beta

To interpret the topics, we are interested in the words with the highest probability for each topic.

```
Socbooks.lda.beta.top10 <- Socbooks.lda.beta %>%
  group_by(topic) %>% # we group by the 4 topics
  slice_max(beta, n = 10) %>% # for each we keep the 10 words with the largest beta value
  ungroup() %>% # ungroup the grouping
  arrange(topic, -beta) # arrange the scores by topic and beta
```

```
## # A tibble: 40 × 3
##    topic term           beta
##    <int> <chr>         <dbl>
##  1     1 money       0.0236
##  2     1 gold        0.0184
##  3     1 commodities 0.0176
##  4     1 circulation 0.0160
##  5     1 exchange    0.0157
##  6     1 labor       0.0148
##  7     1 commodity   0.0123
##  8     1 production  0.0118
##  9     1 form        0.00962
## 10     1 time        0.00942
## # … with 30 more rows
```

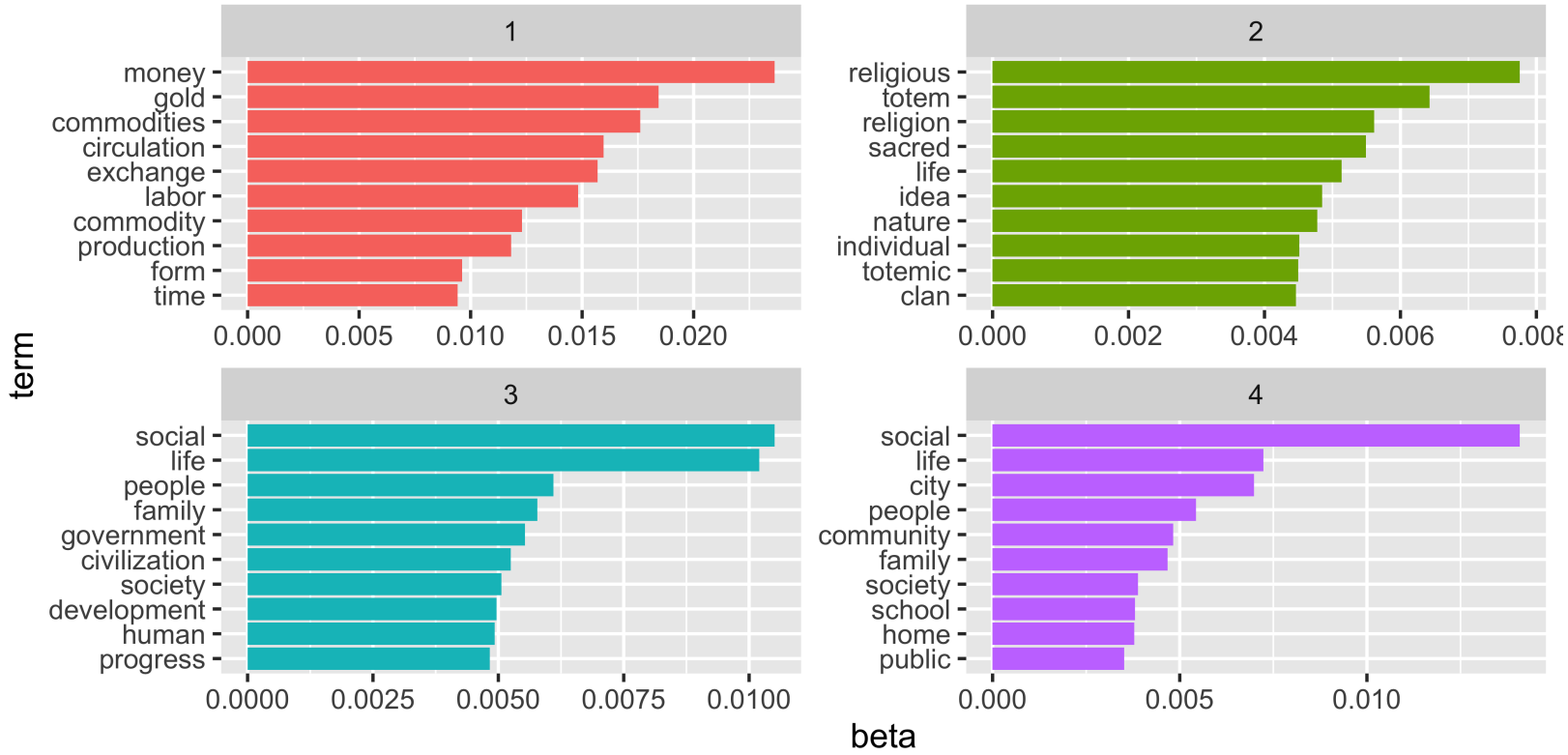
Viz topic model beta (1/2)

Let us visualize the top words, topic by topic.

```
library(ggplot2)

Socbooks.lda.beta.top10 %>%
  mutate(term = reorder_within(term, beta, topic)) %>% # order terms after their beta values in each topic
  ggplot(aes(beta, term, fill = factor(topic))) + # beta values as x, terms as y and fill by the topics as factors
  geom_col(show.legend = FALSE) + # bar chart without legend
  facet_wrap(~ topic, scales = "free") + # free scale wrap after topic
  scale_y_reordered() # order after terms
```

Viz topic model beta (2/2)



Inspecting the gamma

In a topic model, the gamma tells us how probable it for topics to show up in a document.

```
Socbooks.lda.gamma <- tidy(Socbooks.lda, matrix = "gamma") # the tidy function can easily generate a gamma matrix
```

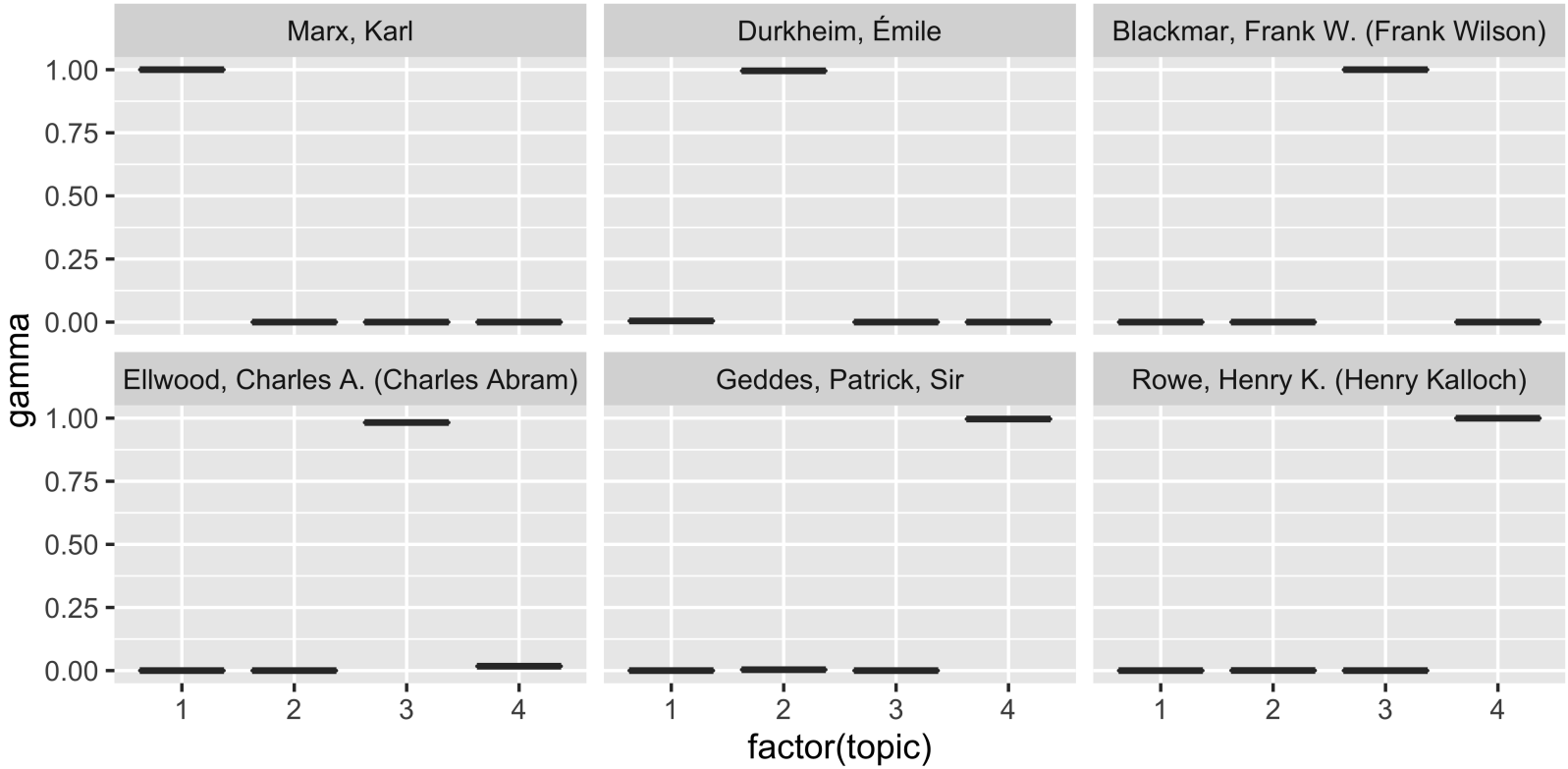
```
## # A tibble: 24 × 3
##    document                              topic       gamma
##    <chr>                                 <int>       <dbl>
##  1 Blackmar, Frank W. (Frank Wilson)         1 0.000000468
##  2 Durkheim, Émile                           1 0.00458
##  3 Ellwood, Charles A. (Charles Abram)       1 0.000000998
##  4 Geddes, Patrick, Sir                      1 0.00000214
##  5 Marx, Karl                                1 1.00
##  6 Rowe, Henry K. (Henry Kalloch)            1 0.000000637
##  7 Blackmar, Frank W. (Frank Wilson)         2 0.000000468
##  8 Durkheim, Émile                           2 0.995
##  9 Ellwood, Charles A. (Charles Abram)       2 0.000000998
## 10 Geddes, Patrick, Sir                      2 0.00351
## # … with 14 more rows
```

Viz topic model gamma (1/2)

Let us visualize how the topics and the books relate to one another.

```
Socbooks.lda.gamma %>%
  mutate(document = reorder(document, gamma * topic)) %>% # order the books after their gamma values vis-a-vis each
   topic
  ggplot(aes(factor(topic), gamma)) + # factor topics
  geom_boxplot() + # choose box plot for easy comparison
  facet_wrap(~ document)
```

Viz topic model gamma (2/2)



Adding sentiments to your analysis

Lastly, you might be interested in a more supervised approach, such as investigating the emotional architecture of your documents. This leads us to sentiment analysis.

```
bing <- get_sentiments("bing") # tidytext provides you with sentiment lexicons like the famous Bing

Socbooks.senti.count <- Socbooks.tidy.stop %>% # go back to our tidy corpus
  inner_join(bing) %>% # connect it with the word lexicon and its positive/negative sentiment system
  count(word, sentiment, sort = TRUE) %>% # count words by their sentiment and sort
```

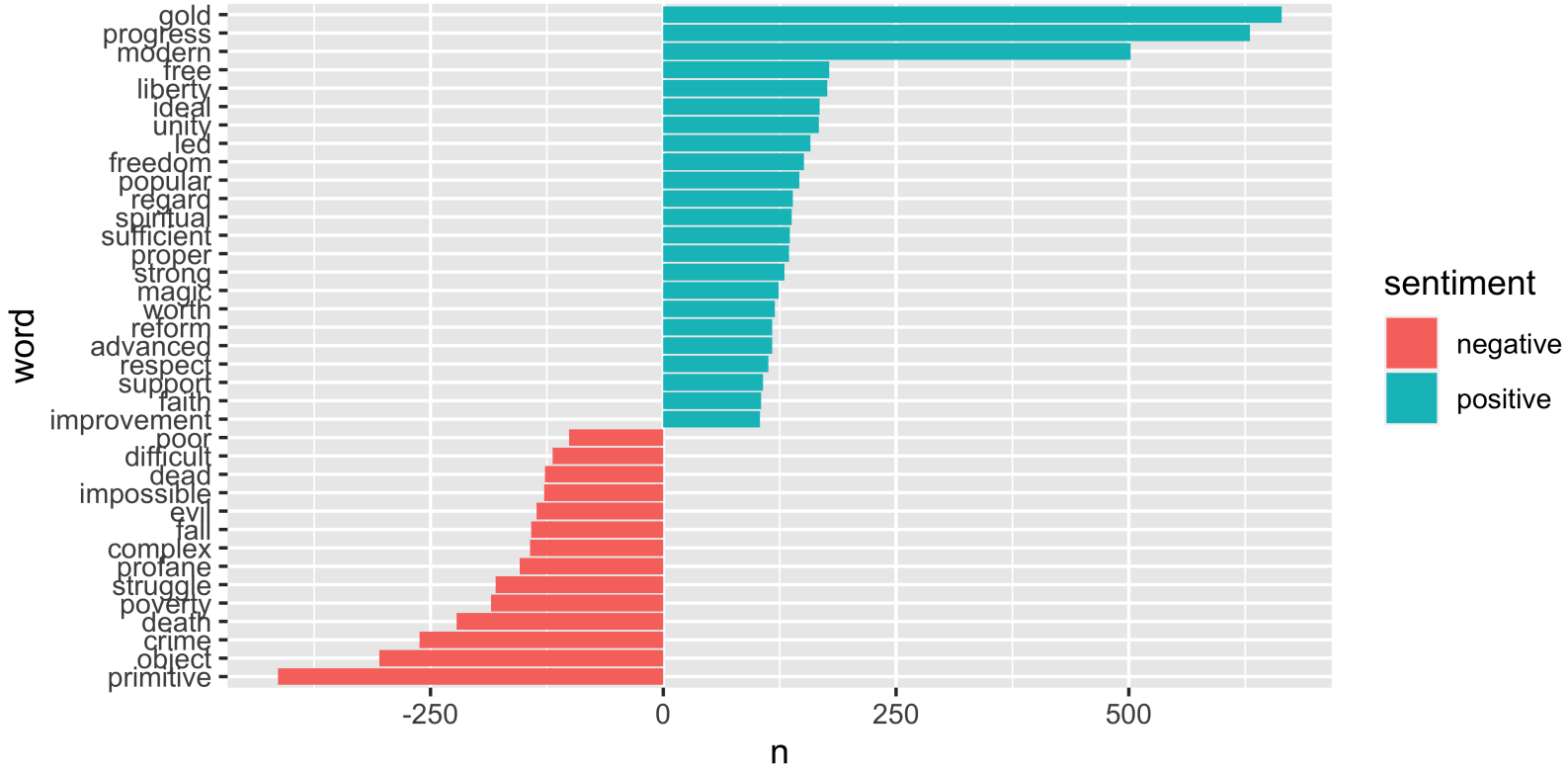
```
## # A tibble: 2,773 × 3
##    word      sentiment     n
##    <chr>     <chr>     <int>
##  1 gold      positive    664
##  2 progress  positive    630
##  3 modern    positive    502
##  4 primitive negative    414
##  5 object    negative    305
##  6 crime     negative    262
##  7 death     negative    222
##  8 poverty   negative    185
##  9 struggle  negative    180
## 10 free      positive    178
## # … with 2,763 more rows
```

Viz top sentiments (1/2)

We can now generate a visualization to get a feeling for the overall distribution of sentimental words over the corpus.

```
Socbooks.senti.count %>%
  filter(n > 100) %>% # keep only the most prevalent words
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) + # plot the words and color them by sentiment
  geom_col() + # go for a bar chart
  coord_flip() # flip the axes
```

Viz top sentiments (2/2)



The sentiment narrative unfolds (1/3)

As often is the case, we might not only want to compare how emotional each document is but also how it develops over the pages. This is possible.

```
Socbooks.senti.comp <- Socbooks.tidy.stop %>%
  inner_join(bing) %>% # connect the tidy corpus with the sentiment lexicon
  count(author, index = line %/% 100, sentiment) %>% # count sentiment by author and create an index based on the book
   lines
  spread(sentiment, n, fill = 0) %>% # increasing the number of columns and decreasing the number of rows
  mutate(sentiment = positive - negative) # add a new sentiment variable
```

```
## # A tibble: 771 × 5
##    author                           index negative positive sentiment
##    <chr>                            <dbl>    <dbl>    <dbl>     <dbl>
##  1 Blackmar, Frank W. (Frank Wilson)   268        0        1         1
##  2 Blackmar, Frank W. (Frank Wilson)   269        7       31        24
##  3 Blackmar, Frank W. (Frank Wilson)   270       16       20         4
##  4 Blackmar, Frank W. (Frank Wilson)   271       11       13         2
##  5 Blackmar, Frank W. (Frank Wilson)   272        7       37        30
##  6 Blackmar, Frank W. (Frank Wilson)   273       14       20         6
##  7 Blackmar, Frank W. (Frank Wilson)   274       19       35        16
##  8 Blackmar, Frank W. (Frank Wilson)   275       25       42        17
##  9 Blackmar, Frank W. (Frank Wilson)   276        8       31        23
## 10 Blackmar, Frank W. (Frank Wilson)   277       19       51        32
## # … with 761 more rows
```

The sentiment narrative unfolds (2/3)

So, let us visualize the comparison for all books.

```
Socbooks.senti.comp %>%
  ggplot(aes(index, sentiment, fill = author)) + # we use our index as x and sentiment as y and color by author
  geom_bar(stat = "identity", show.legend = FALSE) + # bar chart with an identified y value ("identity") without the
   legend
  facet_wrap(~author, ncol = 2, scales = "free_x") # we wrap by author in two free scaled columns
```

The sentiment narrative unfolds (3/3)

Thank you for your time!

Do not hesitate to contact me


josef.ginnerskov@soc.uu.se


@doeparen


@doeparen