Chapter 11. Text Mining with R

Reference: Silge, J. and Robinson, D. (2017). Text Minging With R. O'Reilly. Available online at `https://www.tidytextmining.com/`

1. Tidy text format: a table with one token per row.

Token (or *Term*): a word, a phrase, or several connected words.

Other text data structures:

String: text data is often imported into R as strings (i.e. character vectors).

Corpus: a collection of raw strings annotated with additional meta data and details.

Document-term matrix: a sparse matrix representing a collection (i.e. a corpus) of documents, in which each row stands for a document, each column stands for a term/token, and each entry is, e.g. TFIDF.

unnest_tokens: a function in R-package tidytext which transform text strings to tidy text format via data_frame

```
text1 = c("Long ago, big data was a thick screen, I was here, mainframe computing was
        there", "And now, big data is a thin smart-phone, I am here, cloud computing
        is there", "In future, big data will be tiny particles, I will be here,
        quantum computing will be there")
               # Poem "Big Data" by Professor Yazhen Wang
> class(text1)
[1] "character"
 length(text1)
[1] 3
> text1
[1] "Long ago, big data was a thick screen, I was here, mainframe computing was there"
[2] "And now, big data is a thin smart-phone, I am here, cloud computing is there"
[3] "In future, big data will be tiny particles, I will be here, quantum computing
       will be there"
```

text1 is a typical text vector to be analysed. In order to turn it into a tidy text dataset, we need to put it into a data frame using data_frame (NOT data.frame!!!).

```
> library(dplyr)
> text1_df = data_frame(text1)
> text1_df
```

```
# A tibble: 3 x 1
   text1
1 Long ago, big data was a thick screen, I am here, mainframe computing was there
2 And now, big data is a thin smart-phone, I am here, cloud computing is there
3 In future, big data will be tiny particles, I will be here, quantum computing ...
```

A tibble is a modern version of data frame. `read_csv` imports data into the tibble format.

```
> install.packages("tidytext")
> library(tidytext)
> unnest_tokens(text1_df, word1, text1)
# A tibble: 48 x 1
   word1
   <chr>
 1 long
 2 ago
 3 big
 4 data
 5 was
 6 a
 7 thick
 8 screen
 9 i
10 was
# ... with 38 more rows
```

```
> text1_tidy=unnest_tokens(text1_df, word1, text1)
> text1_tidy$word1
 [1] "long"      "ago"       "big"       "data"      "was"       "a"         "thick"
 [8] "screen"    "i"         "was"        "here"      "mainframe" "computing" "was"
[15] "there"     "and"       "now"       "big"       "data"      "is"        "a"
[22] "thin"      "smart"     "phone"     "i"         "am"        "here"      "cloud"
[29] "computing" "is"        "there"     "in"        "future"    "big"       "data"
[36] "will"      "be"        "tiny"      "particles"  "i"         "will"      "be"
[43] "here"      "quantum"   "computing" "will"      "be"        "there"
```

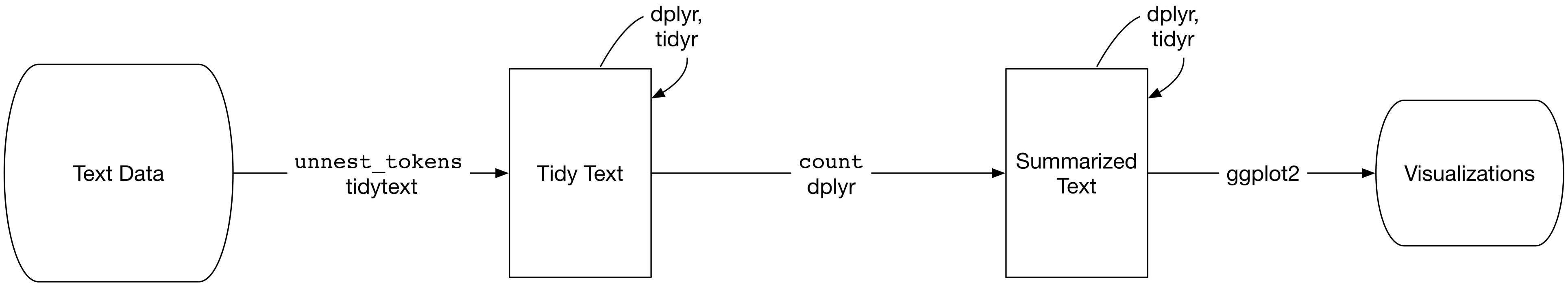
Output vector `word1` discards puctuation, converts the tockens (i.e. words) to lowercase, change `smart-phone` to `smart` and `phone`.

Using pipe: %>%

Command `text1_tidy=unnest_tokens(text1_df, word1, text1)` can be equivalently written as

```
> text1_tidy = text1_df %>% unnest_tokens(word1, text1)
```

Data in tidy-text format allow further analysis as illustrated below



For example

```
> count(text1_tidy, word1, sort=T)
# A tibble: 30 x 2
   word1          n
   <chr>      <int>
 1 be             3
 2 big            3
 3 data           3
 4 here           3
 5 i              3
 6 there          3
 7 will           3
 8 a              2
```

```
 9 was           2
10 computing     2
# ... with 20 more rows
```

The above steps can be combined together using pipes:

```
> text1_df %>% unnest_tokens(word1, text1) %>% count(word1, sort=T)
```

Now let us look at the novels by Jane Austen.

```
> install.packages("janeaustenr")
> library(janeaustenr); library(dplyr); library(tidytext)
> prideprejudice[1:11]
 [1] "PRIDE AND PREJUDICE"
 [2] ""
 [3] "By Jane Austen"
 [4] ""
 [5] ""
 [6] ""
 [7] "Chapter 1"
 [8] ""
 [9] ""
[10] "It is a truth universally acknowledged, that a single man
          in possession"
[11] "of a good fortune, must be in want of a wife."
> PP_df <- data_frame(prideprejudice)
> PP_tidy <- PP_df %>% unnest_tokens(word, prideprejudice)
```

Now all the words in *Pride & Prejudice* are in tidy text file `PP_tidy`.

To load the database of stop words, `data(stop_words)`. Note vector `stop_words` contains all the stop words from 3 lexicons `SMART`, `snowball`, `onix`. To use only the stop words from one lexicon,
`stopwords1 = filter(stop_words, lexicon=="SMART")`.

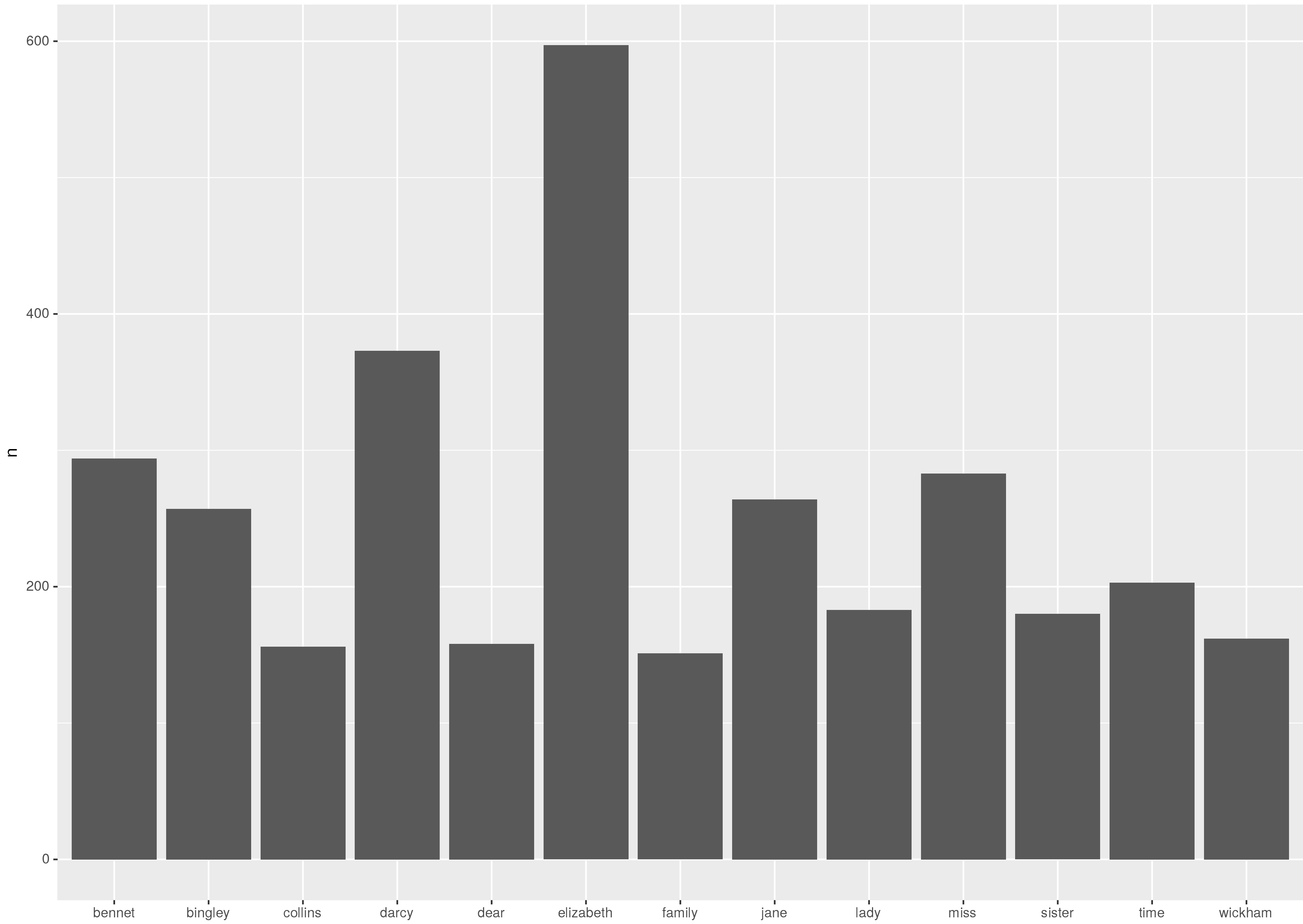
To separate stop words from the others in `PP_tidy`:

```
> PP_noS <- anti_join(PP_tidy, stop_words)
          # extract non-stop words from PP_tidy
> PP_stop <- semi_join(PP_tidy, stop_words)
          # extract stop words from PP_tidy
```

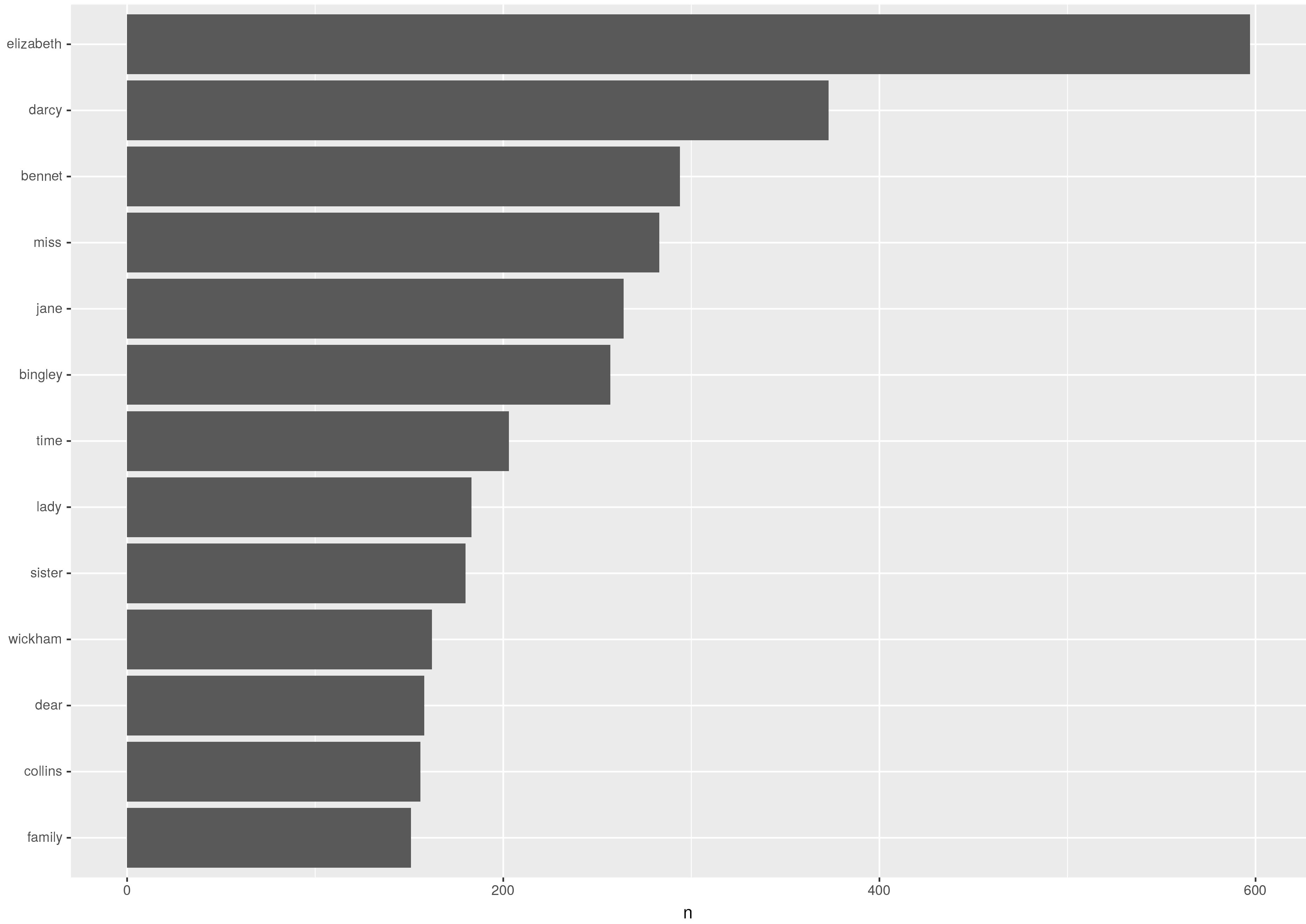
Now we produce a word-frequency bar-chart using `ggplot2`. It also illustrates the usefulness of piping `%>%`.

```
> library(ggplot2)
> PP_noS %>% count(word, sort=T)
# A tibble: 6,009 x 2
   word           n
   <chr>      <int>
 1 elizabeth    597
 2 darcy        373
 3 bennet       294
 4 miss         283
 5 jane         264
 6 bingley      257
 7 time         203
 8 lady         183
 9 sister       180
10 wickham      162
# ... with 5,999 more rows
> PP_noS %>% count(word, sort=T) %>% filter(n>150)
# A tibble: 13 x 2
   word           n
   <chr>      <int>
 1 elizabeth    597
 2 darcy        373
 3 bennet       294
 4 miss         283
 5 jane         264
```

```
 6 bingley     257
 7 time        203
 8 lady        183
 9 sister      180
10 wickham     162
11 dear        158
12 collins     156
13 family      151
> PP_noS %>% count(word, sort=T) %>% filter(n>150) %>% ggplot(aes(word,n)) +
+ geom_col() +
+ xlab(NULL)
            # Produce 1st figure
> PP_noS %>% count(word, sort=T) %>% filter(n>150) %>% mutate(word=reorder(word,n)) %>%
+ ggplot(aes(word,n)) +
+ geom_col() +
+ xlab(NULL) +
+ coord_flip()
            # Produce 2nd figure
>
```



600
400
200
0
n
bennet
bingley
collins
darcy
dear
elizabeth
family
jane
lady
miss
sister
time
wickham


elizabeth
darcy
bennet
miss
jane
bingley
time
lady
sister
wickham
dear
collins
family
0
200
400
600
n

To produce a word cloud plot:

```
> install.packages("wordcloud2")
> library(wordcloud2)
> PP_noS %>% count(word, sort=T) %>% wordcloud2()
```

You may also try

```
> PP_noS %>% count(word, sort=T) %>% filter(n>60) %>% wordcloud2()
```


elizabeth
darcy
bingley
bennet
miss
jane
lady
collins
wickham
sister
time
family
dear
mother
lydia
letter
catherine
father
friend
day
longbourn
netherfield
pleasure
hope
house
replied
lizzy
manner
kitty
till
charlotte
gardiner
love
aunt
colonel
happy
daughter
feelings
attention
conversation
acquaintance
opinion
subject
moment
pemberley
immediately
answer
sisters
heard
ladies
found
cried
marriage
affection
happiness
married
character
behaviour
rosings
london
meryton
morning
uncle
brother
evening
daughters
speak
told
half
town
home
hear
left
life
sir
ill
pride
lucas
world
moment
received
returned
mind
means
coming
leave
carriage
cousin
people
object
business
thousand
handsome
expected
fortune
countenance
amiable
spirits
situation
looked
honour
person
party
regard
surprise
chapter
added
called
friends
passed
believed
manners
advantage
suppose
astonishment
settled
visit
short
return
idea
poor
agreeable
forster
husband
spoke
officers
ladyship
talking
continued
consequence
neighbourhood
wickham's
bingley's
darcy's
elizabeth's
hertfordshire
gentlemen
perfectly
impossible
scarcely
bourgh
understand
kindness
word
dance
night
assure
answered
curiosity
satisfaction
pleased
appearance
wished
importance
brought
circumstances
entered
william
convinced
daughters
occasion
satisfied
hurst
woman
head
dinner
eyes
girls
talk
table
glad
doubt
meet
living
ball
care
sense
rest
real
fixed
walk
country
hour
mary
offer
engaged
reason
civility
absence
speaking
allowed
attentions
week
hours
days
marry
door
proper
sort
wife
son
fine
sat
set
vain
fair
sorrow
stay
ten
walked
sisters
heart
lydia's
invitation
pretty
instantly
knowing
nature
degree
receive
talked
head
ready
comfort
joined
anxious
silence
equal
account
words
usual
park
attached
conduct
expressed
praise
mentioned
express
resolved
particulars
months
hunsford
forward
minutes
terms
question
design
reply
acquainted
bennet's
fitzwilliam
information
inclination
supposed
gratitude
delight
relations
company
exceedingly
surprised
children
gentleman
prevented
pounds
promised
knowledge
hand
favour
success
library
determined
phillips
written
drawing
giving
imagine
anxiety
hoped
afraid
wishes
expect
spent
money
sister's
obliged
meant
blame
forced
hopes
proud
bear
taking
pain
view
met
lost
feel
air
de
that
you
call
excellent
word
pray
mistaken
dancing
considered
intended
addressed
twenty
picture
health
scene
parsonage
confess
favourite
superior
distress
justice
madam
trust
remember
dearest
compliment
disposition
breakfast
remained
hearing
mention
persuaded
delighted
pleasing
understanding
extraordinary
natural
charming
admiration
inquiries
opportunity
depend
concern
meeting
fortnight
regret
beauty
laugh
notice
chance
credit
tone
society
expression
engagement
brighton
mother's
accepted
marrying
removed
prevailed
receiving
frequently
temper
belief
affected
collins's
common
dislike
easily
spite
length
humour
expectation
inquiry
journey
directly
observed
ashamed
appeared
wholly
address
estate
respect
regiment
older
circumstance
acknowledged
nephew
sake
dine
girl
match
real
fear
bring
beg
send
ago
choose
easy
equally
read
true
dare
cold
strong
voice
write
truth
arrival
news
eliza
smile
usual
silent
happened
declared
ignorant
smiled
connection
prevent
intelligence
arrived
capable
pleasant
formed
indifference
questions
fortunate
purpose
objection
politeness
concerned
derbyshire
chief
worth
accept
excuse
intention
clothes
application
charlotte's
listened
admire
reached
suddenly
likewise
letters
scheme
walking
repeated
attempt
forget
recommended
settled
master
play
danger
loss
affair
niece
desire
that
poor
yesterday
resentment
servant
conviction
accomplished
grateful
disappointed
wishing
disgusted
persuaded
observed
advice
change
attachment
wrong
earnest
absolutely

Suppose we try to identify the authorship of a novel. One effective approach is to compare the relative frequencies of stop words in novels.

```
> emma_df <-  data_frame(emma)
> emma_tidy <- emma_df %>% unnest_tokens(word, emma)
> emma_stop <- emma_tidy %>% semi_join(stop_words)
> emma_noS <- emma_tidy %>% anti_join(stop_words)
> dim(emma_noS); dim(emma_stop); dim(PP_noS); dim(PP_stop)
[1] 46775       1
[1] 114221      1
[1] 37246       1
[1] 84958       1
```

Surprisingly both novels contain far more stop words than non-stop words.

```
> bind_rows(mutate(PP_stop, book="Pride & Prejudice"), mutate(emma_stop, book="Emma"))
        # mutate adds a new column to data.frame
        # bind_rows binds data.frames with the same number columns together
# A tibble: 199,179 x 2
   word  book
   <chr> <chr>
 1 and   Pride & Prejudice
 2 by    Pride & Prejudice
```

```
 3 it    Pride & Prejudice
 4 is    Pride & Prejudice
 5 a     Pride & Prejudice
 6 that  Pride & Prejudice
 7 a     Pride & Prejudice
 8 man   Pride & Prejudice
 9 in    Pride & Prejudice
10 of    Pride & Prejudice
# ... with 199,169 more rows

> bind_rows(mutate(PP_stop,book="Pride & Prejudice"),mutate(emma_stop,book="Emma")) %>%
+ count(book, word)
# A tibble: 1,056 x 3
   book  word             n
   <chr> <chr>        <int>
 1 Emma  a             3129
 2 Emma  able            72
 3 Emma  about          249
 4 Emma  above           12
 5 Emma  according        5
 6 Emma  accordingly      4
 7 Emma  across           7
 8 Emma  actually        29
 9 Emma  after          161
10 Emma  afterwards      41
```

```
# ... with 1,046 more rows

> bind_rows(mutate(PP_stop,book="Pride & Prejudice"),mutate(emma_stop,book="Emma")) %>%
+ count(book, word) %>% mutate(proportion=n/sum(n))
# A tibble: 1,056 x 4
   book  word            n proportion
   <chr> <chr>       <int>      <dbl>
 1 Emma  a            3129  0.0157
 2 Emma  able           72  0.000361
 3 Emma  about         249  0.00125
 4 Emma  above          12  0.0000602
 5 Emma  according       5  0.0000251
 6 Emma  accordingly     4  0.0000201
 7 Emma  across          7  0.0000351
 8 Emma  actually       29  0.000146
 9 Emma  after         161  0.000808
10 Emma  afterwards     41  0.000206

> bind_rows(mutate(PP_stop,book="Pride & Prejudice"),mutate(emma_stop,book="Emma")) %>%
+ count(book, word) %>% mutate(proportion=n/sum(n)) %>% select(-n)
# A tibble: 1,056 x 3
   book  word        proportion
   <chr> <chr>            <dbl>
 1 Emma  a            0.0157
 2 Emma  able         0.000361
```

```
 3 Emma   about         0.00125
 4 Emma   above         0.0000602
 5 Emma   according     0.0000251
 6 Emma   accordingly   0.0000201
 7 Emma   across        0.0000351
 8 Emma   actually      0.000146
 9 Emma   after         0.000808
10 Emma   afterwards    0.000206
# ... with 1,046 more rows
```

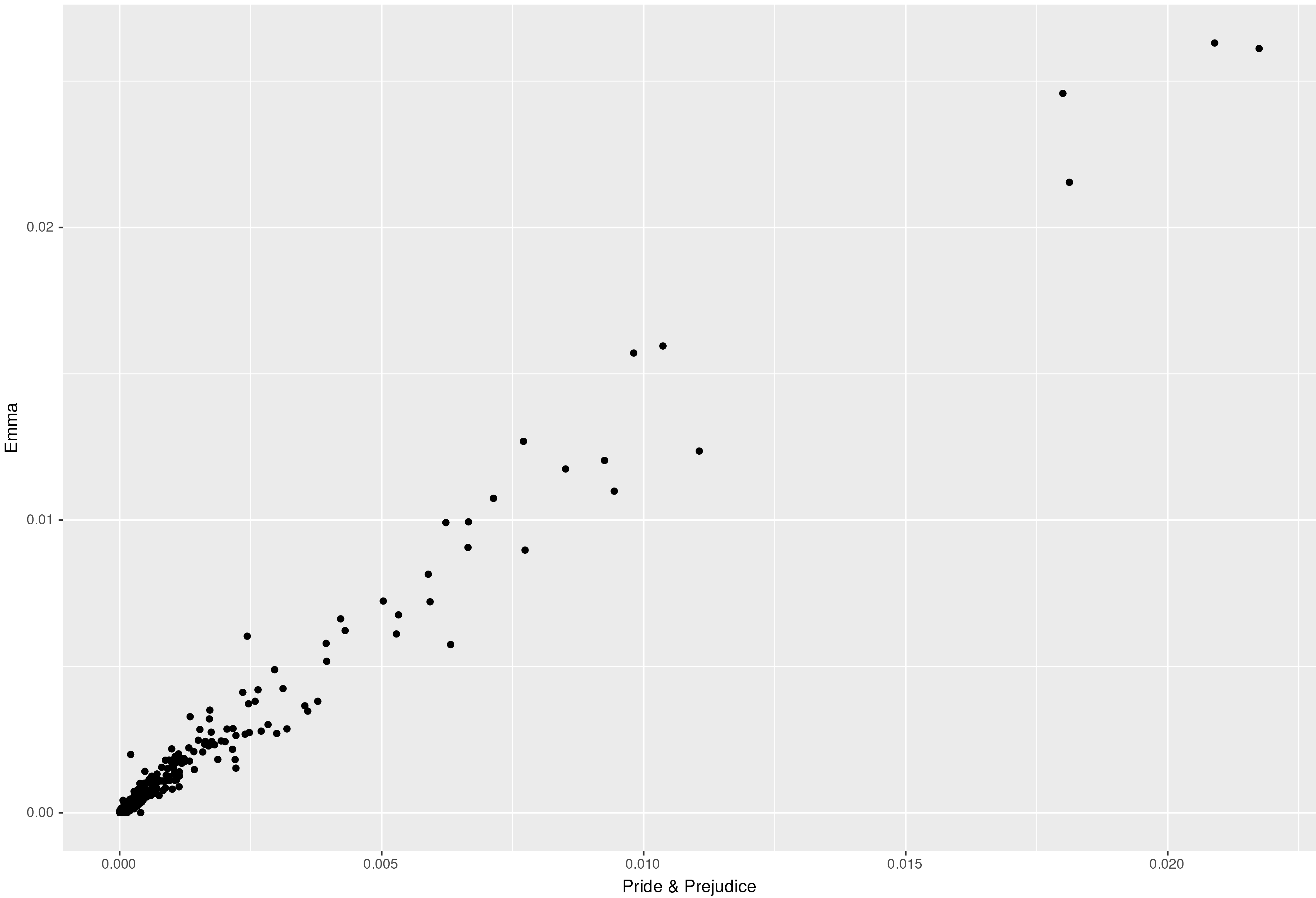
Now we use `spread` (& `gather`) in `tidyr` to put the data in the shape for comparison:

```
> library(tidyr)
> bind_rows(mutate(PP_stop,book="Pride & Prejudice"),mutate(emma_stop,book="Emma")) %>%
+ count(book, word) %>% mutate(proportion=n/sum(n)) %>% select(-n) %>%
+ spread(book, proportion)
# A tibble: 564 x 3
   word          Emma                 `Pride & Prejudice`
   <chr>            <dbl>                    <dbl>
 1 a             0.0157               0.00981
 2 able          0.000361             0.000271
 3 about         0.00125              0.000613
 4 above         0.0000602            0.000105
 5 according     0.0000251            0.0000402
```

```
 6 accordingly 0.0000201           0.0000301
 7 across      0.0000351           0.0000251
 8 actually    0.000146            0.0000602
 9 after       0.000808            0.00100
10 afterwards  0.000206            0.000161
# ... with 554 more rows
> rF = bind_rows(mutate(PP_stop, book="Pride & Prejudice"),
                mutate(emma_stop, book="Emma")) %>%
+ count(book, word) %>% mutate(proportion=n/sum(n)) %>% select(-n) %>%
+ spread(book, proportion)
> qplot(rF[,3], rF[,2], ylab="Emma", xlab="Pride & Prejudice",
       main="Relative frequencies of stop words in two novels")
```

The figure shows that the relative frequencies of the occurrency of stop words in the two Austen's novels are similar.

Relative frequencies of stop words in two novels



To compare Austen's writings with others, we download 2 Dickens' books from http://www.gutenberg.org/ebooks/.

First, Dickens' Great Expectation in html format.

```
> install.packages("rvest") # Package for easy scrape of web pages
> library(rvest)
> GE <- read_html("http://www.gutenberg.org/files/1400/1400-h/1400-h.htm")
> GE_text=GE %>% html_nodes("p") %>% html_text()  # Extract text from html file
> GE_df = data_frame(GE_text)
> GE_tidy = GE_df %>% unnest_tokens(word, GE_text)
> GE_stop = GE_tidy %>% semi_join(stop_words)
```

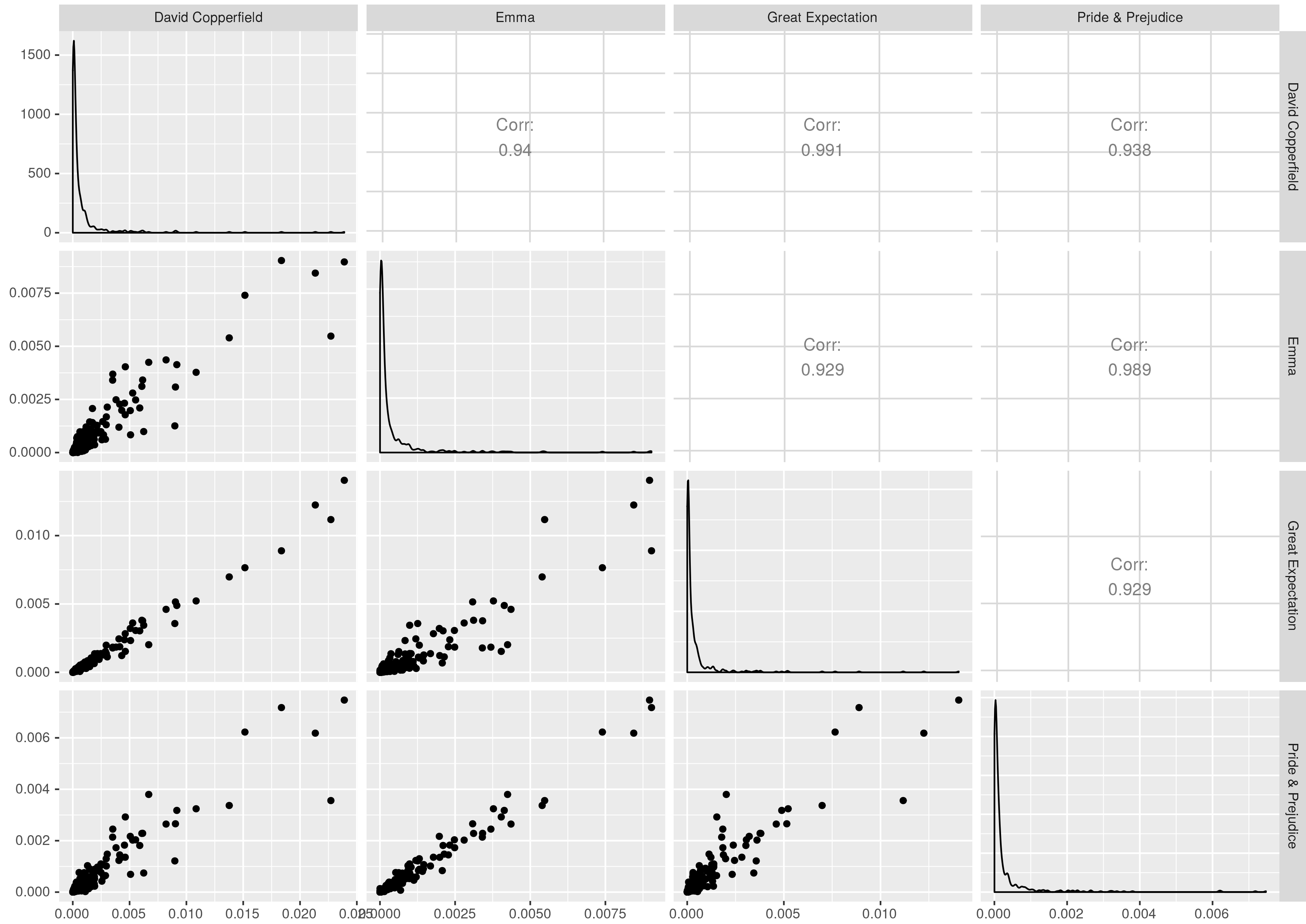
To get Dickens' David Copperfield,

```
> DC = read_html("http://www.gutenberg.org/files/9744/9744-index.htm")
> DC_text = DC %>% html_nodes("p") %>% html_text()
> DC_df = data_frame(DC_text)
> DC_tidy = DC_df %>% unnest_tokens(word, DC_text)
> DC_stop = DC_tidy %>% semi_join(stop_words)
```

Now we combine the relative frequencies of stop words in 4 books together to produce a plot for comparison.

```
> rF4 = bind_rows(mutate(PP_stop, book="Pride & Prejudice"),
     mutate(emma_stop, book="Emma"),
     mutate(GE_stop, book="Great Expectation"),
     mutate(DC_stop, book="David Copperfield")) %>%
+ count(book, word) %>% mutate(proportion=n/sum(n)) %>% select(-n) %>%
+ spread(book, proportion)
> rF4
# A tibble: 659 x 5
   word        `David Copperfield`   Emma     `Great Expectation`   `Pride & Prejudice`
   <chr>                   <dbl>      <dbl>                 <dbl>                 <dbl>
 1 a               0.0138         0.00540           0.00698               0.00337
 2 able            0.0000707      0.000124          0.0000552             0.0000931
 3 about           0.00114        0.000429          0.000552              0.000210
 4 above           0.0000966      0.0000207         0.0000552             0.0000362
 5 according       0.0000310      0.00000862        0.0000310             0.0000138
 6 accordingly     0.0000362      0.00000690        0.00000345            0.0000103
 7 across          0.0000948      0.0000121         0.0000759             0.00000862
 8 actually        0.0000276      0.0000500         0.0000172             0.0000207
 9 after           0.000769       0.000278          0.000504              0.000345
10 afterwards      0.000203       0.0000707         0.0000724             0.0000552
# ... with 649 more rows

> rF4c = rF4 %>% drop_na()   # Drop the rows with "na"
> library("GGally", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.2")
> ggpairs(rF4c[,2:5])
```



David Copperfield
Emma
Great Expectation
Pride & Prejudice
Corr:
0.94
Corr:
0.991
Corr:
0.938
Corr:
0.929
Corr:
0.989
Corr:
0.929