Solution slides
Introduction to R & Data for Humanities

Afternoon session
Text-mining with Tidyverse

Exercise 7

7a.

```
```{r}
# Exercise 7a.Now let's start tidy-texting Dickinson's poem. Run the following lines of code:

text <- c("Because I could not stop for Death -",
          "He kindly stopped for me -",
          "The Carriage held but just Ourselves -",
          "and Immortality")
text
```
```

```
[1] "Because I could not stop for Death -"   "He kindly stopped for me -"           "The Carriage held but just Ourselves -"
[4] "and Immortality"
```

7b.

```{r}
# Excercise 7b. Let's call on a package from the Tidyverse that will give us the right data frame: dplyr. You can call on this package by running the following code:


library(dplyr)
text_df <- tibble(line = 1:4, text = text)

text_df
```

line <int>	text <chr>
1	Because I could not stop for Death -
2	He kindly stopped for me -
3	The Carriage held but just Ourselves -
4	and Immortality

4 rows

7c.

```{r}
# Excercise 7c. We will now break the text into individual tokens (tokenization) and transform it to a tidy data structure. To do this, call on the unnest_tokens() function:

library(tidytext)

text_df %>%
  unnest_tokens(word, text)
```

line <int>	word <chr>
1	because
1	i
1	could
1	not
1	stop
1	for
1	death
2	he
2	kindly
2	stopped

1-10 of 20 rows

Previous 1 2 Next

8a.

```r
# Excercise 8a. Based on the previous exercises with Dickenson's poem, are you now able to call on the janeaustenr package, as well as the dplyr and stringr packages needed for your analysis?

library(janeaustenr)
library(dplyr)
library(stringr)
```

```
package 'janeaustenr' was built under R version 4.0.5package 'stringr' was built under R version 4.0.5
```

← Don't worry about the warnings, the packages should work just fine!

8b.

```r
# Excercise 8b. Run the following code… and then challenge yourself with exercise 8c!

original_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                                     regex("^chapter [\\divxlc]",
                                           ignore_case = TRUE)))) %>%
  ungroup()

original_books
```

text <chr>	book <fctr>	linenumber <int>
	Sense & Sensibility	31
By a former marriage, Mr. Henry Dashwood had one son: by his present	Sense & Sensibility	32
lady, three daughters. The son, a steady respectable young man, was	Sense & Sensibility	33
amply provided for by the fortune of his mother, which had been large,	Sense & Sensibility	34
and half of which devolved on him on his coming of age. By his own	Sense & Sensibility	35
marriage, likewise, which happened soon afterwards, he added to his	Sense & Sensibility	36
wealth. To him therefore the succession to the Norland estate was not	Sense & Sensibility	37
so really important as to his sisters; for their fortune, independent	Sense & Sensibility	38
of what might arise to them from their father's inheriting that	Sense & Sensibility	39
property, could be but small. Their mother had nothing, and their	Sense & Sensibility	40

31-40 of 73,422 rows | 1-3 of 4 columns

Previous 1 2 3 4 5 6 … 100 Next

When you see the output at first, you see a lot of blank lines in the ‘text’ column. This is because the title page is also taken into account. When you browse through the table, you will soon encounter actual lines of text.

If you press the arrow next to the ‘linenumber’ column, you see the number of the chapter the line is in.

8c.

```r
# Excercise 8c

library(tidytext)
tidy_books <- original_books %>%
  unnest_tokens(word, text)

tidy_books
```

book <fctr>	linenumber <int>	chapter <int>	word <chr>
Sense & Sensibility	1	0	sense
Sense & Sensibility	1	0	and
Sense & Sensibility	1	0	sensibility
Sense & Sensibility	3	0	by
Sense & Sensibility	3	0	jane
Sense & Sensibility	3	0	austen
Sense & Sensibility	5	0	1811
Sense & Sensibility	10	1	chapter
Sense & Sensibility	10	1	1
Sense & Sensibility	13	1	the

1-10 of 725,055 rows

Previous 1 2 3 4 5 6 ... 100 Next

8d.

```r
# Excercise 8d. This is the first time you call on a dataset from a package. Can you call on the anti_join() function by completing the code below?


data(stop_words)

tidy_books <- tidy_books %>%
  anti_join(stop_words)

# We can also use dplyr's count() to find the most common words in all the books as a whole.
tidy_books %>%
  count(word, sort = TRUE)

```

word <chr>	n <int>
miss	1855
time	1337
fanny	862
dear	822
lady	817
sir	806
day	797
emma	787
sister	727
house	699

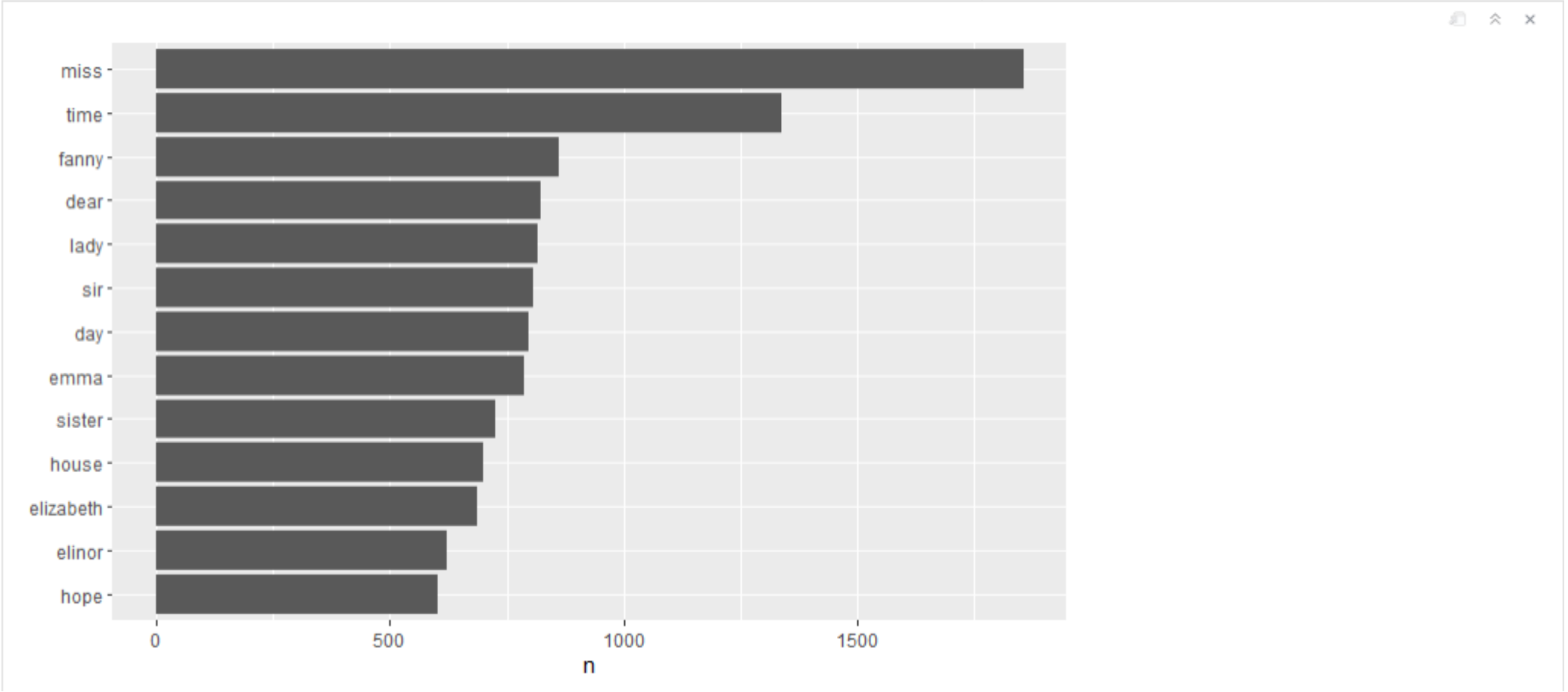
1-10 of 13,914 rows

8e.

```{r}
# Excercise 8e. Let's build ourselves a pipeline! Run this code and see what happens…

library(ggplot2)

tidy_books %>%
  count(word, sort = TRUE) %>%
  filter(n > 600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



Exercise 9

9a.

```r
#Exercise 9a. Can you call on the tidytext package and then use the function mentioned above to get tibbles of the three lexicons mentioned on the slide? Press 'enter' twice and start coding!

library(tidytext)

get_sentiments("afinn")
get_sentiments("bing")
get_sentiments("nrc")
```

spec_tbl_df 2477 x 2 | tbl_df 6786 x 2 | tbl_df 13901 x 2

word <chr>	sentiment <chr>
abacus	trust
abandon	fear
abandon	negative
abandon	sadness
abandoned	anger
abandoned	fear
abandoned	negative
abandoned	sadness
abandonment	anger
abandonment	fear

1-10 of 13,901 rows

Previous 1 2 3 4 5 6 ... 100 Next

See next slide for a characteri-zation of how these lexicons score sentiment

9a. (resumed)

All three of these lexicons are based on unigrams, i.e., single words. These lexicons contain many English words and the words are assigned scores for positive/negative sentiment, and also possibly emotions like joy, anger, sadness, and so forth. The nrc lexicon categorizes words in a binary fashion (“yes”/“no”) into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. The bing lexicon categorizes words in a binary fashion into positive and negative categories. The AFINN lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

9b.

```
153
154 # Sentiment analysis of Emma I
155 ```{r}
156
157 # Exercise 9b.Let's ask ourselves: What are the most common joy words in Austen's novel Emma? Run this code in order to make
    your data tidy first and do some real code work in the next exercise!
158
159 library(janeaustenr)
160 library(dplyr)
161 library(stringr)
162
163 tidy_books <- austen_books() %>%
164   group_by(book) %>%
165   mutate(
166     linenumber = row_number(),
167     chapter = cumsum(str_detect(text,
168                                 regex("^chapter [\\divxlc]",
169                                       ignore_case = TRUE)))) %>%
170   ungroup() %>%
171   unnest_tokens(word, text)
172 ```
```

170:16 | Chunk 12 | R Markdown

Console | Terminal | Jobs

C:/WINDOWS/system32/

```
>
> library(janeaustenr)
> library(dplyr)
> library(stringr)
>
> tidy_books <- austen_books() %>%
+   group_by(book) %>%
+   mutate(
+     linenumber = row_number(),
+     chapter = cumsum(str_detect(text,
+                                 regex("^chapter [\\divxlc]",
+                                       ignore_case = TRUE)))) %>%
+   ungroup() %>%
+   unnest_tokens(word, text)
>
```

We only run this code to make sure that our data is tidy; there is no visible output you need to take into account.

9c.

```r
# Exercise 9c.We want to know what the most common joy words in Emma are. Can you complete the code and run the script based on the pointers on the slide?

nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_books %>%
  filter(book == "Emma") %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)
```

R Console

tbl_df
303 x 2

word <chr>	n <int>
good	359
young	192
friend	166
hope	143
happy	125
love	117
deal	92
found	92
present	89
kind	82

1-10 of 303 rows

Previous 1 2 3 4 5 6 ... 31 Next

9d.

```r
# Exercise 9d. We can also examine how sentiment changes throughout each of Austen's novels. We can do this with just a handful
of lines that are mostly dplyr functions. Can you complete the code and run the script based on the pointers on the slide?


library(tidyr)

jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
package 'tidyr' was built under R version 4.0.5Joining, by = "word"
```

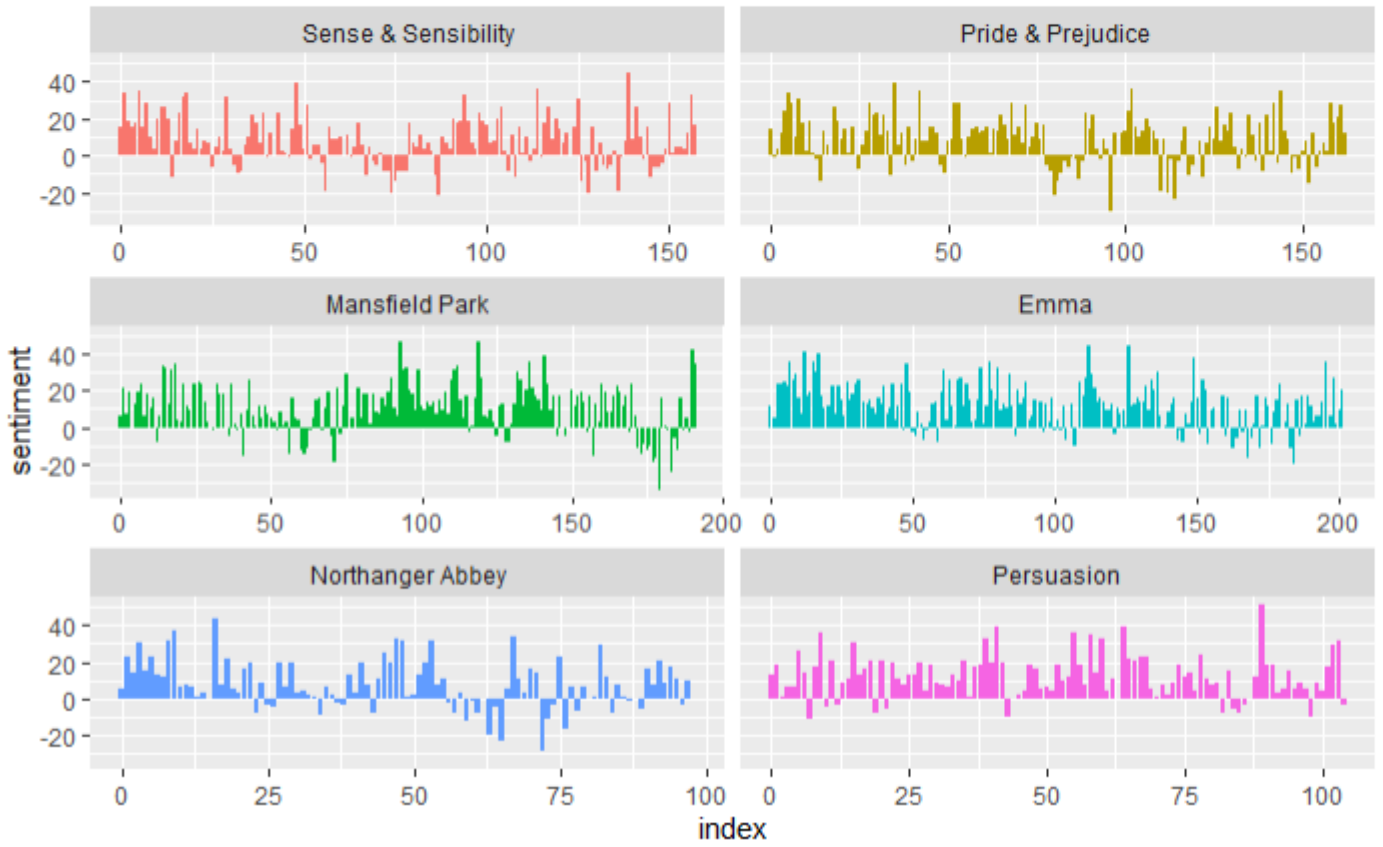
We run this code as a precursor to visualizing how sentiment changes throughout each of Austen's novels, so there is no visible output you need to take into account right now.

If you see the warning in red, you can safely ignore it.

9e.

```r
#Exercise 9e. Call on the ggplot2 package (do you remember how to call on a package? You have done so numerous times before!)
and then run the following code:


library(ggplot2)
ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```



9e. (resumed)

Based on these graphs, we can begin to explore trends or differences in the novels' sentiment structures. For example, how the plot of each novel changes toward more positive or negative sentiment over the trajectory of the story. Based on your observations of the visualization you might want to start close reading certain passages of the novels, in order to analyze the specific language used in specific sections. You could also use these graphs as a starting point to look into how Austen's writing changes over time when it comes to the sentiment character of her novels.

Exercise 10

10a.

Note that the usual suspects are here with the highest n, “the”, “and”, “to”, and so forth.

```{r}
# Exercise 10a.Let's start by looking at the novels of Austen and examine first term frequency, then tf-idf. We can start just by using dplyr verbs such as group_by() and join(). Can you fill in the blanks in the code below based on what you have learned so far and determine the most commonly used words in the novels? (Let's also calculate the total words in each novel here, for later use)

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)

book_words
```

book <fctr>	word <chr>	n <int>	total <int>
Mansfield Park	the	6206	160460
Mansfield Park	to	5475	160460
Mansfield Park	and	5438	160460
Emma	to	5239	160996
Emma	the	5201	160996
Emma	and	4896	160996
Mansfield Park	of	4778	160460
Pride & Prejudice	the	4331	122204
Emma	of	4291	160996
Pride & Prejudice	to	4162	122204

1-10 of 40,379 rows

Previous 1 2 3 4 5 6 ... 100 Next

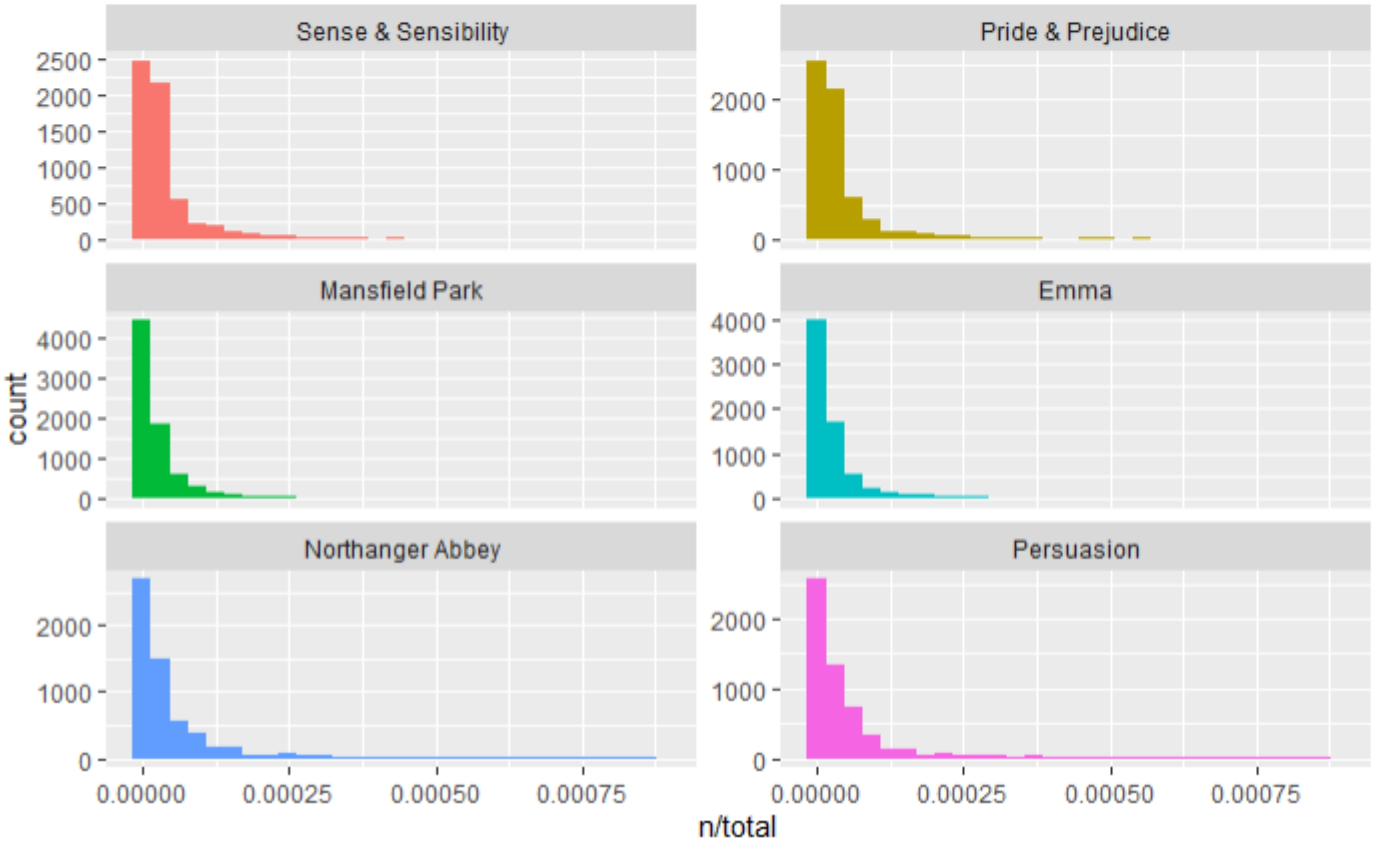
10b.

```r
# Exercise 10b. Now let's plot the distribution of n/total = the number of times a word is used in a book/the total words in that book. Do you remember what package to call on to plot this distribution?

library(ggplot2)

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`.
Removed 896 rows containing non-finite values (stat_bin).
Removed 6 rows containing missing values (geom_bar).
```



← These plots exhibit similar distributions for all the novels, with many words that occur rarely and fewer words that occur frequently.

10c.

```r
# Exercise 10c.Based on the column headers on the slide, can you fill in the code below and calculate tf-idf?


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


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

book <fctr>	word <chr>	n <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>
Sense & Sensibility	elinor	623	5.193528e-03	1.7917595	9.305552e-03
Sense & Sensibility	marianne	492	4.101470e-03	1.7917595	7.348847e-03
Mansfield Park	crawford	493	3.072417e-03	1.7917595	5.505032e-03
Pride & Prejudice	darcy	373	3.052273e-03	1.7917595	5.468939e-03
Persuasion	elliot	254	3.036171e-03	1.7917595	5.440088e-03
Emma	emma	786	4.882109e-03	1.0986123	5.363545e-03
Northanger Abbey	tilney	196	2.519928e-03	1.7917595	4.515105e-03
Emma	weston	389	2.416209e-03	1.7917595	4.329266e-03
Pride & Prejudice	bennet	294	2.405813e-03	1.7917595	4.310639e-03
Persuasion	wentworth	191	2.283105e-03	1.7917595	4.090775e-03

1-10 of 40,379 rows

Previous 1 2 3 4 5 6 ... 100 Next

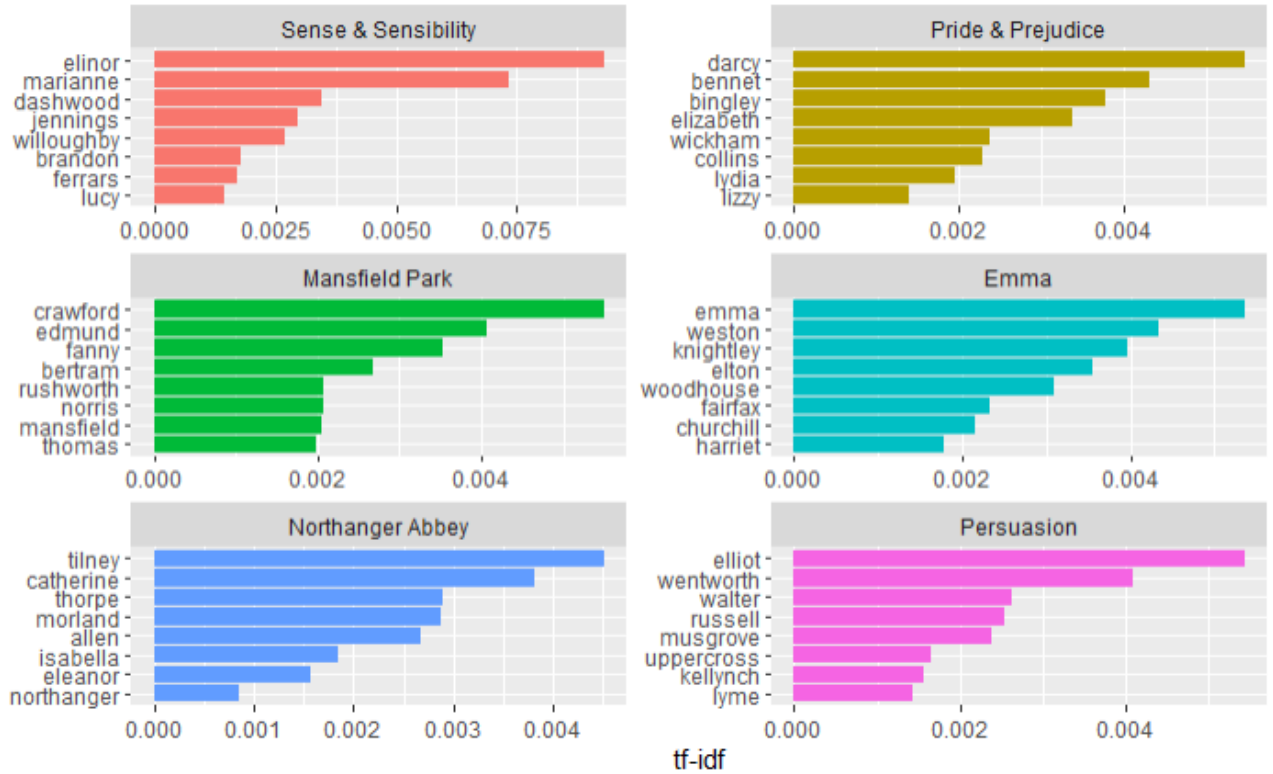
Here we see all proper nouns, names that are in fact important in these novels. None of them occur in all of the novels, and they are important, characteristic words for each text within the corpus of Jane Austen's novels.

10d.

```r
# Excercise 10d.Run the code below to plot the highest tf-idf words in each of Austen's novels. Can you make it so that you plot the scores per novel? And can you make sure that we see tf-idf for the tokens/terms we have been analyzing?

library(forcats)

book_tf_idf %>%
  group_by(book) %>%
  slice_max(tf_idf, n = 8) %>%
  ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```



Exercise 11

11a.

```r
# Excercise 11a. Can you set the number of words in each n-gram to 2 in the code below? This allows us to examine pairs of two consecutive words, called 'bigrams' in Austen's novels

library(dplyr)
library(tidytext)
library(janeaustenr)

austen_bigrams <- austen_books() %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

austen_bigrams
```

book <fctr>	bigram <chr>
Sense & Sensibility	was at
Sense & Sensibility	at norland
Sense & Sensibility	norland park
Sense & Sensibility	park in
Sense & Sensibility	in the
Sense & Sensibility	the centre
Sense & Sensibility	centre of
Sense & Sensibility	their property
Sense & Sensibility	property where
Sense & Sensibility	where for

31-40 of 675,025 rows

Previous 1 2 3 4 5 6 ... 100 Next

Notice that these bigrams overlap: "norland park" is one token, while "park in" is another.

11b.

The output here is based on running the piece of code in blue.

```r
# Exercise 11b.When we count our bigrams using dlpyr's count(), we see that a lot of the most common bigrams are pairs of common words, like stop words. Run this code and you'll see…

austen_bigrams %>%
  count(bigram, sort = TRUE)

# We are of course not only interested in the stop word bigrams. So let's filter our n-grams with tidyr's separate() and remove cases where either word is a stop word. Run it!

library(tidyr)

bigrams_separated <- austen_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)

bigram_counts
```

bigram <chr>	n <int>
NA	12242
of the	2853
to be	2670
in the	2221
it was	1691
i am	1485
she had	1405
of her	1363
to the	1315
she was	1309

1-10 of 193,210 rows

11b. (resumed)

```
# We are of course not only interested in the stop word bigrams. So let's filter our n-grams with tidyr's separate() and remove cases where either word is a stop word. Run it!

library(tidyr)

bigrams_separated <- austen_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)

bigram_counts
```

Looks familiar? Again with the proper nouns, that is: names!

word1 <chr>	word2 <chr>	n <int>
NA	*NA*	12242
sir	thomas	266
miss	crawford	196
captain	wentworth	143
miss	woodhouse	143
frank	churchill	114
lady	russell	110
sir	walter	108
lady	bertram	101
miss	fairfax	98

1-10 of 28,975 rows

Previous 1 2 3 4 5 6 ... 100 Next

11c.

```
# Exercise 11c. We will now use tidyr's unite() function to recombine the columns into one. Using the
"separate/filter/count/unite" combination lets us find the most common bigrams not containing stop-words. Run the code below.


bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")

bigrams_united
```

book <fctr>	bigram <chr>
Sense & Sensibility	fortune independent
Sense & Sensibility	father's inheriting
Sense & Sensibility	thousand pounds
Sense & Sensibility	remaining moiety
Sense & Sensibility	wife's fortune
Sense & Sensibility	NA NA
Sense & Sensibility	gentleman died
Sense & Sensibility	destroyed half
Sense & Sensibility	son's son
Sense & Sensibility	valuable woods

31-40 of 51,155 rows

Previous 1 2 3 4 5 6 ... 100 Next

11d.

```
# Exercise 11d. We can look at the tf-idf of bigrams across Austen's novels. These tf-idf values can be visualized within each book, just as we did for words. Can you complete the code below and produce a tibble and visualization of your results all at once?

bigram_tf_idf <- bigrams_united %>%
  count(book, bigram) %>%
  bind_tf_idf(bigram, book, n) %>%
  arrange(desc(tf_idf))

bigram_tf_idf
```

book <fctr>	bigram <chr>	n <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>
Mansfield Park	sir thomas	266	0.0244238362	1.7917595	0.0437616398
Persuasion	captain wentworth	143	0.0232142857	1.7917595	0.0415944162
Mansfield Park	miss crawford	196	0.0179965109	1.7917595	0.0322454188
Persuasion	lady russell	110	0.0178571429	1.7917595	0.0319957048
Persuasion	sir walter	108	0.0175324675	1.7917595	0.0314139647
Emma	miss woodhouse	143	0.0128817224	1.7917595	0.0230809480
Northanger Abbey	miss tilney	74	0.0127828641	1.7917595	0.0229038177
Sense & Sensibility	colonel brandon	96	0.0114572145	1.7917595	0.0205285725
Sense & Sensibility	sir john	94	0.0112185225	1.7917595	0.0201008939
Emma	frank churchill	114	0.0102693451	1.7917595	0.0184001963

1-10 of 31,397 rows

Exercise 12

12a.

```r
# Exercise 12a. We want to find out what words tend to appear within each 10-line section of Austen's Pride and Prejudice.
Let's first find the Most Common Words, filtering out stop words. Can you complete the code?

austen_section_words <- austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  mutate(section = row_number() %/% 10) %>%
  filter(section > 0) %>%
  unnest_tokens(word, text) %>%
  filter(!word %in% stop_words$word)

austen_section_words
```

book <fctr>	section <dbl>	word <chr>
Pride & Prejudice	1	truth
Pride & Prejudice	1	universally
Pride & Prejudice	1	acknowledged
Pride & Prejudice	1	single
Pride & Prejudice	1	possession
Pride & Prejudice	1	fortune
Pride & Prejudice	1	wife
Pride & Prejudice	1	feelings
Pride & Prejudice	1	views
Pride & Prejudice	1	entering

1-10 of 37,240 rows

Previous 1 2 3 4 5 6 ... 100 Next

12b.

```r
# Exercise 12b.Can you complete the count by using the function mentioned above and providing it with the information on what to count?

library(widyr)

# count words co-occuring within sections
word_pairs <- austen_section_words %>%
  pairwise_count(word, section, sort = TRUE)

word_pairs
```

item1 <chr>	item2 <chr>	n <dbl>
darcy	elizabeth	144
elizabeth	darcy	144
miss	elizabeth	110
elizabeth	miss	110
elizabeth	jane	106
jane	elizabeth	106
miss	darcy	92
darcy	miss	92
elizabeth	bingley	91
bingley	elizabeth	91

1-10 of 796,008 rows

Previous 1 2 3 4 5 6 ... 100 Next

12c.

```{r}

# Excercise 12c.The syntax of the pairwise_corr() function is similar to that of pairwise_count(). Just run it!

# we need to filter for at least relatively common words first
word_cors <- austen_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section, sort = TRUE)

word_cors
```

item1 <chr>	item2 <chr>	correlation <dbl>
bourgh	de	0.9508501
de	bourgh	0.9508501
pounds	thousand	0.7005808
thousand	pounds	0.7005808
william	sir	0.6644719
sir	william	0.6644719
catherine	lady	0.6633048
lady	catherine	0.6633048
forster	colonel	0.6220950
colonel	forster	0.6220950

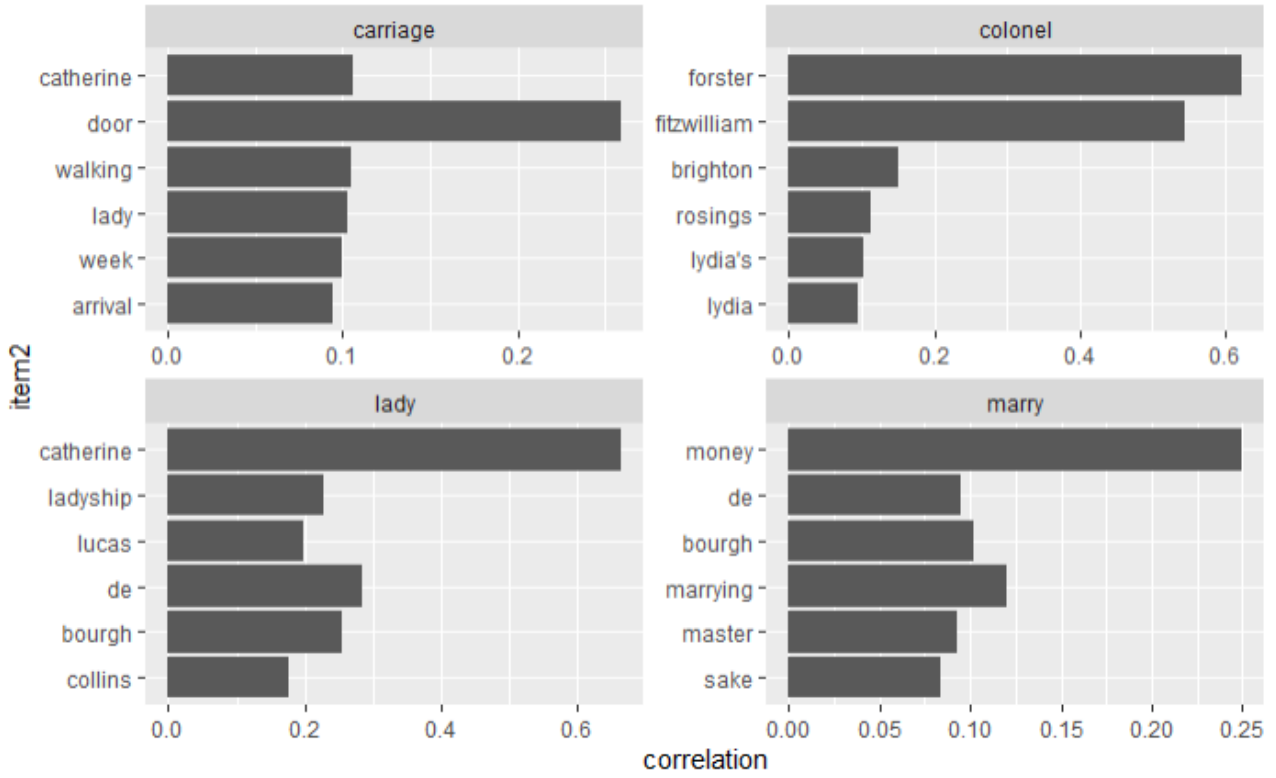
1-10 of 154,842 rows

Previous 1 2 3 4 5 6 ... 100 Next

12d.

```r
# Exercise 12d.Let's pick some interesting words and find the other words most associated with them! You can pick your own and add them to the code below. And do you remember the function we have used a few times to plot your results? Fill it in as well!

word_cors %>%
  filter(item1 %in% c("lady", "colonel", "carriage", "marry")) %>%
  group_by(item1) %>%
  slice_max(correlation, n = 6) %>%
  ungroup() %>%
  mutate(item2 = reorder(item2, correlation)) %>%
  ggplot(aes(item2, correlation)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ item1, scales = "free") +
  coord_flip()
```



One of the word correlations that stand out (both in the tibble in exercise 12c and in this visualization, is the one between “marry” and “money”. This could be a semantic relationship to explore in further analyses.