ETC5510: Introduction to Data Analysis

Week 8, part A

Text analysis

Lecturer: *Nicholas Tierney & Stuart Lee*
Department of Econometrics and Business Statistics
ETC5510.Clayton-x@monash.edu

May 2020

recap

- linear models (are awesome)
- many models

Announcements

- Assignment
- Project
- Peer review marking

Why text analysis?

- Predict Melbourne house prices from realtor descriptions
- Determine the extent of public discontent with train stoppages in Melbourne
- The differences between Darwin's first edition of the Origin of the Species and the 6th edition
- Does the sentiment of posts on Newcastle Jets public facebook page reflect their win/loss record?

Typical Process

1. Read in text
2. Pre-processing: remove punctuation signs, remove numbers, stop words, stem words
3. Tokenise: words, sentences, ngrams, chapters
4. Summarise
5. model

Packages

In addition to `tidyverse` we will be using three other packages today

```
library(tidytext)
library(gutenbergr)
```

Tidytext

- Using tidy data principles can make many text mining tasks easier, more effective, and consistent with tools already in wide use.
- Learn more at https://www.tidytextmining.com/, by Julia Silge and David Robinson.

What is tidy text?

```r
text <- c("This will be an uncertain time for us my love",
          "I can hear the echo of your voice in my head",
          "Singing my love",
          "I can see your face there in my hands my love",
          "I have been blessed by your grace and care my love",
          "Singing my love")

text
## [1] "This will be an uncertain time for us my love"
## [2] "I can hear the echo of your voice in my head"
## [3] "Singing my love"
## [4] "I can see your face there in my hands my love"
## [5] "I have been blessed by your grace and care my love"
## [6] "Singing my love"
```

What is tidy text?

```
text_df <- tibble(line = seq_along(text), text = text)

text_df
## # A tibble: 6 x 2
##    line text
##   <int> <chr>
## 1     1 This will be an uncertain time for us my love
## 2     2 I can hear the echo of your voice in my head
## 3     3 Singing my love
## 4     4 I can see your face there in my hands my love
## 5     5 I have been blessed by your grace and care my love
## 6     6 Singing my love
```

What is tidy text?

```r
text_df %>%
  unnest_tokens(output = word,
                input = text,
                token = "words")
```

```
## # A tibble: 49 x 2
##      line word
##     <int> <chr>
##  1      1 this
##  2      1 will
##  3      1 be
##  4      1 an
##  5      1 uncertain
##  6      1 time
##  7      1 for
##  8      1 us
##  9      1 my
## 10      1 love
## # … with 39 more rows
```

What is unnesting?

```r
text_df %>%
  unnest_tokens(output = word,
                input = text,
                token = "charact
```

```
## # A tibble: 171 x 2
##       line word
##      <int> <chr>
##  1       1 t
##  2       1 h
##  3       1 i
##  4       1 s
##  5       1 w
##  6       1 i
##  7       1 l
##  8       1 l
##  9       1 b
## 10       1 e
## # … with 161 more rows
```

What is unnesting - ngrams length 2

```
text_df %>%
  unnest_tokens(output = word,
                input = text,
                token = "ngrams'
                n = 2)
```

```
## # A tibble: 43 x 2
##      line word
##     <int> <chr>
##  1      1 this will
##  2      1 will be
##  3      1 be an
##  4      1 an uncertain
##  5      1 uncertain time
##  6      1 time for
##  7      1 for us
##  8      1 us my
##  9      1 my love
## 10      2 i can
## # … with 33 more rows
```

What is unnesting - ngrams length 3

```
text_df %>%
  unnest_tokens(output = word,
                input = text,
                token = "ngrams",
                n = 3)
```

```
## # A tibble: 37 x 2
##      line word
##     <int> <chr>
##  1      1 this will be
##  2      1 will be an
##  3      1 be an uncertain
##  4      1 an uncertain time
##  5      1 uncertain time for
##  6      1 time for us
##  7      1 for us my
##  8      1 us my love
##  9      2 i can hear
## 10      2 can hear the
## # … with 27 more rows
```

Your Turn:

Complete "8a-tokenizing.Rmd"

Analyzing user reviews for Animal Crossing: New Horizons

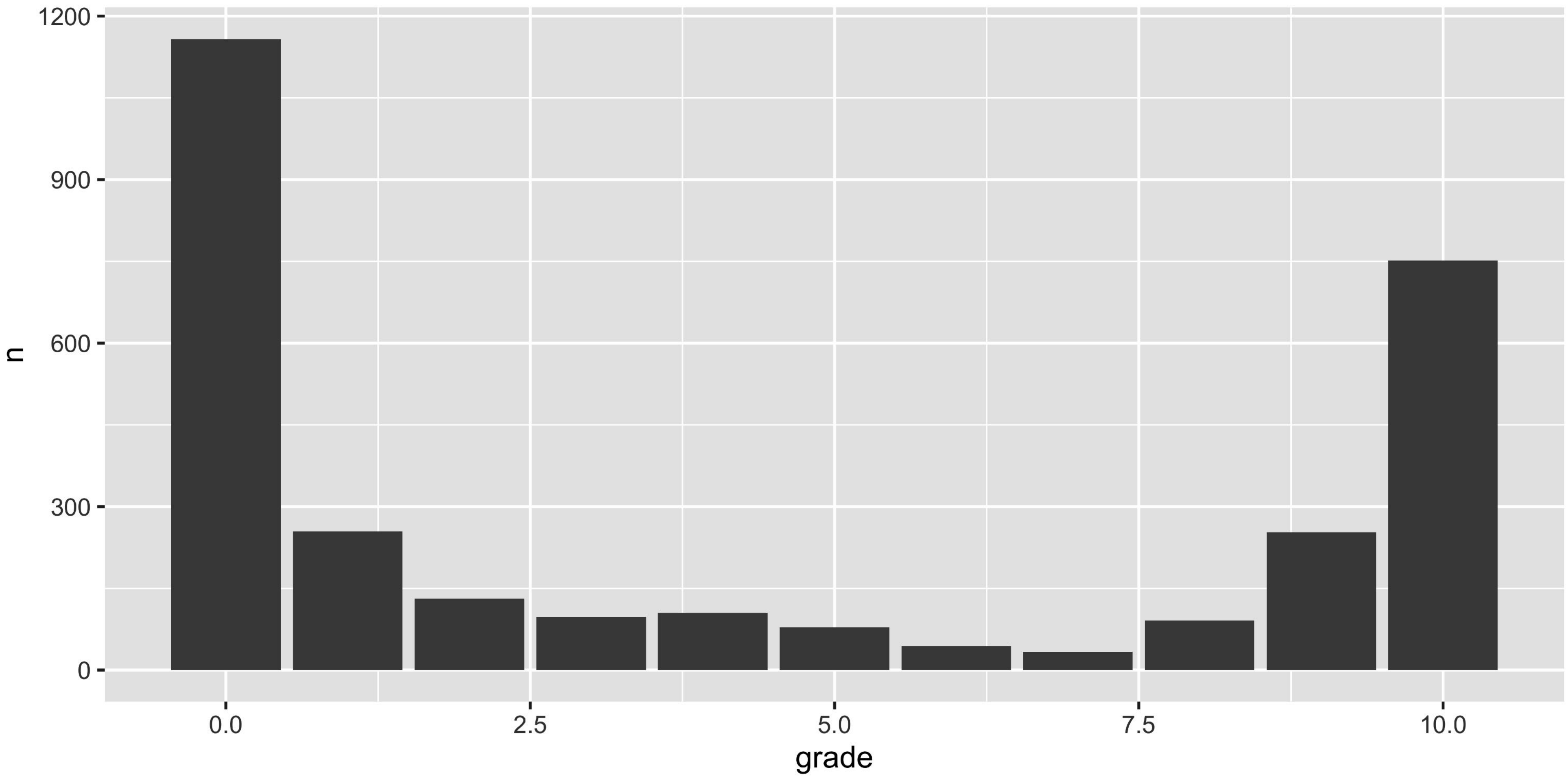
About the data

- User and critic reviews for the game Animal Crossing scraped from Metacritc
- This data comes from a #TidyTuesday challenge.

What do the user reviews look like?

```
acnh_user_reviews <- read_tsv(here::here("slides/data/acnh_user_reviews.tsv"))
glimpse(acnh_user_reviews)
## Rows: 2,999
## Columns: 4
## $ grade     <dbl> 4, 5, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## $ user_name <chr> "mds27272", "lolo2178", "Roachant", "Houndf", "ProfessorFox", "tb
## $ text      <chr> "My gf started playing before me. No option to create my own isla
## $ date      <date> 2020-03-20, 2020-03-20, 2020-03-20, 2020-03-20, 2020-03-20, 2020
```

Let's look at the grade distrubtion



Read a few of the positive reviews

```
set.seed(1999)
acnh_user_reviews %>%
  filter(grade > 8) %>%
  sample_n(3) %>%
  pull(text)
```

```
## [1] "The game is absolutely fantastic and everything we have been waiting for for s
## [2] "I've never played an Animal Crossing before and I can't stop playing it now! A
## [3] "This review contains spoilers, click expand to view.
```

And some negative reviews

```
set.seed(2099)
acnh_user_reviews %>%
  filter(grade == 0) %>%
  sample_n(3) %>%
  pull(text)
```

```
## [1] "It's just a typical mobile grind-game with time-wall. And poor interface. Wors
## [2] "One island per console, very family friendly. How unbelievably greedy by Ninte
## [3] "Separate islands for each profile should be a given. I'm not buying another sw
```

Looks like the scraping is messed up a bit

Long reviews are compressed from the scraping procedure...

```
acnh_user_reviews_parsed <- acnh_user_reviews %>%
  mutate(text = str_remove(text, "Expand$"))
```

We will remove these characters from the text..

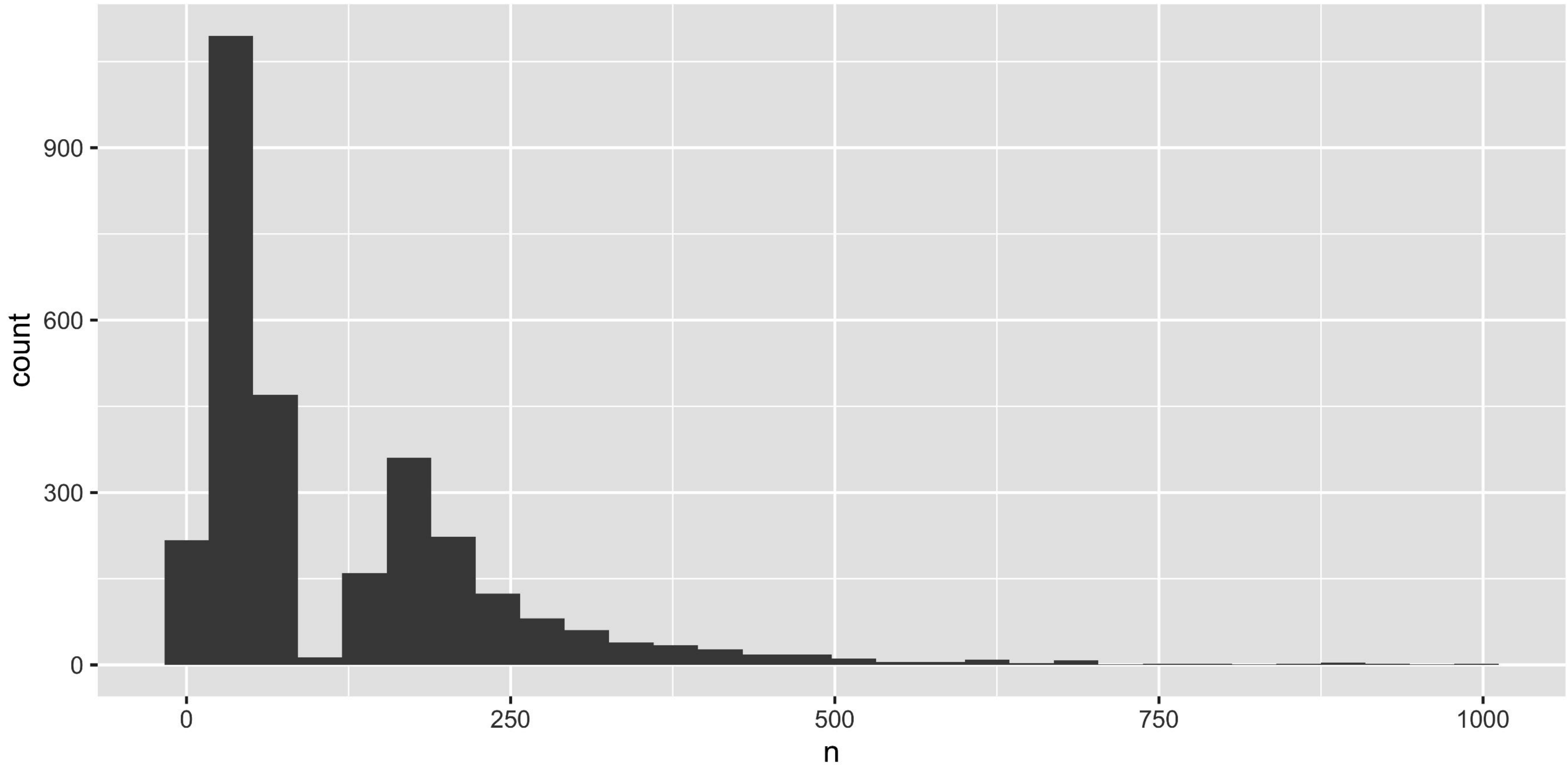
Tidy up the reviews!

```
user_reviews_words <- acnh_user_reviews_parsed %>%
  unnest_tokens(output = word, input = text)

user_reviews_words
## # A tibble: 362,729 x 4
##    grade user_name date       word
##    <dbl> <chr>     <date>     <chr>
##  1     4 mds27272  2020-03-20 my
##  2     4 mds27272  2020-03-20 gf
##  3     4 mds27272  2020-03-20 started
##  4     4 mds27272  2020-03-20 playing
##  5     4 mds27272  2020-03-20 before
##  6     4 mds27272  2020-03-20 me
##  7     4 mds27272  2020-03-20 no
##  8     4 mds27272  2020-03-20 option
##  9     4 mds27272  2020-03-20 to
## 10     4 mds27272  2020-03-20 create
## # … with 362,719 more rows
```

Distribution of words per review?

```
user_reviews_words %>%
  count(user_name) %>%
  ggplot(aes(x = n)) +
  geom_histogram()
```



What are the most common words?

```
user_reviews_words %>%
  count(word, sort = TRUE)
## # A tibble: 13,454 x 2
##    word      n
##    <chr> <int>
##  1 the   17739
##  2 to    11857
##  3 game   8769
##  4 and    8740
##  5 a      8330
##  6 i      7211
##  7 is     6858
##  8 this   5777
##  9 of     5383
## 10 it     4711
## # … with 13,444 more rows
```

Stop words

- In computing, stop words are words which are filtered out before or after processing of natural language data (text).
- They usually refer to the most common words in a language, but there is not a single list of stop words used by all natural language processing tools.

English stop words

```
get_stopwords()
## # A tibble: 175 x 2
##    word      lexicon
##    <chr>     <chr>
##  1 i         snowball
##  2 me        snowball
##  3 my        snowball
##  4 myself    snowball
##  5 we        snowball
##  6 our       snowball
##  7 ours      snowball
##  8 ourselves snowball
##  9 you       snowball
## 10 your      snowball
## # … with 165 more rows
```

Spanish stop words

```
get_stopwords(language = "es")
## # A tibble: 308 x 2
##    word  lexicon
##    <chr> <chr>
##  1 de    snowball
##  2 la    snowball
##  3 que   snowball
##  4 el    snowball
##  5 en    snowball
##  6 y     snowball
##  7 a     snowball
##  8 los   snowball
##  9 del   snowball
## 10 se    snowball
## # … with 298 more rows
```

Various lexicons

See `?get_stopwords` for more info.

```
get_stopwords(source = "smart")
## # A tibble: 571 x 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 561 more rows
```

What are the most common words?

```
user_reviews_words %>%
  count(word, sort = TRUE)
## # A tibble: 13,454 x 2
##    word      n
##    <chr> <int>
##  1 the   17739
##  2 to    11857
##  3 game   8769
##  4 and    8740
##  5 a      8330
##  6 i      7211
##  7 is     6858
##  8 this   5777
##  9 of     5383
## 10 it     4711
## # … with 13,444 more rows
```

What are the most common words?

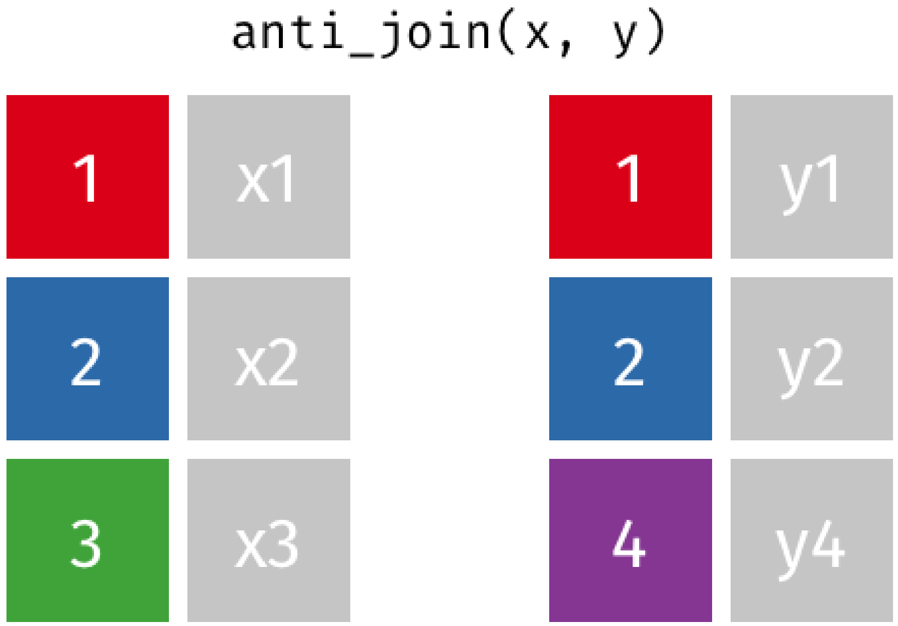
```
stopwords_smart <- get_stopwords(source = "smart")

user_reviews_words %>%
  anti_join(stopwords_smart)
## # A tibble: 145,444 x 4
##    grade user_name date       word
##    <dbl> <chr>     <date>     <chr>
##  1     4 mds27272  2020-03-20 gf
##  2     4 mds27272  2020-03-20 started
##  3     4 mds27272  2020-03-20 playing
##  4     4 mds27272  2020-03-20 option
##  5     4 mds27272  2020-03-20 create
##  6     4 mds27272  2020-03-20 island
##  7     4 mds27272  2020-03-20 guys
##  8     4 mds27272  2020-03-20 2nd
##  9     4 mds27272  2020-03-20 player
## 10     4 mds27272  2020-03-20 start
## # … with 145,434 more rows
```

Aside: the anti-join

- A type of filtering join, will return all rows on the left when there are no matches on the right
- Only keeps columns on the left

As a picture



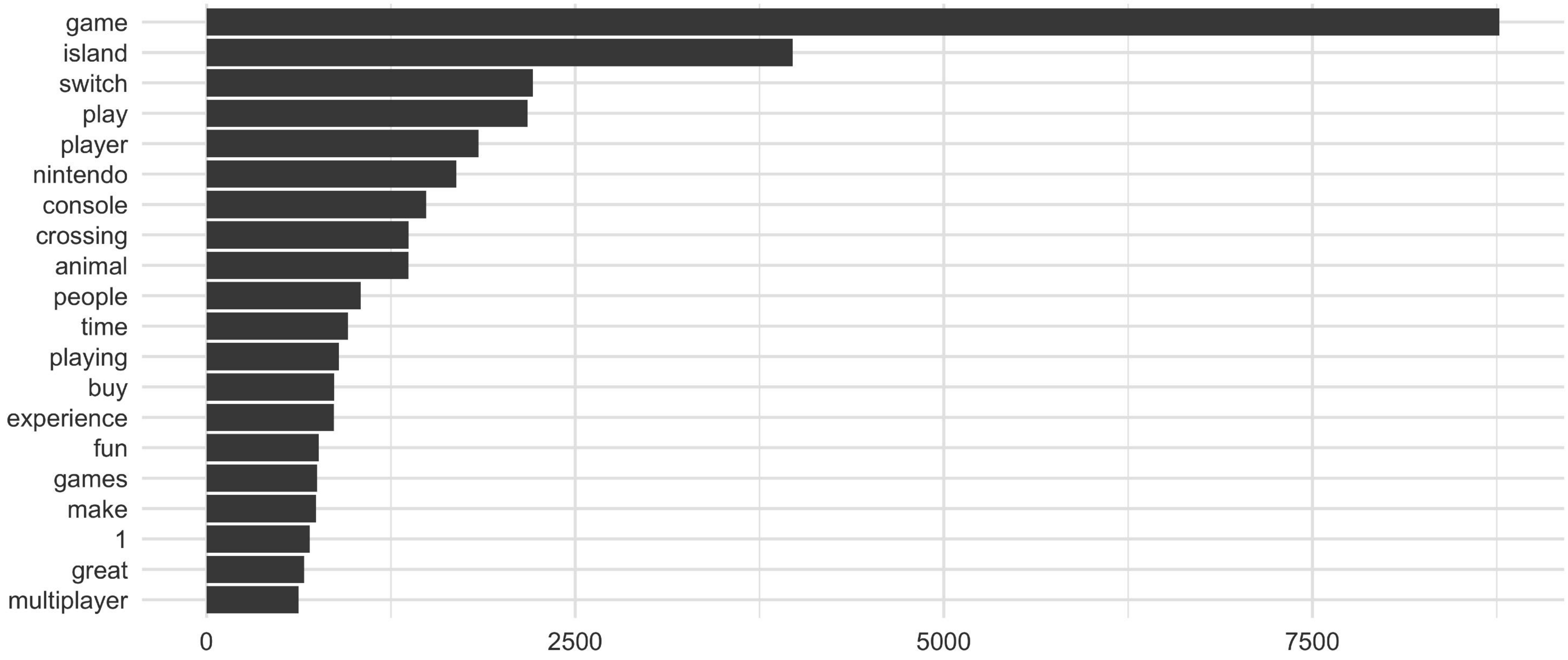
What are the most common words?

```
user_reviews_words %>%
  anti_join(stopwords_smart) %>%
  count(word, sort = TRUE)
## # A tibble: 12,938 x 2
##    word         n
##    <chr>    <int>
##  1 game      8769
##  2 island    3974
##  3 switch    2214
##  4 play      2176
##  5 player    1844
##  6 nintendo  1694
##  7 console   1489
##  8 crossing  1371
##  9 animal    1369
## 10 people    1045
## # … with 12,928 more rows
```

What are the most common words?

```r
user_reviews_words %>%
  anti_join(stopwords_smart) %>%
  count(word) %>%
  arrange(-n) %>%
  top_n(20) %>%
  ggplot(aes(fct_reorder(word, n), n)) +
  geom_col() +
  coord_flip() +
  theme_minimal() +
  labs(title = "Frequency of words in user reviews",
       subtitle = "",
       y = "",
       x = "")
```

Frequency of words in user reviews



Your turn:

Complete "8a-stopwords.Rmd"

Sentiment Analysis

Sentiment analysis

- One way to analyze the sentiment of a text is to consider the text as a combination of its individual words
- and the sentiment content of the whole text as the sum of the sentiment content of the individual words

Sentiment lexicons

```
get_sentiments("afinn")
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  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 2,467 more rows
```

```
get_sentiments("bing")
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # … with 6,776 more rows
```

Sentiment lexicons

```
get_sentiments(lexicon = "bing")
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # … with 6,776 more rows
```

```
get_sentiments(lexicon = "loughran")
## # A tibble: 4,150 x 2
##    word         sentiment
##    <chr>        <chr>
##  1 abandon      negative
##  2 abandoned    negative
##  3 abandoning   negative
##  4 abandonment  negative
##  5 abandonments negative
##  6 abandons     negative
##  7 abdicated    negative
##  8 abdicates    negative
##  9 abdicating   negative
## 10 abdication   negative
## # … with 4,140 more rows
```

Sentiments in the reviews

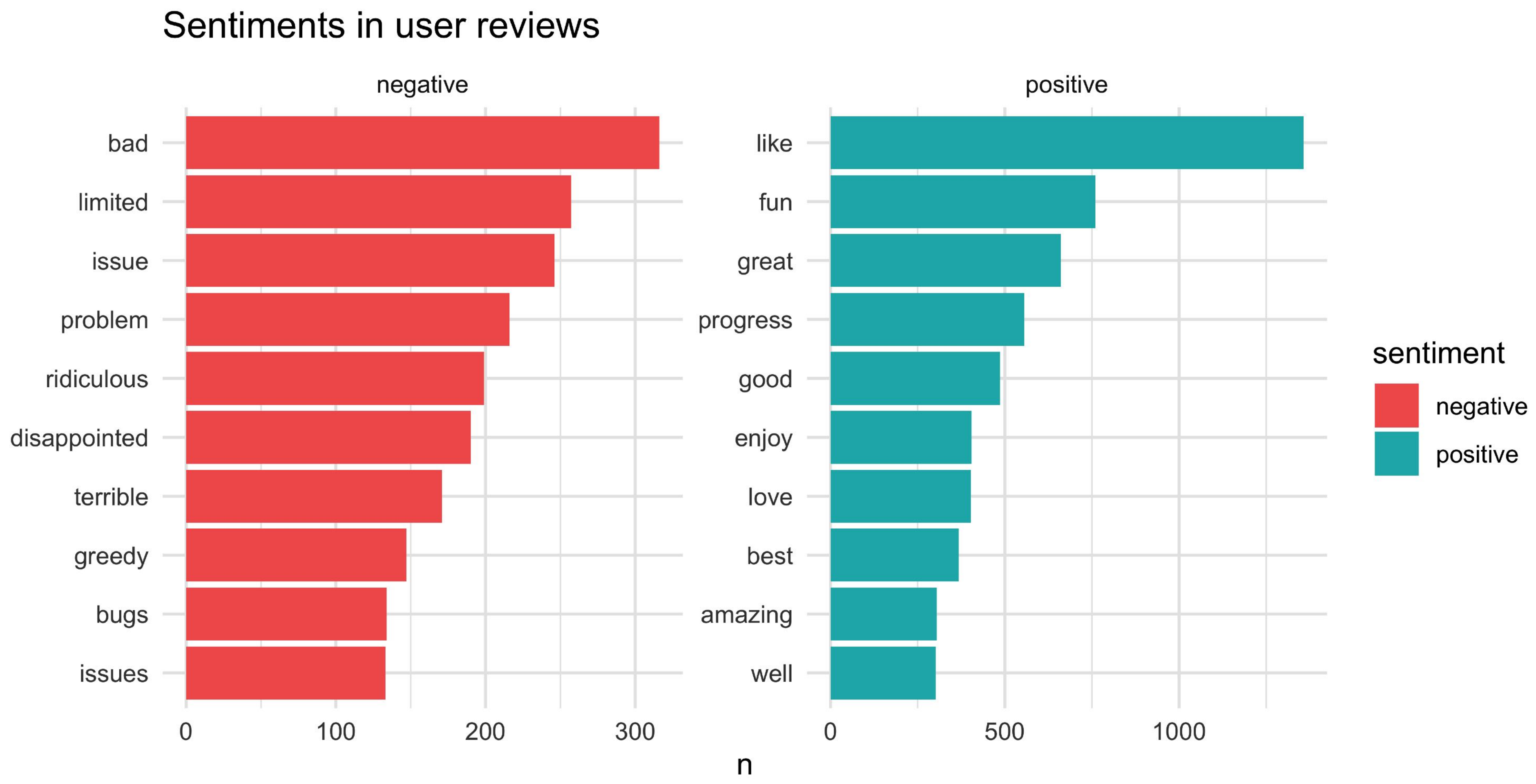
```
sentiments_bing <- get_sentiments("bing")

user_reviews_words %>%
  inner_join(sentiments_bing) %>%
  count(sentiment, word, sort = TRUE)
## # A tibble: 1,622 x 3
##    sentiment word         n
##    <chr>     <chr>    <int>
##  1 positive  like      1357
##  2 positive  fun        760
##  3 positive  great      661
##  4 positive  progress   556
##  5 positive  good       486
##  6 positive  enjoy      405
##  7 positive  love       403
##  8 positive  best       368
##  9 negative  bad        316
## 10 positive  amazing    304
## # … with 1,612 more rows
```

Visualising sentiments

```
user_reviews_words %>%
  inner_join(sentiments_bing) %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  ggplot(aes(fct_reorder(word, n), n, fill = sentiment)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~sentiment, scales = "free") +
  theme_minimal() +
  labs(
    title = "Sentiments in user reviews",
    x = ""
  )
```

Visualising sentiments



Common words over grades

```
user_reviews_words %>%
  anti_join(stopwords_smart) %>%
  count(grade, word, sort = TRUE)
## # A tibble: 29,010 x 3
##    grade word         n
##    <dbl> <chr>    <int>
##  1     0 game      2989
##  2     0 island    1783
##  3    10 game      1685
##  4     0 switch    1058
##  5     0 play      1027
##  6     0 player     921
##  7     0 nintendo   902
##  8     1 game       898
##  9     9 game       802
## 10     0 console    738
## # … with 29,000 more rows
```

Common review words by grade - With stop words:

```
user_reviews_words %>%
  count(grade, word, sort = TRUE)
## # A tibble: 33,237 x 3
##    grade word      n
##    <dbl> <chr> <int>
##  1     0 the    5865
##  2     0 to     4421
##  3     0 game   2989
##  4    10 the    2896
##  5     0 and    2767
##  6     0 a      2663
##  7     0 i      2656
##  8     0 is     2304
##  9     0 this   2171
## 10     9 the    1913
## # … with 33,227 more rows
```

Your turn:

Complete "8a-sentiment.Rmd"

What is a document about?

How do we measure the importance of a word to a document in a collection of documents?

i.e a novel in a collection of novels or a review in a set of reviews...

We combine the following statistics:

- Term frequency
- Inverse document frequency

Term frequency

The raw frequency of a word w in a document d. It is a function of the word and the document.

$$tf(w, d) = \frac{\text{count of w in d}}{\text{total count in d}}$$

Term frequency

For our reviews a document is a single user's review.

```
document <- user_reviews_words %>%
    anti_join(stopwords_smart) %>%
    filter(user_name == "Discoduckasaur")
document
## # A tibble: 372 x 4
##    grade user_name      date       word
##    <dbl> <chr>          <date>     <chr>
##  1     4 Discoduckasaur 2020-04-23 start
##  2     4 Discoduckasaur 2020-04-23 game
##  3     4 Discoduckasaur 2020-04-23 incredibly
##  4     4 Discoduckasaur 2020-04-23 fun
##  5     4 Discoduckasaur 2020-04-23 base
##  6     4 Discoduckasaur 2020-04-23 asinine
##  7     4 Discoduckasaur 2020-04-23 decisions
##  8     4 Discoduckasaur 2020-04-23 made
##  9     4 Discoduckasaur 2020-04-23 nintendo
## 10     4 Discoduckasaur 2020-04-23 simply
## # … with 362 more rows
```

Term frequency

The term frequency for each word is the number of times that word occurs divided by the total number of words in the document.

```
tbl_tf <- document %>%
  count(word, sort = TRUE) %>%
  mutate(tf = n / sum(n))
```

```
tbl_tf %>%
  arrange(desc(tf))
## # A tibble: 246 x 3
##    word           n     tf
##    <chr>      <int>  <dbl>
##  1 game          15 0.0403
##  2 time          14 0.0376
##  3 it's           7 0.0188
##  4 nintendo       6 0.0161
##  5 i'm            5 0.0134
##  6 bad            4 0.0108
##  7 fun            4 0.0108
##  8 give           4 0.0108
##  9 incredibly     4 0.0108
## 10 means          4 0.0108
## # … with 236 more rows
```

Inverse-document frequency

The inverse document frequency tells how common or rare a word is accross a collection of documents. It is a function of a word w, and the collection of documents D.

$$idf(w, D) = \log\left(\frac{\text{size of } D}{\text{number of documents that contain } w}\right)$$

Inverse document frequency

For the reviews data set, our collection is all the reviews. You could compute this in a somewhat roundabout as follows:

```
tbl_idf <- user_reviews_words %>%
    anti_join(stopwords_smart) %>%
    mutate(collection_size = n_distinct(user_name)) %>%
    group_by(collection_size, word) %>%
    summarise(times_word_used = n_distinct(user_name)) %>%
    mutate(freq = collection_size / times_word_used,
           idf = log(freq))
arrange(tbl_idf, idf)
## # A tibble: 12,938 x 5
## # Groups:   collection_size [1]
##    collection_size word     times_word_used  freq   idf
##              <int> <chr>              <int> <dbl> <dbl>
##  1            2999 game                2354  1.27 0.242
##  2            2999 island              1671  1.79 0.585
##  3            2999 switch              1123  2.67 0.982
##  4            2999 play                1049  2.86 1.05
```

Putting it together term frequency, inverse document frequency

Multiply tf and idf together. This is a function of a word w, a document d, and the collection of documents $\mathcal{D}$:

$$tf_idf(w, d, \mathcal{D}) = tf(w, d) \times idf(w, \mathcal{D})$$

High value of tf_idf that a word has a high frequency within a document but is quite rare over all documents. Likewise if a word occurs in a lot of documents idf will be close to zero, so tf_idf will be small.

Putting it together, tf-idf

We illustrate the example for a single user review:

```
tbl_tf %>%
    left_join(tbl_idf) %>%
    select(word, tf, idf) %>%
    mutate(tf_idf = tf * idf) %>%
    arrange(desc(tf_idf))
## # A tibble: 246 x 4
##    word              tf   idf tf_idf
##    <chr>          <dbl> <dbl>  <dbl>
##  1 turnips      0.0108   6.62 0.0712
##  2 time         0.0376   1.78 0.0669
##  3 zone         0.00806  7.31 0.0590
##  4 it's         0.0188   2.81 0.0528
##  5 i'm          0.0134   3.64 0.0489
##  6 you'll       0.00806  5.70 0.0460
##  7 incredibly 0.0108   4.05 0.0436
##  8 hurting      0.00538  8.01 0.0430
##  9 means        0.0108   3.62 0.0390
```

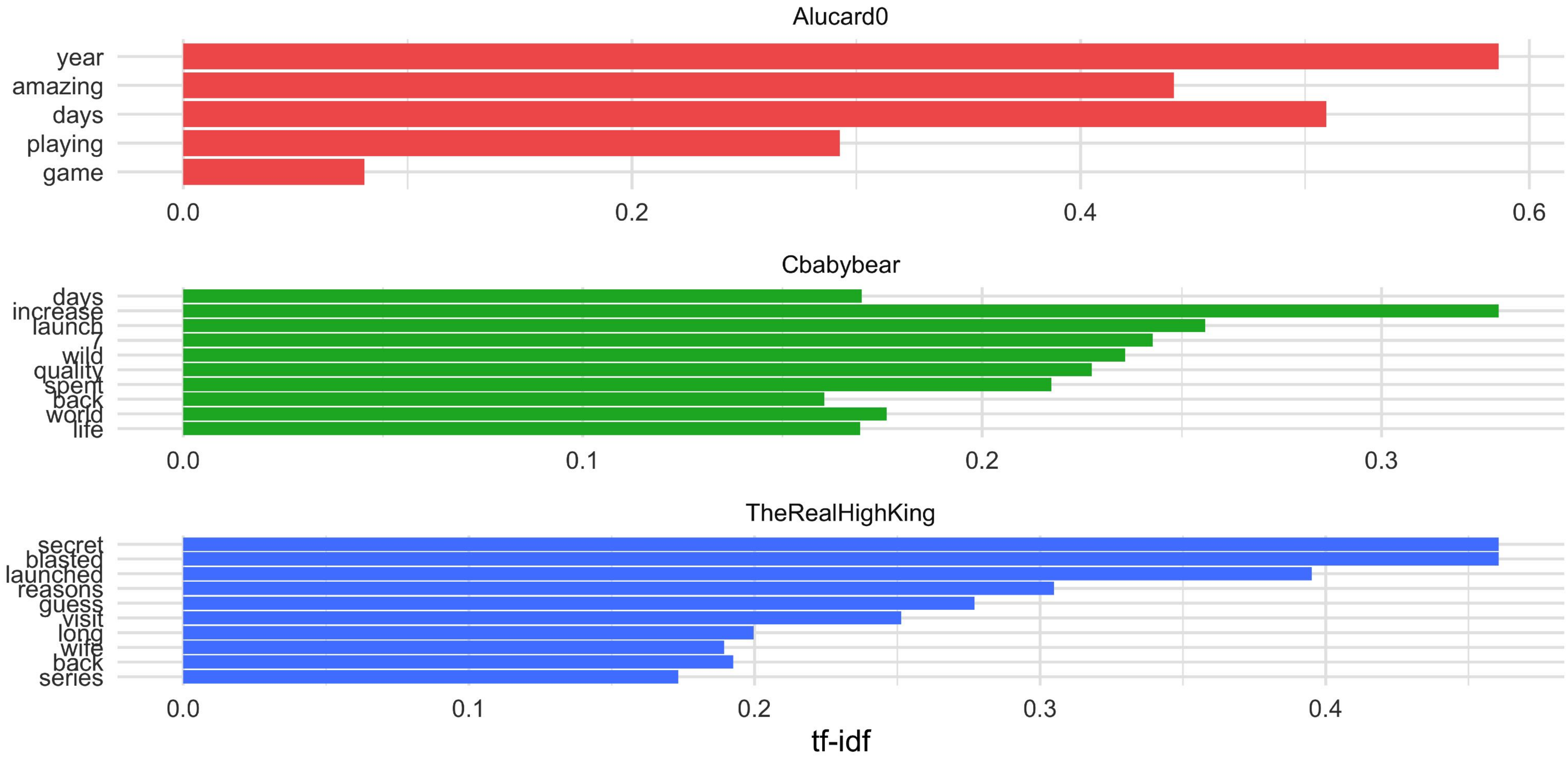
Calculating tf-idf: Perhaps not that exciting...

Instead of rolling our own, we can use `tidytext`

```
user_reviews_counts <- user_reviews_words %>%
      anti_join(stopwords_smart) %>%
      count(user_name, word, sort = TRUE) %>%
      bind_tf_idf(term = word, document = user_name, n = n)

user_reviews_counts
## # A tibble: 93,540 x 6
##    user_name    word         n     tf   idf tf_idf
##    <chr>        <chr>    <int>  <dbl> <dbl>  <dbl>
##  1 Melondrea    island      49 0.5    0.585 0.292
##  2 Melondrea    console     48 0.490  1.34  0.655
##  3 ScissorSheep game        29 0.254  0.242 0.0616
##  4 ScissorSheep bombing     28 0.246  3.62  0.890
##  5 ScissorSheep review      28 0.246  2.19  0.538
##  6 ScissorSheep stop        28 0.246  3.36  0.826
##  7 Interruptor  eggs        27 0.203  5.70  1.16
##  8 Ditobi       de          26 0.0568 3.44  0.195
##  9 Lucishungry  game        26 0.0912 0.242 0.0221
```

What words were important to (a sample of) users that had positive reviews?



Your Turn

Complete "8a-animal-crossing.Rmd"

- This time we will look at critics reviews

Lab exercise (bonus!)

Text Mining with R has an example comparing historical physics textbooks: *Discourse on Floating Bodies* by Galileo Galilei, *Treatise on Light* by Christiaan Huygens, *Experiments with Alternate Currents of High Potential and High Frequency* by Nikola Tesla, and *Relativity: The Special and General Theory* by Albert Einstein. All are available on the Gutenberg project.

Work your way through the comparison of physics books. It is section 3.4.

Thanks

- Dr. Mine Çetinkaya-Rundel
- Dr. Julia Silge: https://github.com/juliasilge/tidytext-tutorial and https://juliasilge.com/blog/animal-crossing/
- Dr. Julia Silge and Dr. David Robinson: https://www.tidytextmining.com/