Text Mining in R

Sara Weston and Debbie Yee

03/24/2017

What is Text Mining?

- Text mining is the process by which textual data is broken down into pieces for analysis.
- Pieces can be words or phrases.
- Pieces can be analyzed as they are or as the sentiment they represent.
- Text mining can be used to test hypotheses or gain descriptive insight into data.

Necessary packages

```
library(tm) #for reading in text documents
library(tidytext) # for cleaning text and sentiments
library(topicmodels) # for topic analysis
library(janeaustenr) # for free data
library(dplyr) # for data manipulation
library(tidyr)
library(stringr) # for manipulating string/text data
library(ggplot2) # for pretty graphs
library(wordcloud) #duh
```

Load in fun data

We're using data from the janeaustenr package, which includes all six of Jane Austen's novels.

- Require some preprocessing.
- We restructure the data so that each chapter is it's own "observation", with data on which book and which chapter it comes from.
- Code is included in the Rmd file, but not shown here.

```
## # A tibble: 6 × 3
##                   book chapter
##                 <fctr>   <int>
## 1 Sense & Sensibility       0
## 2 Sense & Sensibility       1
## 3 Sense & Sensibility       2
## 4 Sense & Sensibility       3
## 5 Sense & Sensibility       4
## 6 Sense & Sensibility       5
## # ... with 1 more variables: text <chr>
```

Types of information

We can analyze text data in a lot of ways. Today we will talk about three kinds of ways to measure or 'code' text data:

- Word frequenices
- Word sentiment
- Topics, based on word clusters

Each of these ways of measuring data require that data be restructured in different ways.

- Make use of the dplyr and tidyr packages

Types of data structures

- Long form
  - Each word gets its own row in a data frame.
  - Sometimes each word in each document (person).
  - Columns contain information about the word (and document).
- Short form
  - Each document (person) gets its own row.
  - Columns contain information about the documents, plus there is one column for every unique word in the corpus.

Long form

It's easy to get the data into a form where each word gets its own row.

```
long_austen <- original_books %>%
  unnest_tokens(output = word,
                input = text,
                token = "words")
head(long_austen)
```

```
## # A tibble: 6 × 3
##                   book chapter         word
##                 <fctr>   <int>        <chr>
## 1 Sense & Sensibility       0        sense
## 2 Sense & Sensibility       0          and
## 3 Sense & Sensibility       0 sensibility
## 4 Sense & Sensibility       0           by
## 5 Sense & Sensibility       0         jane
## 6 Sense & Sensibility       0       austen
```

A note about stop words.

Stop words are words in the English language that connect other words, but often have little or no content.

Examples: - Conjunctions - Articles - Prepositions

You will likely want to remove these words before you proceed.

Remove stop words

```
head(stop_words)
```

```
## # A tibble: 6 × 2
##         word lexicon
##        <chr>   <chr>
## 1          a   SMART
## 2        a's   SMART
## 3       able   SMART
## 4      about   SMART
## 5      above   SMART
## 6 according   SMART
```

```
long_austen <- long_austen %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

Short form

And you can use the long-form version to create the short form.

```
short_austen <- long_austen %>%
  mutate(bookchap = paste(book, chapter, sep="_")) %>%
  select(-c(book, chapter)) %>%
  group_by(bookchap) %>%
  count(word) %>%
  cast_dtm(document = bookchap, term = word, value = n)
```

Back to information

We can use these two data structures to calculate frequenices, match sentiments and estimate topics.

First, frequencies.

Frequiences

```
long_austen %>%
  count(word)
```

```
## # A tibble: 13,816 × 2
##             word     n
##            <chr> <int>
## 1             _a_     2
## 2    _accepted_     1
## 3    _accident_     1
## 4       _adair_     1
## 5    _addition_     1
## 6  _advantages_     1
## 7       _affect_     1
## 8     _against_     1
## 9   _agreeable_     1
## 10         _air_     1
## # ... with 13,806 more rows
```

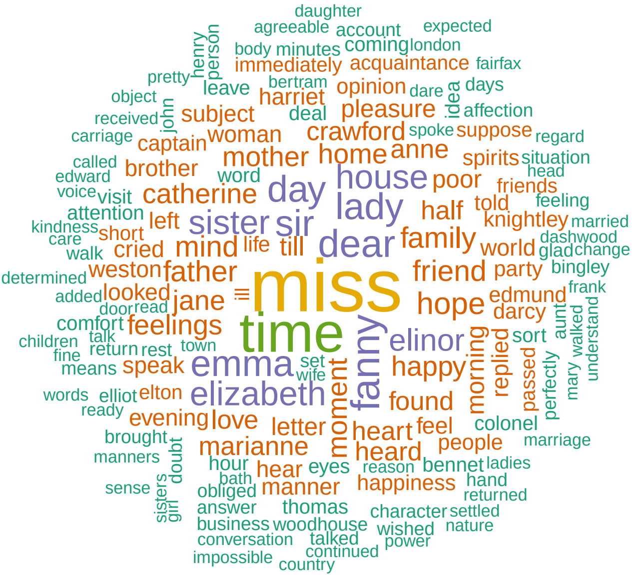
What can you do with frequencies?

- Summarize your data
- Estimate relationships between single words and other variables

Wordcloud

```
austen_freq <- long_austen %>%
  count(word)
wordcloud(words = austen_freq$word, freq = austen_freq$n, #
          min.freq = 200, # fun arguments
          random.order = FALSE,
          random.color = F,
          colors=brewer.pal(6, "Dark2"))
```

Wordcloud


daughter
agreeable
account
expected
henry
person
body
minutes
coming
london
immediately
acquaintance
fairfax
pretty
leave
bertram
opinion
dare
idea
days
object
harriet
john
received
subject
deal
pleasure
affection
carriage
captain
woman
crawford
spoke
suppose
regard
called
mother
home
anne
spirits
situation
edward
brother
word
house
poor
head
voice
catherine
day
friends
visit
lady
told
feeling
attention
half
kindness
left
sister
sir
knightley
married
care
short
dear
family
dashwood
walk
cried
mind
life
till
world
glad
change
determined
weston
father
miss
friend
party
bingley
added
looked
jane
ii
hope
edmund
frank
door
read
darcy
comfort
feelings
time
aunt
walked
understand
children
talk
fanny
elinor
sort
fine
return
rest
town
morning
replied
means
speak
emma
set
happy
passed
perfectly
mary
wife
moment
words
elliot
elton
elizabeth
found
ready
evening
love
feel
colonel
brought
letter
heart
marianne
people
marriage
manners
doubt
heard
hour
reason
bennet
ladies
hear
eyes
bath
hand
sense
sisters
obliged
manner
happiness
returned
girl
answer
thomas
character
settled
business
woodhouse
wished
nature
conversation
talked
power
continued
impossible
country

Estimate relationships

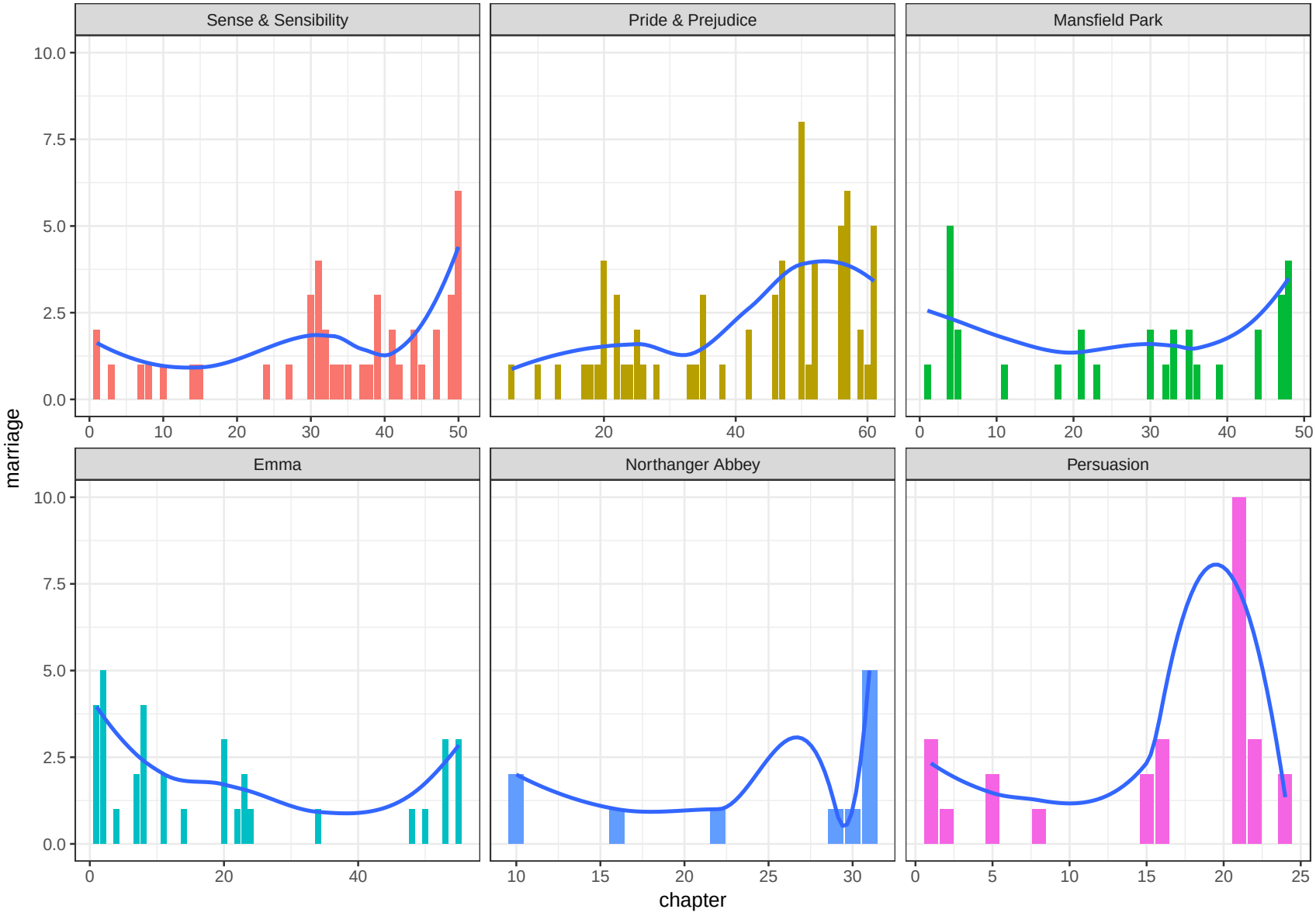
```
short_freq <- long_austen %>%
  group_by(book, chapter) %>%
  count(word) %>%
  spread(key = word, value = n)
short_freq
```

```
## Source: local data frame [275 x 13,818]
## Groups: book, chapter [275]
##
##                  book chapter `_a_` `_accepted_` `_acc
## *                <fctr>   <int> <int>         <int>
## 1  Sense & Sensibility       0    NA            NA
## 2  Sense & Sensibility       1    NA            NA
## 3  Sense & Sensibility       2    NA            NA
## 4  Sense & Sensibility       3    NA            NA
## 5  Sense & Sensibility       4    NA            NA
## 6  Sense & Sensibility       5    NA            NA
## 7  Sense & Sensibility       6    NA            NA
```

Estimate relationships

```
ggplot(short_freq, aes(x = chapter,
                       y = family,
                       fill = book)) +
  geom_bar(stat = "identity") +
  geom_smooth(se=F)+
  guides(fill=F) +
  facet_wrap(~book, scales = "free_x") +
  theme_bw()
```

Estimate relationships



Sentiment

Words have sentimental value. There are three ways you can operationalize the sentimental value of a word.

- Positive or negative
- Numeric (-3 to 3)
- Emotion (joy, fear, trust, disgust, etc)

Use the get_sentiments function to get the operationalize you want.

Sentiment

```
get_sentiments("bing")
```

```
## # A tibble: 6,788 × 2
##           word sentiment
##          <chr>     <chr>
## 1      2-faced  negative
## 2      2-faces  negative
## 3          a+  positive
## 4     abnormal  negative
## 5      abolish  negative
## 6   abominable  negative
## 7   abominably  negative
## 8    abominate  negative
## 9  abomination  negative
## 10       abort  negative
## # ... with 6,778 more rows
```

Sentiment

```
get_sentiments("afinn")
```

```
## # A tibble: 2,476 × 2
##          word score
##         <chr> <int>
## 1     abandon    -2
## 2   abandoned    -2
## 3    abandons    -2
## 4    abducted    -2
## 5   abduction    -2
## 6  abductions    -2
## 7       abhor    -3
## 8    abhorred    -3
## 9   abhorrent    -3
## 10     abhors    -3
## # ... with 2,466 more rows
```

Sentiment

```
get_sentiments("nrc")
```

```
## # A tibble: 13,901 × 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
```

Attaching sentiments

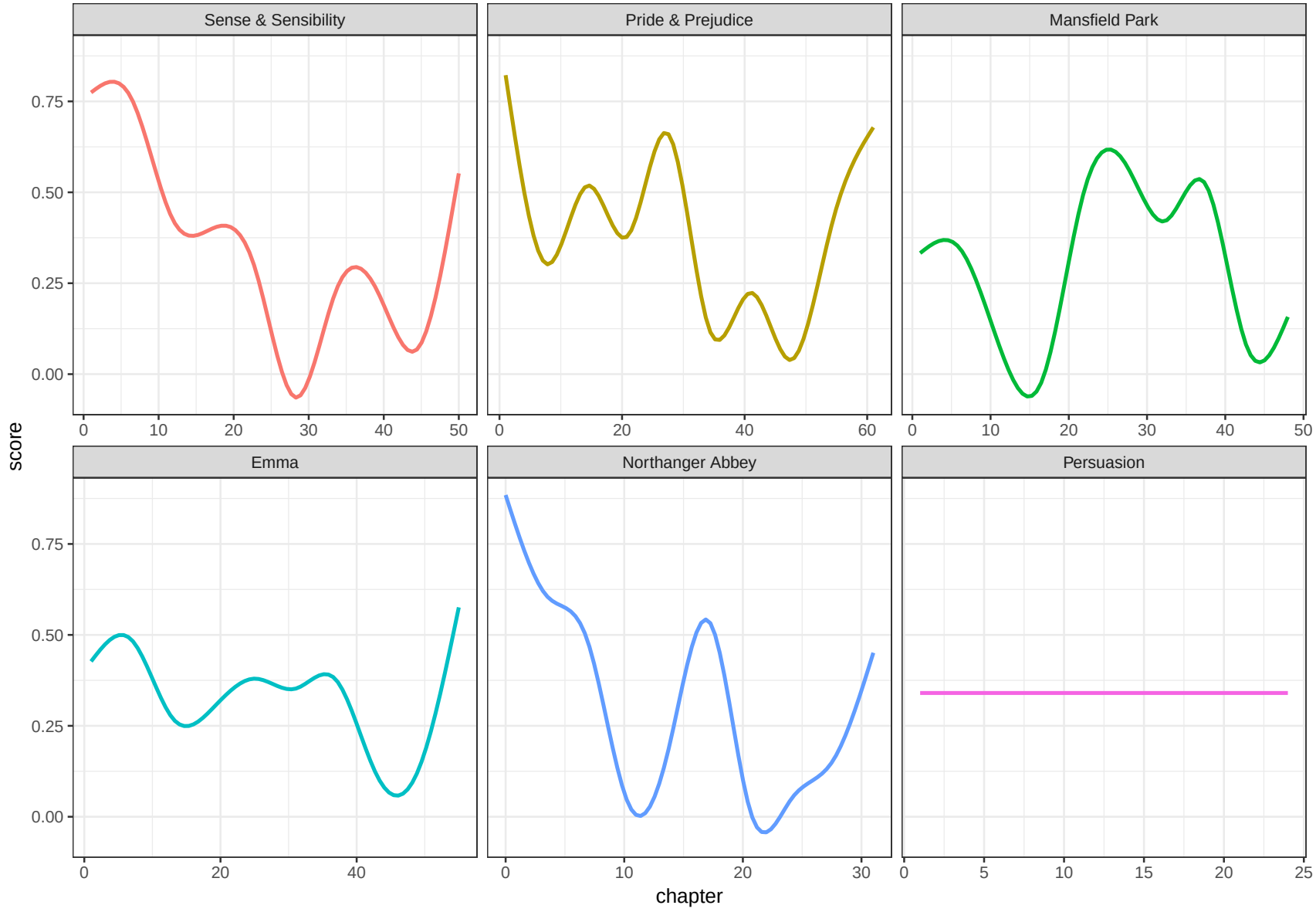
To use these, you can join the sentiments data frame with your long-form word data frame.

```
long_austen <- long_austen %>%
  inner_join(get_sentiments("afinn"))
```

Use sentiments as a new variable

```
ggplot(long_austen, aes(x = chapter, y = score, color = boo
  geom_smooth(se = F) +
  guides(color = FALSE) +
  facet_wrap(~book, scales = "free_x") +
  theme_bw()
```

Use sentiments as a new variable



Topics

You can try to infer what topics are coming up in your text data.

- Does require you to make some guesses.
- Probably more useful for describing data and synthesizing comments and pilot data than for inferential stats.
- Use short-form data set.

Topics

Latent Dirichlet allocation

```
books_lda <- LDA(short_austen, k = 6,
                 control = list(seed = 1234))
```

Topics

- Extract from that the beta matrix.
- In this, each word gets one row for each topic.
- Beta is the probability of that term being generated from that topic.

```
book_topics <- tidy(books_lda, matrix = "beta")
head(book_topics)
```

```
## # A tibble: 6 × 3
##   topic   term           beta
##   <int>  <chr>          <dbl>
## 1     1 austen  1.185131e-04
## 2     2 austen 5.053303e-320
## 3     3 austen  3.720076e-44
## 4     4 austen  1.742293e-95
## 5     5 austen  3.114776e-05
## 6     6 austen  3.114320e-37
```

Find the top terms in each topic

The best way to work with these data is to find the "top terms" in each topic, to try and figure out what the topic might be.

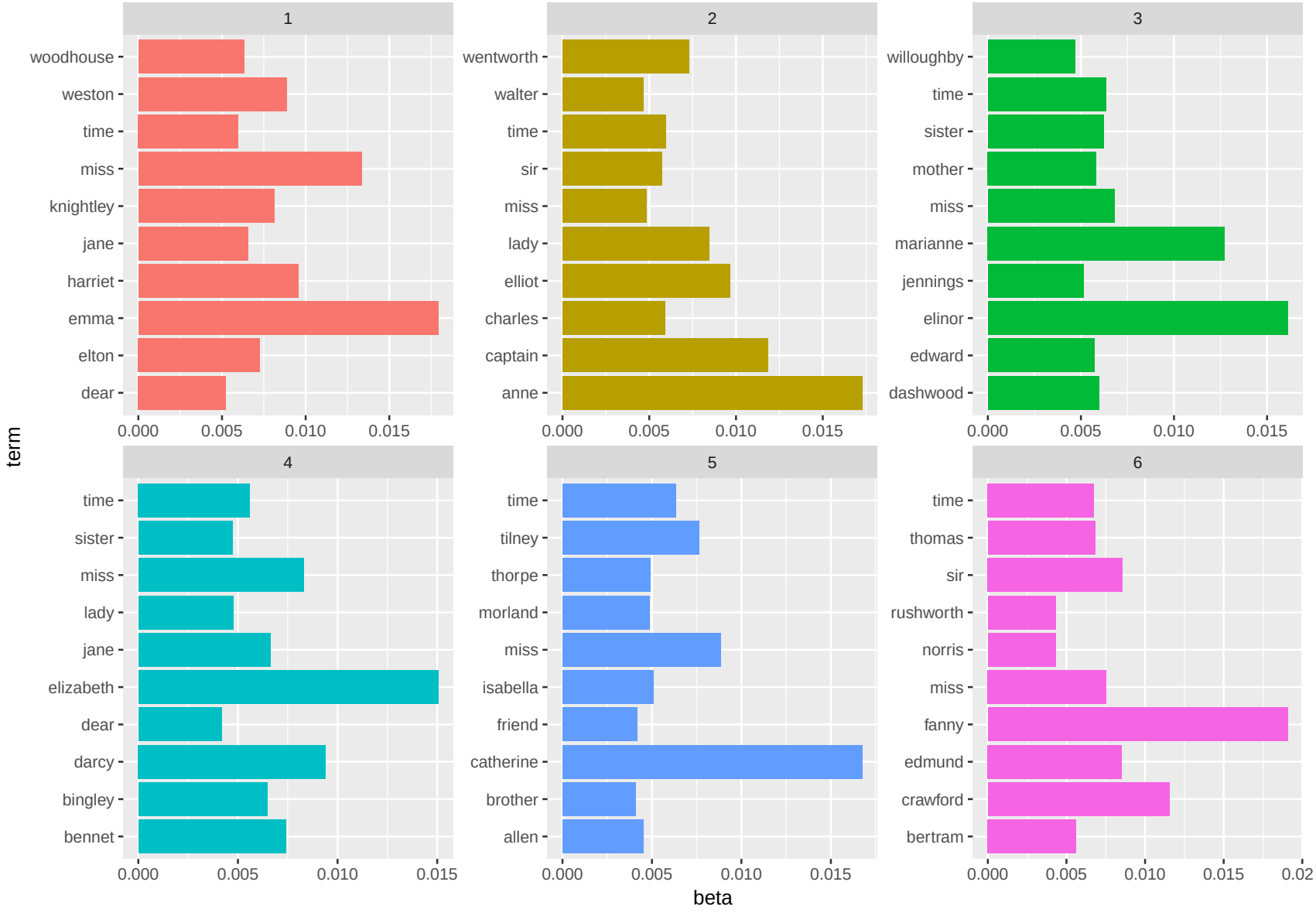
```
top_terms <- book_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

head(top_terms)
```

```
## # A tibble: 6 × 3
##   topic      term        beta
##   <int>     <chr>       <dbl>
## 1     1      emma 0.017916669
## 2     1      miss 0.013341364
## 3     1   harriet 0.009551112
## 4     1    weston 0.008867266
```

Plot top terms

```
ggplot(top_terms, aes(term, beta, fill=factor(topic))) +
  geom_bar(stat="identity", show.legend = F)+
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

Plot top terms



Use in Psychology Research

- Data was generously provided by Alexa Lord.
- Study on self-affirmation - “the act of affirming an important, typically non-threatened, aspect of the self”
- Does self-affirmation reduce rejection sensitivity

Experimental manipulation

- Self-affirmation condition: Rank values or traits on importance. Write for five minutes about the trait or value you listed as most important.
- Control condition: Think of a T.V. character and rank values or traits on importance to that character. Write for five minutes about why that T.V. character values the number one character or trait.

Load in data

```
#typical data
saf <- read.csv("saf.csv")
#text data
docs <- VCorpus(DirSource("tm data/Text Files"))
docs.tidy <- tidy(docs)
docs.tidy$ID <- gsub("\\.txt", "", docs.tidy$id)
```

Tokenize text

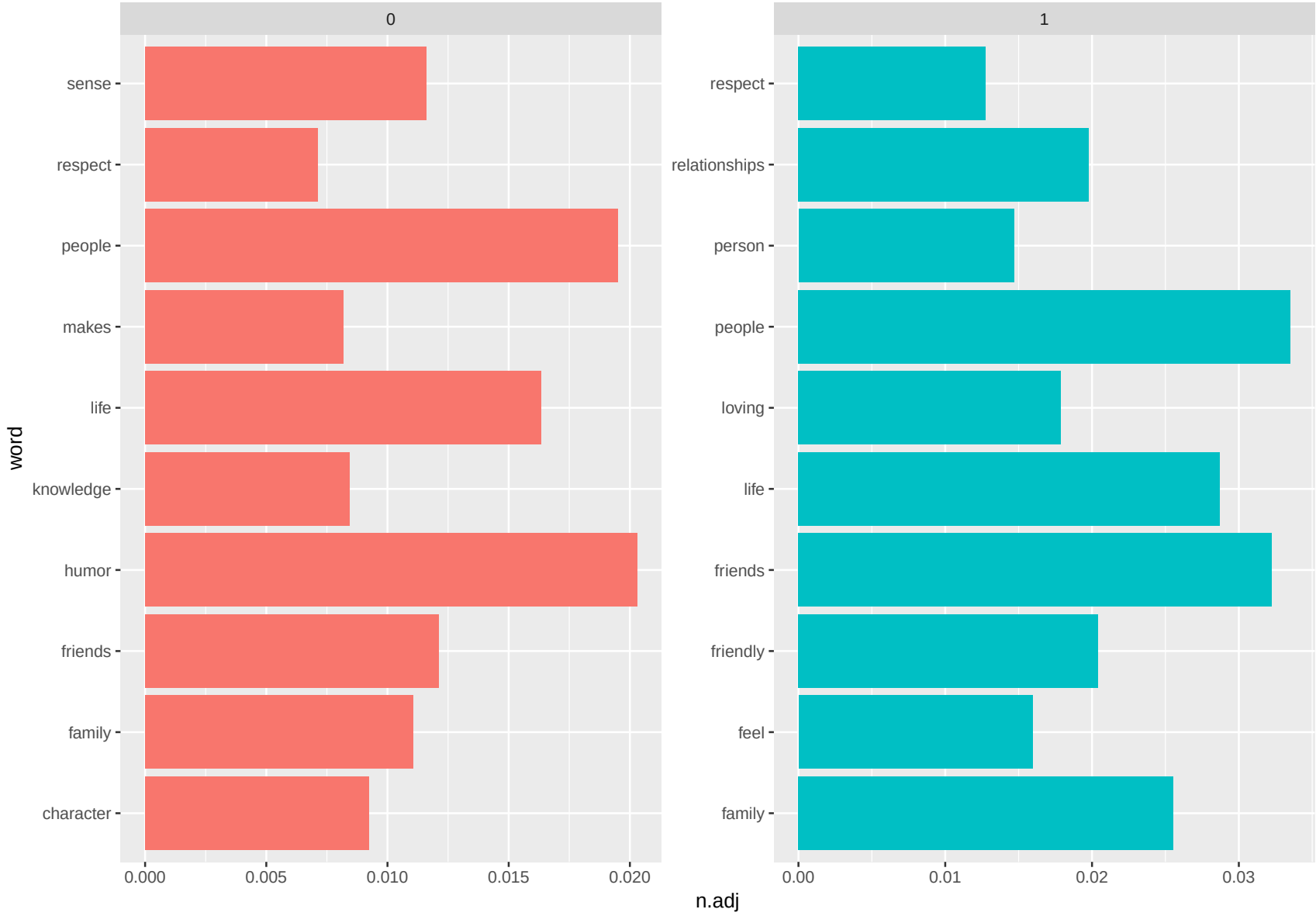
```
docs.tidy2 <-  docs.tidy %>%
  unnest_tokens(output = word,
                input = text,
                token = "words") %>%
  anti_join(stop_words) %>%
  select(ID, word)

saf <- merge(saf, docs.tidy2, by = "ID")
```

Frequencies

Which words are used more frequently in the SAF condition than in the control condition?

```
saf.freq <- saf %>%
  group_by(SAF) %>%
  count(word) %>%
  ungroup()

saf.freq <- saf %>%
  group_by(SAF) %>%
  summarize(n.words = n()) %>%
  ungroup() %>%
  inner_join(saf.freq) %>%
  mutate(n.adj = n/n.words)
```

Plot word frequencies



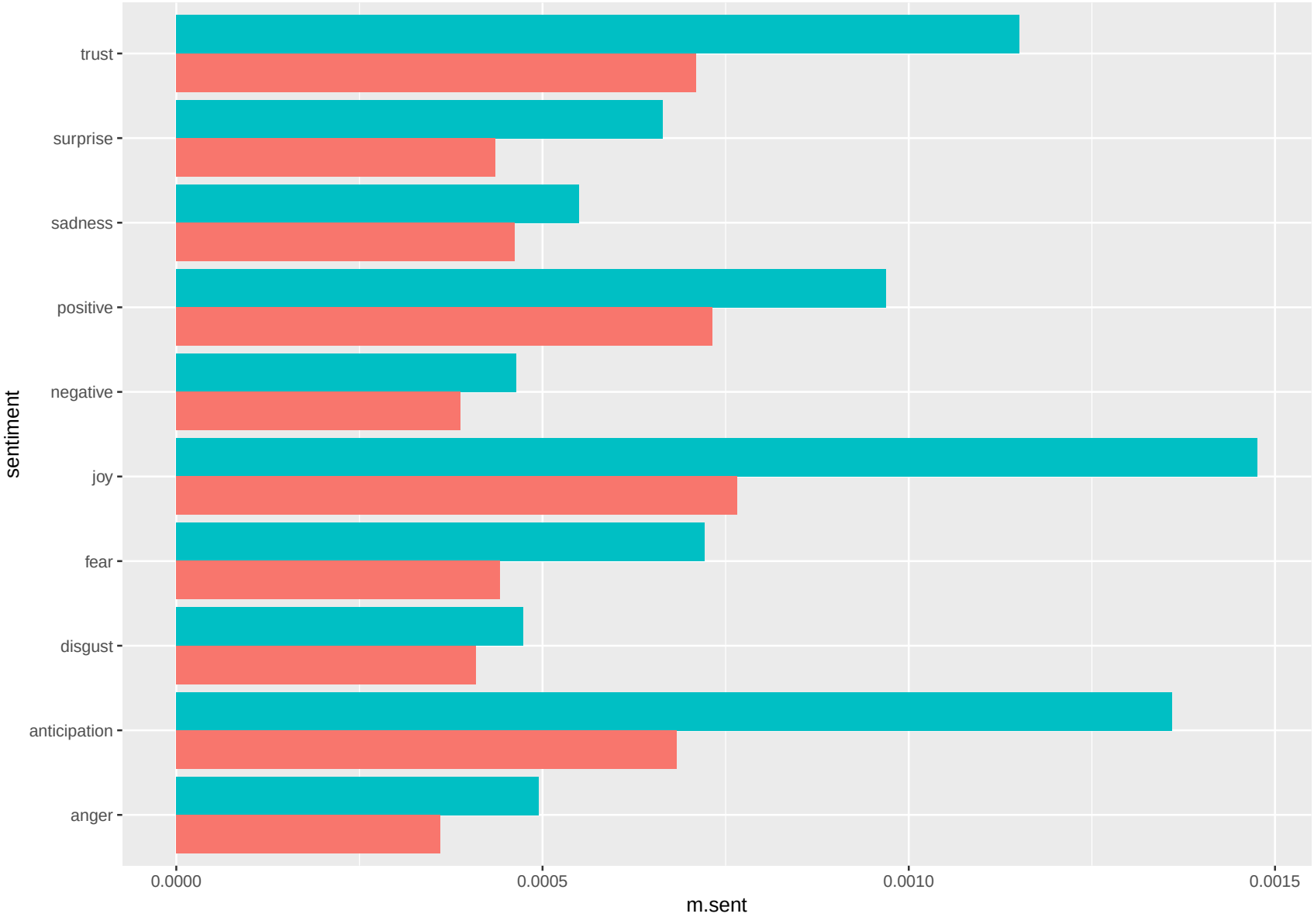
Sentiment

Which sentiments are found in each condition?

```
saf.freq %>%
  inner_join(get_sentiments("nrc")) %>%
  group_by(SAF, sentiment) %>%
  summarize(m.sent = mean(n.adj, na.rm=T))
```

```
## Joining, by = "word"

## Source: local data frame [20 x 3]
## Groups: SAF [?]
##
##      SAF    sentiment       m.sent
##    <int>        <chr>        <dbl>
## 1      0        anger 0.0003602342
## 2      0 anticipation 0.0006829609
## 3      0      disgust 0.0004092731
## 4      0         fear 0.0004422031
## 5      0          joy 0.0007660650
```

Plot sentiments



Topics

What are people talking about?

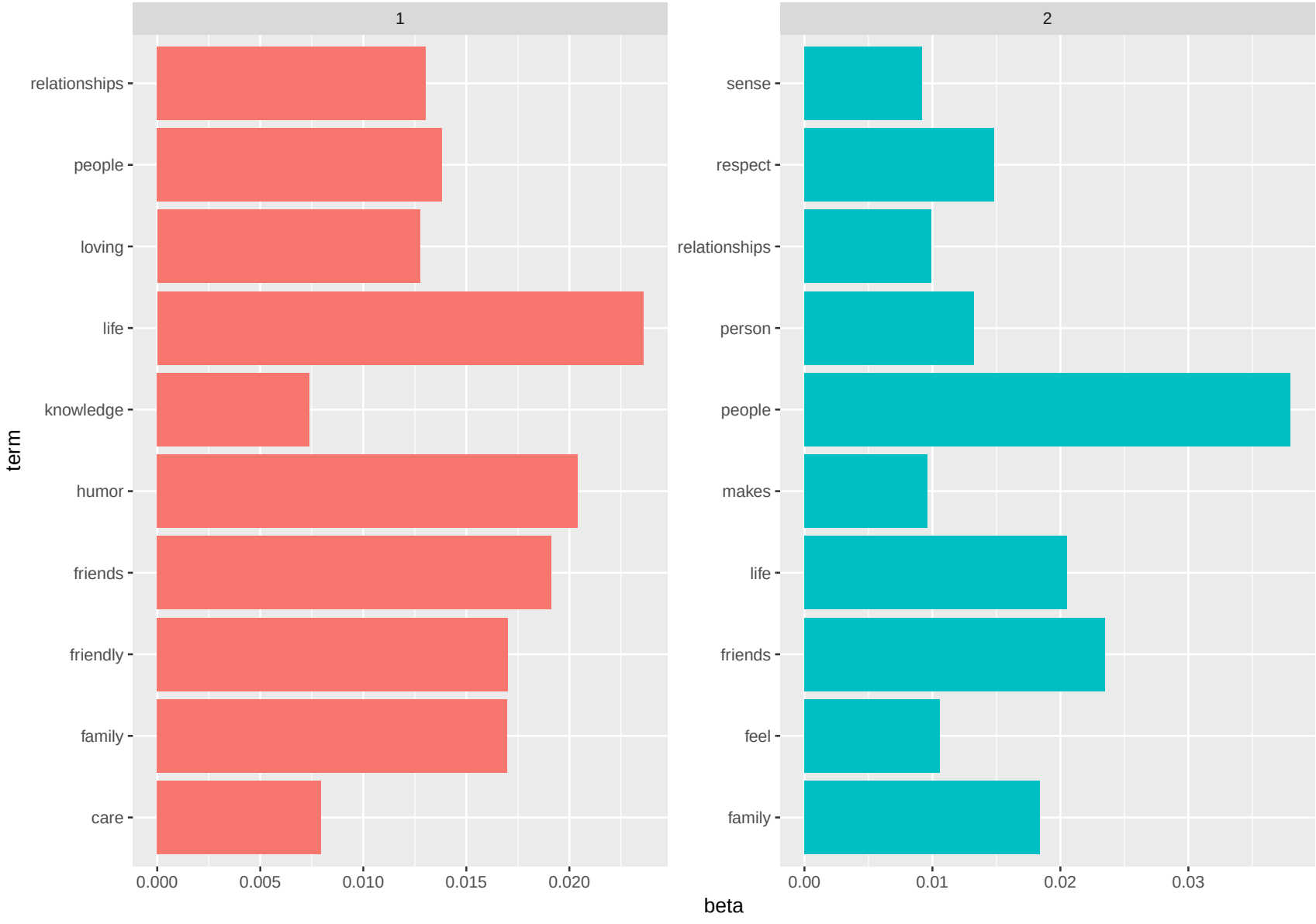
```
short_saf <- saf.freq %>%
  filter(!grepl("[0-9]", word)) %>%
  select(SAF, word, n) %>%
  cast_dtm(document = SAF, term = word, value = n)

saf_lda <- LDA(short_saf, k = 2,
               control = list(seed = 1234))
```

Extract topics

```
saf_topics <- tidy(saf_lda, matrix = "beta")
head(saf_topics)
```

```
## # A tibble: 6 × 3
##   topic      term         beta
##   <int>     <chr>        <dbl>
## 1     1 abilities 4.014336e-04
## 2     2 abilities 7.568858e-04
## 3     1   ability 2.452691e-03
## 4     2   ability 1.609864e-03
## 5     1     abuse 2.138209e-04
## 6     2     abuse 7.667266e-05
```

Plot topics


1
2
relationships
people
loving
life
knowledge
humor
friends
friendly
family
care
sense
respect
relationships
person
people
makes
life
friends
feel
family
term
beta
0.000
0.005
0.010
0.015
0.020
0.00
0.01
0.02
0.03