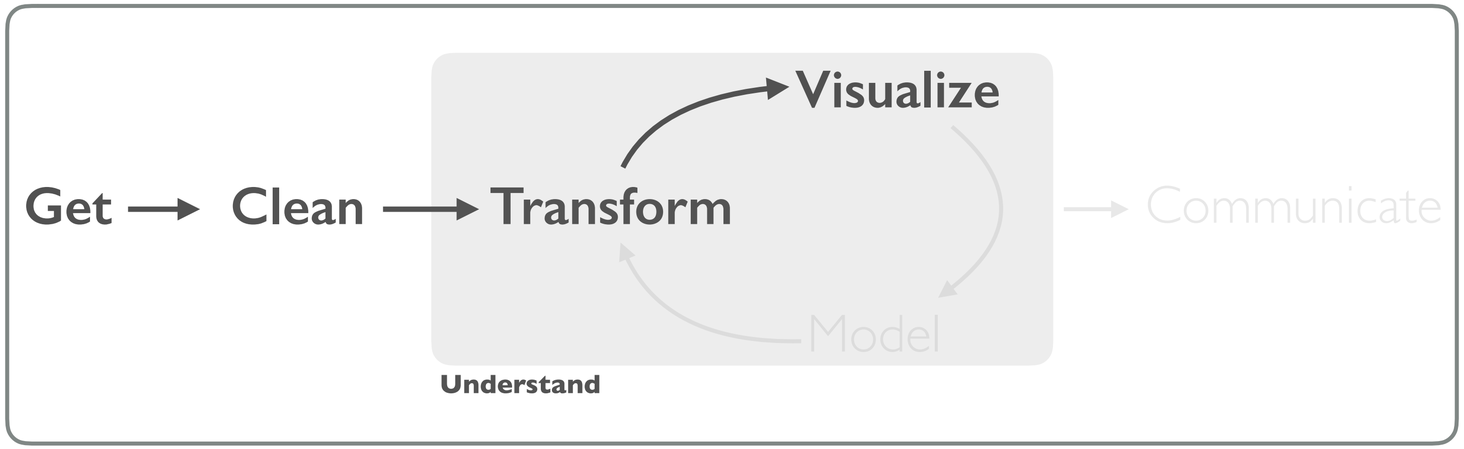
ORGANIZE, CLEAN, & ANALYZE



†A modified version of Hadley Wickham's analytic process

"Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text."

– Julia Silge and David Robinson

WHAT IS TIDY DATA?

- One variable per column
- One observation per row

```
##        userid age dob_day dob_year dob_month gender tenure friend_count
## 1     2094382  14      19     1999        11   male    266            0
## 2     1192601  14       2     1999        11 female      6            0
## 3     2083884  14      16     1999        11   male     13            0
## 4     1203168  14      25     1999        12 female     93            0
## 5     1733186  14       4     1999        12   male     82            0
## 6     1524765  14       1     1999        12   male     15            0
## 7     1136133  13      14     2000         1   male     12            0
## 8     1680361  13       4     2000         1 female      0            0
## 9     1365174  13       1     2000         1   male     81            0
## 10    1712567  13       2     2000         2   male    171            0
## 11    1612453  13      22     2000         2   male     98            0
## 12    2104073  13       1     2000         2   male     55            0
```

TIDY TEXT

From this…

```
philosophers_stone
```

[1] "THE BOY WHO LIVED Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they h... <truncated>

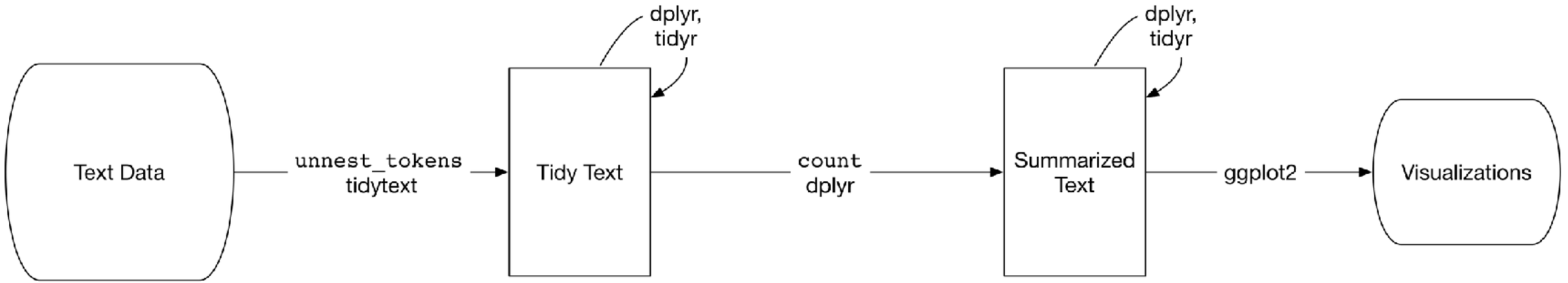
[2] "THE VANISHING GLASS Nearly ten years had passed since the Dursleys had woken up to find their nephew on the front

To this…

	book	chapter	word
	<chr>	<int>	<chr>
1	Philosopher's Stone	1	boy
2	Philosopher's Stone	1	lived
3	Philosopher's Stone	1	dursley
4	Philosopher's Stone	1	privet
5	Philosopher's Stone	1	drive
6	Philosopher's Stone	1	proud
7	Philosopher's Stone	1	perfectly
8	Philosopher's Stone	1	normal
9	Philosopher's Stone	1	people
10	Philosopher's Stone	1	expect

... with 409,328 more rows

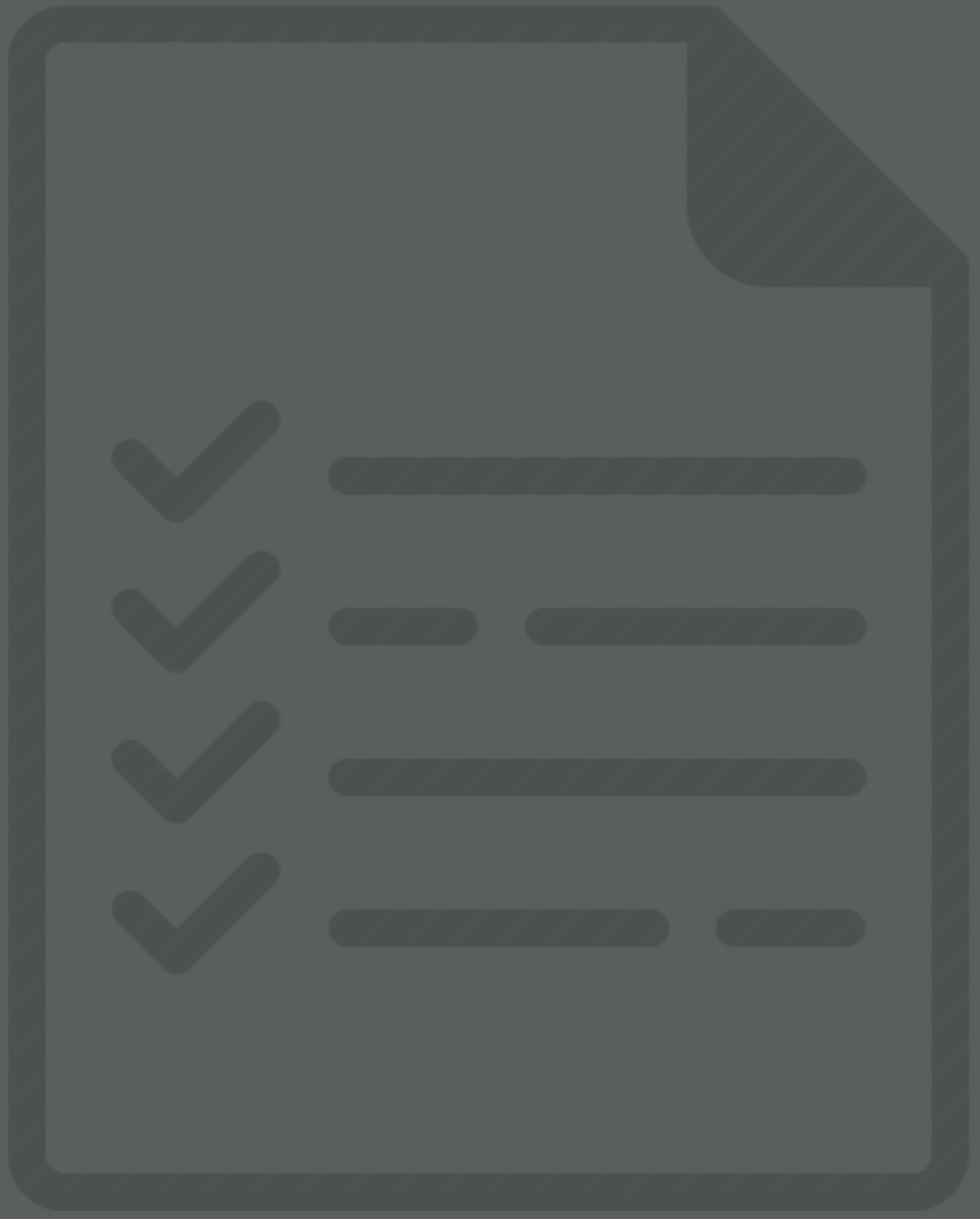
TIDY TEXT


Text Data
unnest_tokens
tidytext
Tidy Text
dplyr,
tidyr
count
dplyr
Summarized
Text
dplyr,
tidyr
ggplot2
Visualizations

STEPS

1. Get text into a data frame
2. Unnest to specified token size
3. Remove stop words
4. Perform frequency analysis
5. Compare proportions across documents

PREREQUISITES



PACKAGE PREREQUISITE

```
library(tidyverse)
library(tidytext)

if (packageVersion("devtools") < 1.6) {
  install.packages("devtools")
}

devtools::install_github("bradleyboehmke/harrypotter")
library(harrypotter)
```

CREATING STRUCTURE

Getting text into a data frame

TEXT TO TIBBLE

- Often text is held in a vector of character strings

```
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 -"
[3] "The Carriage held but just Ourselves -" "and Immortality”
```

TEXT TO TIBBLE

- Often text is held in a vector of character strings
- We can convert these into a data frame (tibble) quite easily

```
tibble(
  index = seq_along(text),
  token = text
)
# A tibble: 4 x 2
  index token
  <int> <chr>
1     1 Because I could not stop for Death -
2     2 He kindly stopped for me -
3     3 The Carriage held but just Ourselves -
4     4 and Immortality
```

Creates a tibble with two variables

- chapter
- all text for that chapter

DATA PREREQUISITE

philosophers_stone

[1] "THE BOY WHO LIVED Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they h... <truncated>

[2] "THE VANISHING GLASS Nearly ten years had passed since the Dursleys had woken up to find their nephew on the front step, but Privet Drive had hardly changed at all. The sun rose on the same tidy front gardens and lit up the brass number four on the Dursleys' front door; it crept into their living room, which was almost exactly the same as it had

TEXT TO TIBBLE

```
text_tb <- tibble(
  chapter = seq_along(philosophers_stone),
  text = philosophers_stone
)
text_tb
# A tibble: 17 x 2
   chapter text
     <int> <chr>
 1       1 "THE BOY WHO LIVED   Mr. and Mrs. Dursley, of n…
 2       2 "THE VANISHING GLASS   Nearly ten years had pas…
 3       3 "THE LETTERS FROM NO ONE   The escape of the Br…
 4       4 "THE KEEPER OF THE KEYS   BOOM. They knocked ag…
 5       5 "DIAGON ALLEY   Harry woke early the next morni…
 6       6 "THE JOURNEY FROM PLATFORM NINE AND THREE-QUAR…
 7       7 "THE SORTING HAT   The door swung open at once.…
 8       8 "THE POTIONS MASTER   There, look.\"   \"Where?\…
 9       9 "THE MIDNIGHT DUEL   Harry had never believed h…
10      10 "HALLOWEEN   Malfoy couldn't believe his eyes w…
```

Creates a tibble with two variables

- chapter
- all text for that chapter

YOUR TURN!

1. *Organize* **`harrypotter::deathly_hallows`** *into a tibble*

2. *Organize* **`harrypotter::chamber_of_secrets`** *into a tibble*

3. *Organize* **`harrypotter::goblet_of_fire`** *into a tibble*

TEXT TO TIBBLE

- But what if we have a series of documents that we want to store and analyze together?

```
df1 <- philosophers_stone
df2 <- chamber_of_secrets
df3 <- prisoner_of_azkaban
df4 <- goblet_of_fire
df5 <- order_of_the_phoenix
df6 <- half_blood_prince
df7 <- deathly_hallows
```

TEXT TO TIBBLE

- But what if we have a series of documents that we want to store and analyze together?

```
df1 <- philosophers_stone
df2 <- chamber_of_secrets
df3 <- prisoner_of_azkaban
df4 <- goblet_of_fire
df5 <- order_of_the_phoenix
df6 <- half_blood_prince
df7 <- deathly_hallows


ls(pattern = “df")
[1] "df1" "df2" "df3" "df4" "df5" "df6" “df7"


books <- mget(ls(pattern = "df"))
```

- `ls` will list all objects with the specified pattern in your global environment
- `mget` will get all the objects specified and hold them in a list

TEXT TO TIBBLE

- Let's set up a process to combine these multiple text objects into a tidied tibble

```
titles <- c(
  "Philosopher's Stone", "Chamber of Secrets",
  "Prisoner of Azkaban", "Goblet of Fire",
  "Order of the Phoenix", "Half-Blood Prince",
  "Deathly Hallows"
  )

series <- tibble()
```

- Step 1: create title names in same order as listed objects

- Step 2: create an empty tibble

TEXT TO TIBBLE

- Let's set up a process to combine these multiple text objects into a tidied tibble

```
titles <- c(
  "Philosopher's Stone", "Chamber of Secrets",
  "Prisoner of Azkaban", "Goblet of Fire",
  "Order of the Phoenix", "Half-Blood Prince",
  "Deathly Hallows"
  )

series <- tibble()

for(i in seq_along(books)) {
  org <- tibble(
    book    = titles[i],
    chapter = seq_along(books[[i]]),
    text    = books[[i]]
  )

  series <- rbind(series, org)
}
```

- Step 1: create title names in same order as listed objects

- Step 2: create an empty tibble
- Step 3: loop through the book list and add the title, chapter, and text for each book and…

…add it to the empty tibble

TEXT TO TIBBLE

- Let's set up a process to combine these multiple text objects into a tidied tibble

```
series
# A tibble: 200 x 3
   book                 chapter text
   <chr>                  <int> <chr>
 1 Philosopher's Stone        1 "THE BOY WHO LIVED   Mr. an…
 2 Philosopher's Stone        2 "THE VANISHING GLASS   Near…
 3 Philosopher's Stone        3 "THE LETTERS FROM NO ONE   …
 4 Philosopher's Stone        4 "THE KEEPER OF THE KEYS   B…
 5 Philosopher's Stone        5 "DIAGON ALLEY   Harry woke …
 6 Philosopher's Stone        6 "THE JOURNEY FROM PLATFORM…
 7 Philosopher's Stone        7 "THE SORTING HAT   The door…
 8 Philosopher's Stone        8 "THE POTIONS MASTER   There…
 9 Philosopher's Stone        9 "THE MIDNIGHT DUEL   Harry …
10 Philosopher's Stone       10 "HALLOWEEN   Malfoy couldn'…
# ... with 190 more rows
```

- We end up with a data frame with:
  - Book title
  - Chapter number
  - Chapter text

YOUR TURN!

In your data folder you have three .txt files (amazon_cells_labelled.txt, imdb_labelled.txt, yelp_labelled.txt).

Import these files and get them organized into a data frame that resembles:

```
  file                          text
  <chr>                         <chr>
1 amazon_cells_labelled.txt     So there is no way for me to plug it in here in the US…
2 amazon_cells_labelled.txt     Good case, Excellent value.
3 amazon_cells_labelled.txt     Great for the jawbone.
4 amazon_cells_labelled.txt     Tied to charger for conversations lasting more than 45…
5 amazon_cells_labelled.txt     The mic is great.
```

UNNESTING

Going from paragraph to words

TEXT TO TIBBLE

```
text_tb <- tibble(
  chapter = seq_along(philosophers_stone),
  text = philosophers_stone
)
text_tb
# A tibble: 17 x 2
   chapter text
     <int> <chr>
 1       1 "THE BOY WHO LIVED   Mr. and Mrs. Dursley, of n…
 2       2 "THE VANISHING GLASS   Nearly ten years had pas…
 3       3 "THE LETTERS FROM NO ONE   The escape of the Br…
 4       4 "THE KEEPER OF THE KEYS   BOOM. They knocked ag…
 5       5 "DIAGON ALLEY   Harry woke early the next morni…
 6       6 "THE JOURNEY FROM PLATFORM NINE AND THREE-QUAR…
 7       7 "THE SORTING HAT   The door swung open at once.…
 8       8 "THE POTIONS MASTER   There, look.\"   \"Where?\…
 9       9 "THE MIDNIGHT DUEL   Harry had never believed h…
10      10 "HALLOWEEN   Malfoy couldn't believe his eyes w…
```

We now have our book in a data frame with each chapter text in its own row.

UNNEST OUR TEXT

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

text_tb %>% unnest_tokens(word, text, token = "sentences")

text_tb %>% unnest_tokens(word, text, token = "lines")

text_tb %>% unnest_tokens(word, text, token = "ngrams", n = 2)
# A tibble: 77,858 × 2
   chapter         word
     <int>        <chr>
1        1     the boy
2        1     boy who
3        1   who lived
4        1    lived mr
5        1      mr and
6        1     and mrs
7        1 mrs dursley
```

We can unnest our text by:

- individual words (default)
- sentences (looks for ".")
- lines
- ngrams

Unnest will also:

- standardize to lowercase
- remove non-alphanumeric chars

YOUR TURN!

Using the **`deathly_hallows`** *data, unnest the text by single words and then bi-grams*

STOP WORDS

Where's the content?

STOP WORDS

Word	Word	Word	Word	Word	Word
i	by	they're	his	then	a
yours	after	they'd	them	both	while
herself	off	weren't	these	not	through
which	where	shouldn't	have	ourselves	in
was	other	there's	could	she	here
does	than	if	you've	theirs	few
she's	my	with	you'll	am	own
he'd	yourselves	below	haven't	had	your
isn't	its	under	cannot	i'm	hers
won't	whom	how	where's	they've	what
who's	be	such	because	she'll	are
the	doing	very	against	doesn't	do
at	we're	we	from	mustn't	he's
before	we'd	him	further	how's	you'd
on	wasn't	they	any	until	they'll
when	shan't	that	nor	into	didn't
most	here's	being	ours	down	that's
so	but	should	himself	once	an

- Stop words are common english words
- Typically do not provide us much context

STOP WORDS

```
tidytext::stop_words
# A tibble: 1,149 × 2
          word lexicon
         <chr>   <chr>
1            a   SMART
2          a's   SMART
3         able   SMART
4        about   SMART
5        above   SMART
6    according   SMART
7  accordingly   SMART
8       across   SMART
9     actually   SMART
10       after   SMART
# … with 1,139 more rows
```

- Stop words are common english words
- Typically do not provide us much context
- `tidytext` provides stop words from three lexicons
  - onix
  - SMART
  - snowball

REMOVING STOP WORDS

```
text_tb %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE)
# A tibble: 5,978 x 2
   word      n
   <chr> <int>
 1 the    3629
 2 and    1923
 3 to     1861
 4 a      1691
 5 he     1528
 6 of     1267
 7 harry  1213
 8 was    1186
```

- If we do not remove stop words, any frequency analysis will be dominated by non-contextual words.

REMOVING STOP WORDS

```
text_tb %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)
# A tibble: 5,421 × 2
         word     n
        <chr> <int>
1       harry  1213
2         ron   410
3      hagrid   336
4    hermione   257
5   professor   181
6      looked   169
7       snape   145
8   dumbledore   143
```

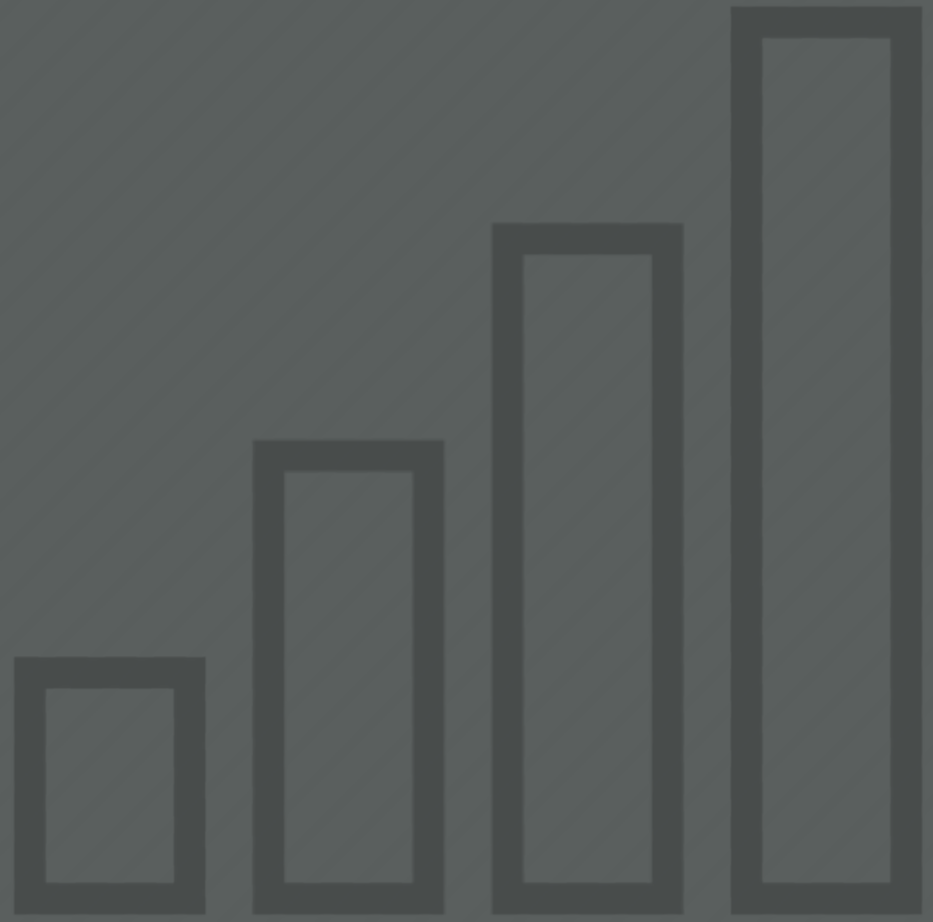
- If we did not remove stop words, any frequency analysis will be dominated by non-contextual words.
- `anti_join` returns all words in our text that are not in the `stop_words` list

YOUR TURN!

Unnest the **`deathly_hallows`** *book and remove the stop words.*

FREQUENCIES

What are the most common words?



WORD FREQUENCY

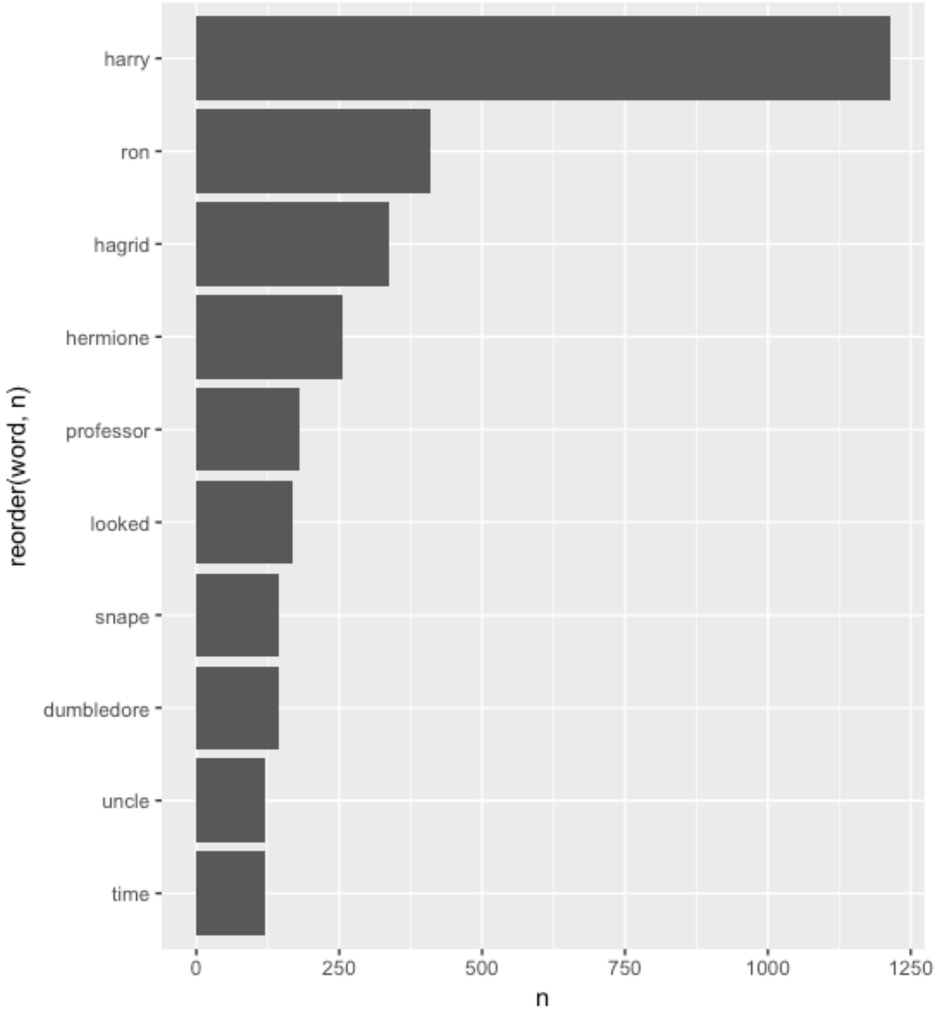
```
text_tb %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  top_n(10)
# A tibble: 10 x 2
        word     n
       <chr> <int>
 1      harry  1213
 2        ron   410
 3     hagrid   336
 4   hermione   257
 5  professor   181
 6     looked   169
 7      angry   145
```

- `count` counts all occurrences of each word (returns **n**).
- `top_n` returns the top **n** words

WORD FREQUENCY

```
text_tb %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  coord_flip()
```

We can pipe this right into ggplot



N-GRAM FREQUENCY

```
text_tb %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  count(bigram, sort = TRUE)
# A tibble: 42,756 x 2
       bigram     n
        <chr> <int>
 1     of the   285
 2     in the   269
 3     on the   218
 4     it was   207
 5     he was   194
 6     to the   173
 7     out of   148
 8     at the   141
 9 said harry   136
```

- We can use the same process to get bi-gram frequencies…
- …but we have a stop word problem again.

How can we remove these?

N-GRAM FREQUENCY

```
text_tb %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ")
# A tibble: 77,858 x 3
   chapter   word1   word2
 *   <int>   <chr>   <chr>
 1       1     the     boy
 2       1     boy     who
 3       1     who   lived
 4       1   lived      mr
 5       1      mr     and
 6       1     and     mrs
 7       1     mrs dursley
 8       1 dursley      of
 9       1      of  number
```

- First, we use **separate** (tidyr package) to separate the bi-gram.

N-GRAM FREQUENCY

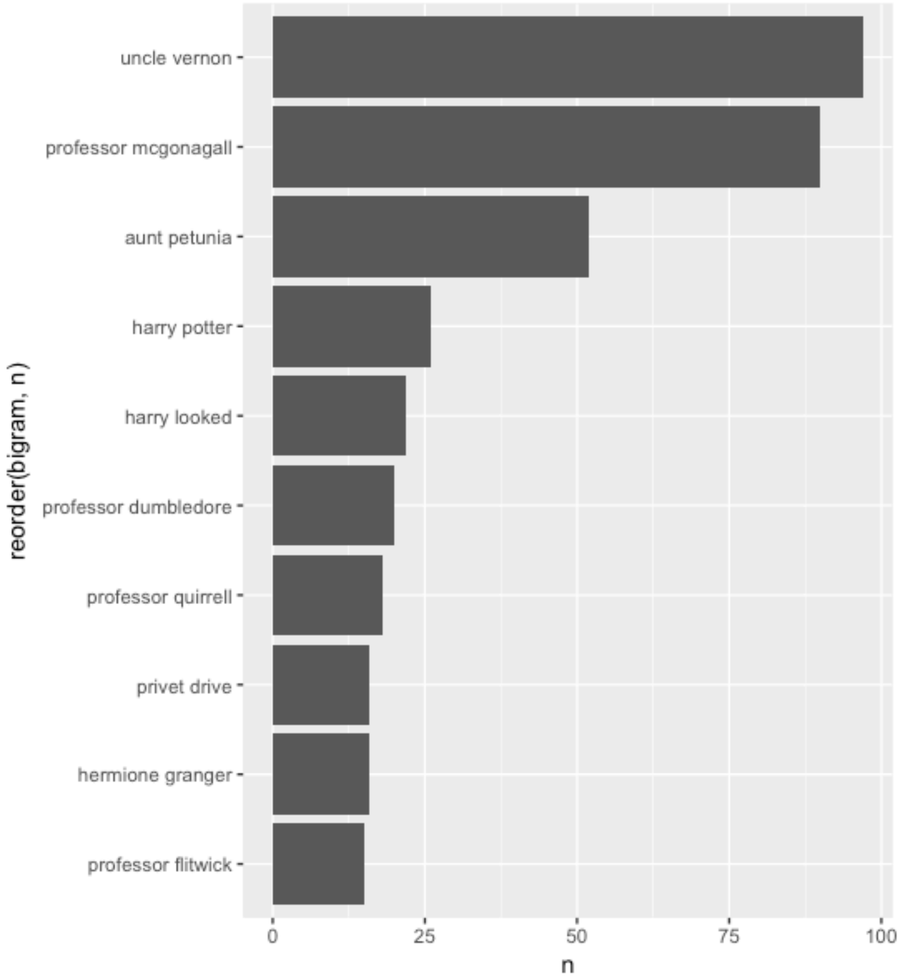
```
text_tb %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(
    !word1 %in% stop_words$word,
    !word2 %in% stop_words$word
  ) %>%
  count(word1, word2, sort = TRUE)
# A tibble: 7,695 x 3
      word1      word2     n
      <chr>      <chr> <int>
 1     uncle     vernon    97
 2 professor mcgonagall    90
 3      aunt    petunia    52
 4     harry     potter    26
```

- First, we use **separate** (tidyr package) to separate the bi-gram.
- Then we can **filter** out stop words from each word variable and
- **count** by word1 & word2 pairs.

N-GRAM FREQUENCY

```
text_tb %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(
    !word1 %in% stop_words$word,
    !word2 %in% stop_words$word
    ) %>%
  count(word1, word2, sort = TRUE) %>%
  unite(bigram, word1, word2, sep = " ") %>%
  top_n(10) %>%
  ggplot(aes(reorder(bigram, n), n)) +
  geom_col() +
  coord_flip()
```

We can pipe this right into ggplot



YOUR TURN!

Find the most common tri-grams in **`deathly_hallows`**
(extra credit for those that can visualize the output).

COMPARISONS

Comparing frequencies across multiple documents

HIGHLY CORRELATED WORDS WITHIN A DOCUMENT

```
ph_cor <- tibble(
  title   = "Philosopher's Stone",
  chapter = seq_along(philosophers_stone),
  text    = philosophers_stone
) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  filter(n() >= 50) %>%
  widyr::pairwise_cor(word, chapter, sort = TRUE)

filter(ph_cor, item1 == "harry")
# A tibble: 5,420 x 3
   item1 item2     correlation
   <chr> <chr>           <dbl>
 1 harry gloom           0.761
 2 harry harry's         0.746
 3 harry paper           0.734
```

- `pairwise_cor` assess words that are most correlated with other words.

COMPARING FREQUENCIES ACROSS BOOKS

```
series %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(book, word, sort = TRUE)
# A tibble: 63,651 x 3
   book                word          n
   <chr>               <chr>     <int>
 1 Order of the Phoenix harry      3730
 2 Goblet of Fire      harry      2936
 3 Deathly Hallows     harry      2770
 4 Half-Blood Prince   harry      2581
 5 Prisoner of Azkaban harry      1824
 6 Chamber of Secrets  harry      1503
 7 Order of the Phoenix hermione  1220
 8 Philosopher's Stone harry      1213
```

- What about comparing frequencies across multiple documents?
- What issue may we run into?

COMPARING FREQUENCIES ACROSS BOOKS

```
books %>%
  map(str_count) %>%
  map_dbl(sum)
    df1     df2     df3     df4     df5     df6     df7   dh_df
 439430  490784  611209 1105151 1514903  984113 1124196 1137387
```

- Total word counts across our documents range from 439,430 to 1,514,903
- Longer books will dominate any word frequency comparisons.
- We can use proportions to compare.

COMPARING TWO DOCUMENTS

```
df1 <- tibble(
  title   = "Philosopher's Stone",
  chapter = seq_along(philosophers_stone),
  text    = philosophers_stone
) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

df2 <- tibble(
  title   = "Deathly Hallows",
  chapter = seq_along(deathly_hallows),
  text    = deathly_hallows
) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)
```

- Step 1: create unnested, clean data for each document

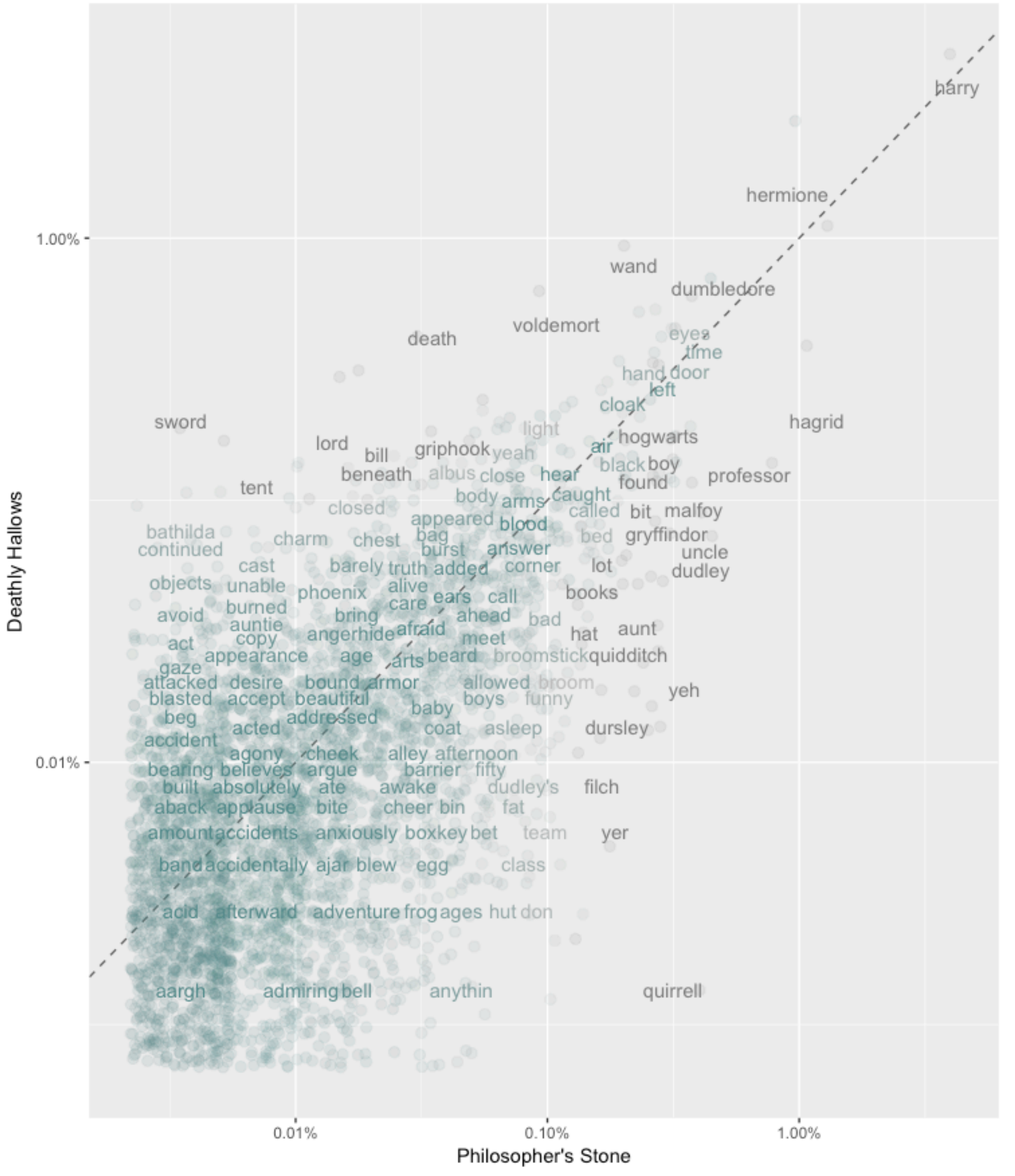
COMPARING TWO DOCUMENTS

```
combined <- df1 %>%
  rbind(df2) %>%
  filter(str_detect(word, "[a-z']+")) %>%
  count(title, word) %>%
  group_by(title) %>%
  mutate(pct = n / sum(n)) %>%
  select(-n) %>%
  spread(title, pct) %>%
  na.omit() %>%
  mutate(delta = abs(`Deathly Hallows` - `Philosopher's Stone`))

combined
# A tibble: 3,880 x 4
   word         `Deathly Hallows` `Philosopher's Stone`     delta
   <chr>                    <dbl>                 <dbl>     <dbl>
 1 aargh                0.0000136             0.0000350 0.0000214
 2 aback                0.0000681             0.0000350 0.0000331
 3 abbott               0.0000681             0.0000350 0.0000331
 4 abnormal             0.0000136             0.0000350 0.0000214
 5 absolutely           0.0000818             0.0000700 0.0000117
 6 absurd               0.0000136             0.0000350 0.0000214
 7 accept               0.000177              0.0000700 0.000107
 8 accepted             0.0000818             0.0000700 0.0000117
 9 accident             0.000123              0.0000350 0.0000876
10 accidentally         0.0000409             0.0000700 0.0000291
# ... with 3,870 more rows
```

- Step 1: create unnested, clean data for each document
- Step 2:
  - rbind data frames together
  - Filter out non-words
  - Count words and compute proportion for each book
  - Spread data

COMPARING TWO DOCUMENTS

```
# we can plot
ggplot(combined, aes(x = `Philosopher's Stone`, y = `Deathly Hallows`, color = delta)) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.2, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE) +
  scale_x_log10("Philosopher's Stone", labels = scales::percent) +
  scale_y_log10("Deathly Hallows", labels = scales::percent) +
  scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
  theme(legend.position = “none")


# or compute correlation
cor.test(combined$`Deathly Hallows`, combined$`Philosopher's Stone`)
      Pearson's product-moment correlation

data:  combined$`Deathly Hallows` and combined$`Philosopher's Stone`
t = 132.6, df = 3878, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8992951 0.9106770
sample estimates:
      cor
0.9051481
```



COMPARING 3+ DOCUMENTS

What about comparing across three or more?

COMPARING 3+ DOCUMENTS

```
# 1. clean
clean_tokens <- series %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

clean_tokens
# A tibble: 409,338 x 3
   book                chapter word
   <chr>                 <int> <chr>
 1 Philosopher's Stone       1 boy
 2 Philosopher's Stone       1 lived
 3 Philosopher's Stone       1 dursley
 4 Philosopher's Stone       1 privet
 5 Philosopher's Stone       1 drive
 6 Philosopher's Stone       1 proud
 7 Philosopher's Stone       1 perfectly
 8 Philosopher's Stone       1 normal
 9 Philosopher's Stone       1 people
10 Philosopher's Stone       1 expect
# ... with 409,328 more rows
```

- Step 1: Unnest and remove stop words.

COMPARING 3+ DOCUMENTS

```
# 1. clean
clean_tokens <- series %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

# 2. compute percent of word use across all novels
series_pct <- clean_tokens %>%
  count(word, sort = TRUE) %>%
  transmute(word, all_words = n / sum(n))

series_pct
# A tibble: 23,795 x 2
   word       all_words
   <chr>          <dbl>
 1 harry        0.0404
 2 ron          0.0140
 3 hermione     0.0120
 4 dumbledore   0.00702
 5 looked       0.00573
 6 professor    0.00490
 7 hagrid       0.00423
 8 time         0.00418
 9 wand         0.00400
10 eyes         0.00392
# ... with 23,785 more rows
```

- Step 1: Unnest and remove stop words.
- Step 2: Compute % of word use across series

COMPARING 3+ DOCUMENTS

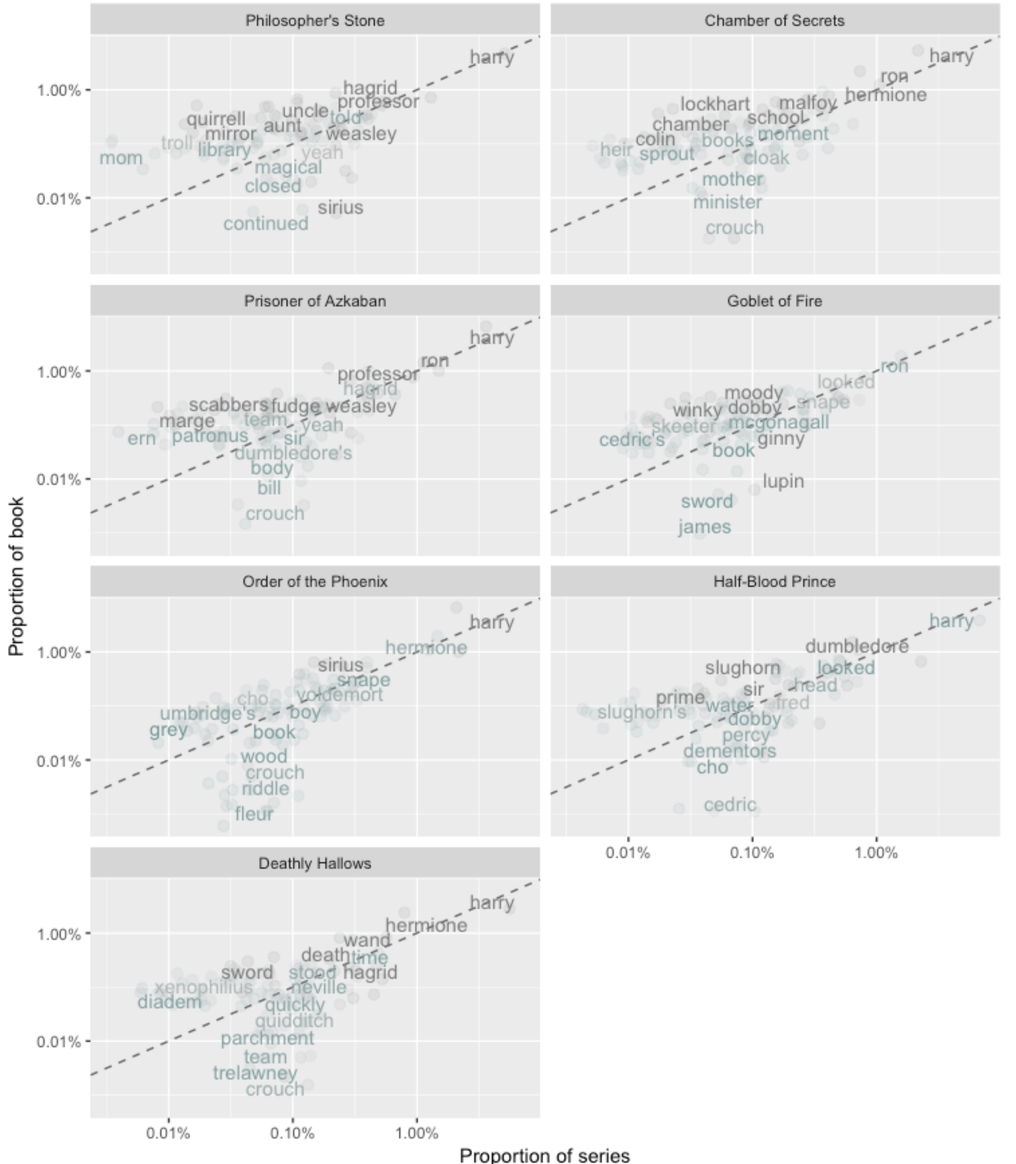
```
# 1. clean
clean_tokens <- series %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

# 2. compute percent of word use across all novels
series_pct <- clean_tokens %>%
  count(word, sort = TRUE) %>%
  transmute(word, all_words = n / sum(n))

# 3. compute percent of word use within each novel
comp_prop <- clean_tokens %>%
  count(book, word, sort = TRUE) %>%
  group_by(book) %>%
  mutate(book_words = n / sum(n)) %>%
  inner_join(series_pct) %>%
  ungroup()
comp_prop
# A tibble: 63,651 x 5
   book                word         n book_words all_words
   <chr>               <chr>    <int>      <dbl>     <dbl>
 1 Order of the Phoenix harry     3730     0.0385    0.0404
 2 Goblet of Fire       harry     2936     0.0404    0.0404
 3 Deathly Hallows      harry     2770     0.0377    0.0404
 4 Half-Blood Prince    harry     2581     0.0409    0.0404
```

- Step 1: Unnest and remove stop words.
- Step 2: Compute % of word use across series
- Step 3: Compute % of word use within each novel & join series %.

Now what can we do?

COMPARING 3+ DOCUMENTS

```
# option 1:  Plot
comp_prop %>%
  mutate(
    delta = abs(all_words - book_words),
    book = factor(book, levels = titles)
    ) %>%
  group_by(book) %>%
  top_n(100, wt = delta) %>%
  ggplot(aes(x = all_words, y = book_words, color = delta)) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE) +
  scale_x_log10("Proportion of series", labels = scales::percent) +
  scale_y_log10("Proportion of book", labels = scales::percent) +
  scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
  facet_wrap(~ book, ncol = 2) +
  theme(legend.position = "none")
```



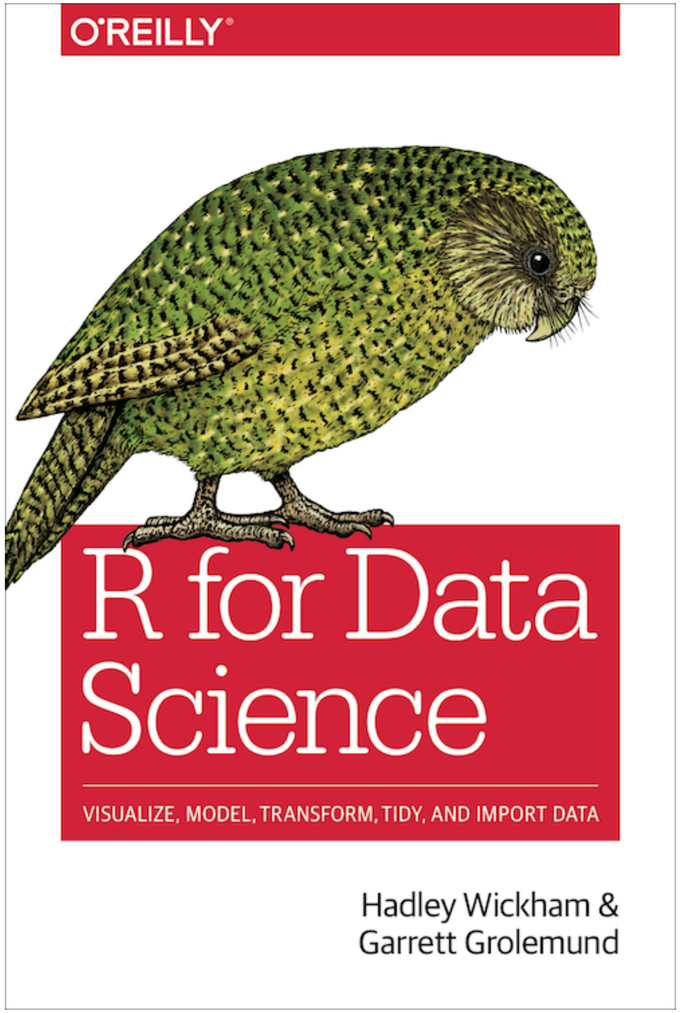
COMPARING 3+ DOCUMENTS

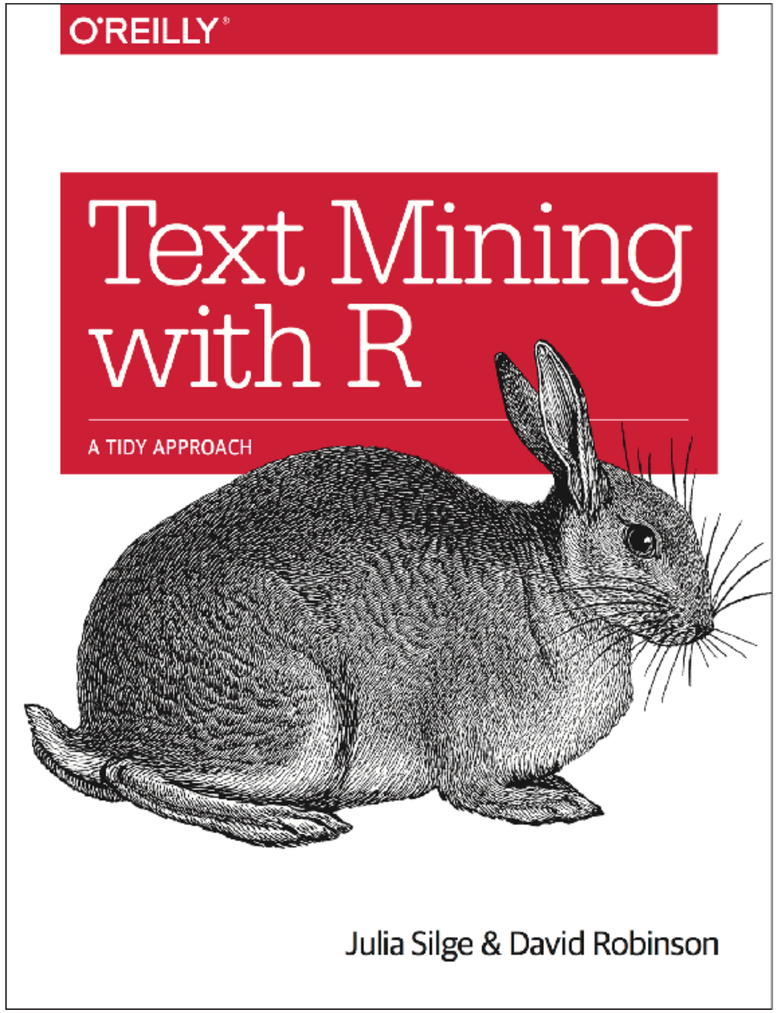
```
# option 2:  Compute correlation of each book to the overall series
comp_prop %>%
  group_by(book) %>%
  summarise(rho = cor(book_words, all_words)) %>%
  arrange(desc(rho))
# A tibble: 7 x 2
  book                   rho
  <chr>                <dbl>
1 Order of the Phoenix 0.984
2 Goblet of Fire       0.979
3 Deathly Hallows      0.970
4 Half-Blood Prince    0.970
5 Chamber of Secrets   0.966
6 Prisoner of Azkaban  0.964
7 Philosopher's Stone  0.955
```

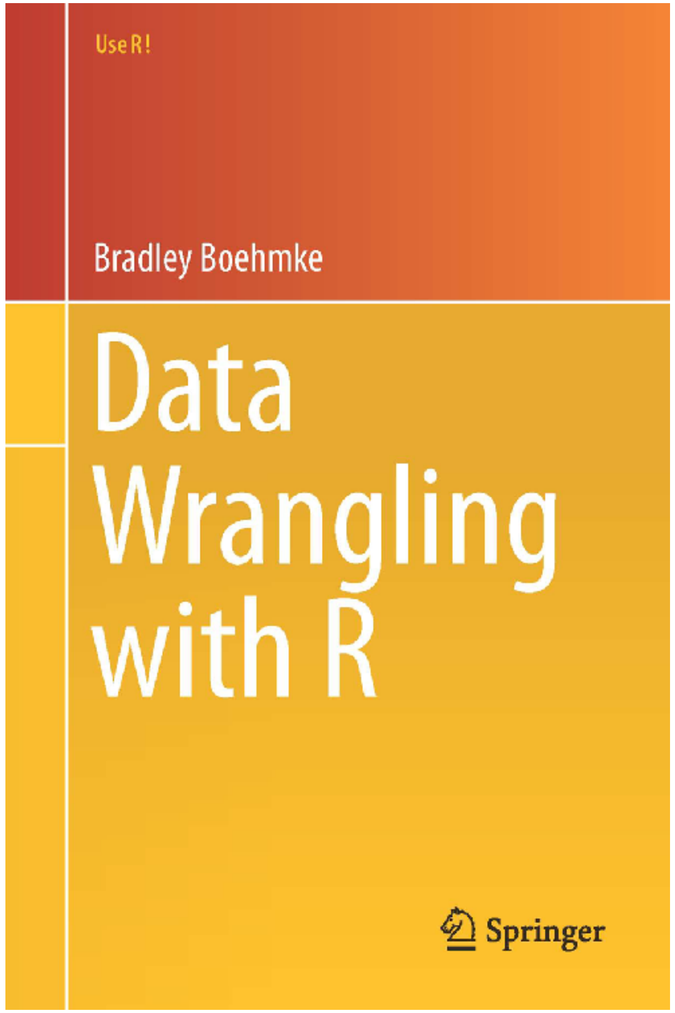
- Philosopher's stone is least representative of the overall series.


SO LITTLE TIME!

LEARN MORE


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund


O'REILLY
Text Mining with R
A TIDY APPROACH
Julia Silge & David Robinson


Use R!
Bradley Boehmke
Data Wrangling with R
Springer

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
`tibble`	Convert character strings into a data frame (tibble)
`unnest_tokens`	Unnest and clean text
`stop_words`	Commonly used English words that lack context
`anti_join, filter, count`	dplyr functions commonly used