THE OHIO STATE UNIVERSITY
FISHER COLLEGE OF BUSINESS

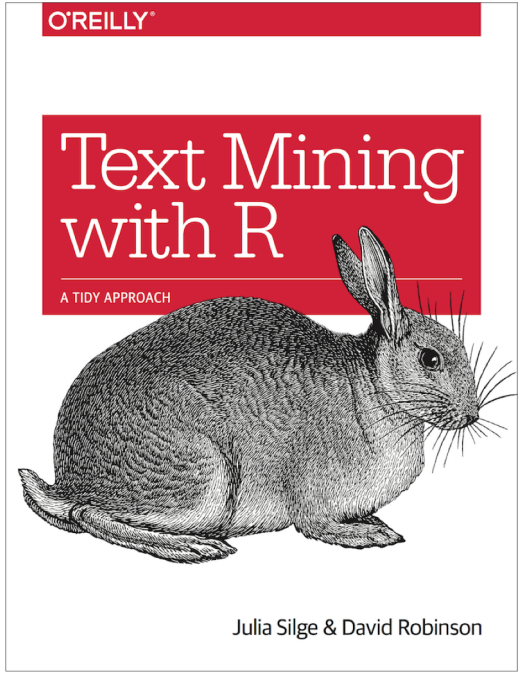
BUSMGT 7331: Descriptive Analytics and Visualization
Week 3

Describing Textual Data

Hyunwoo Park
Fisher College of Business
The Ohio State University

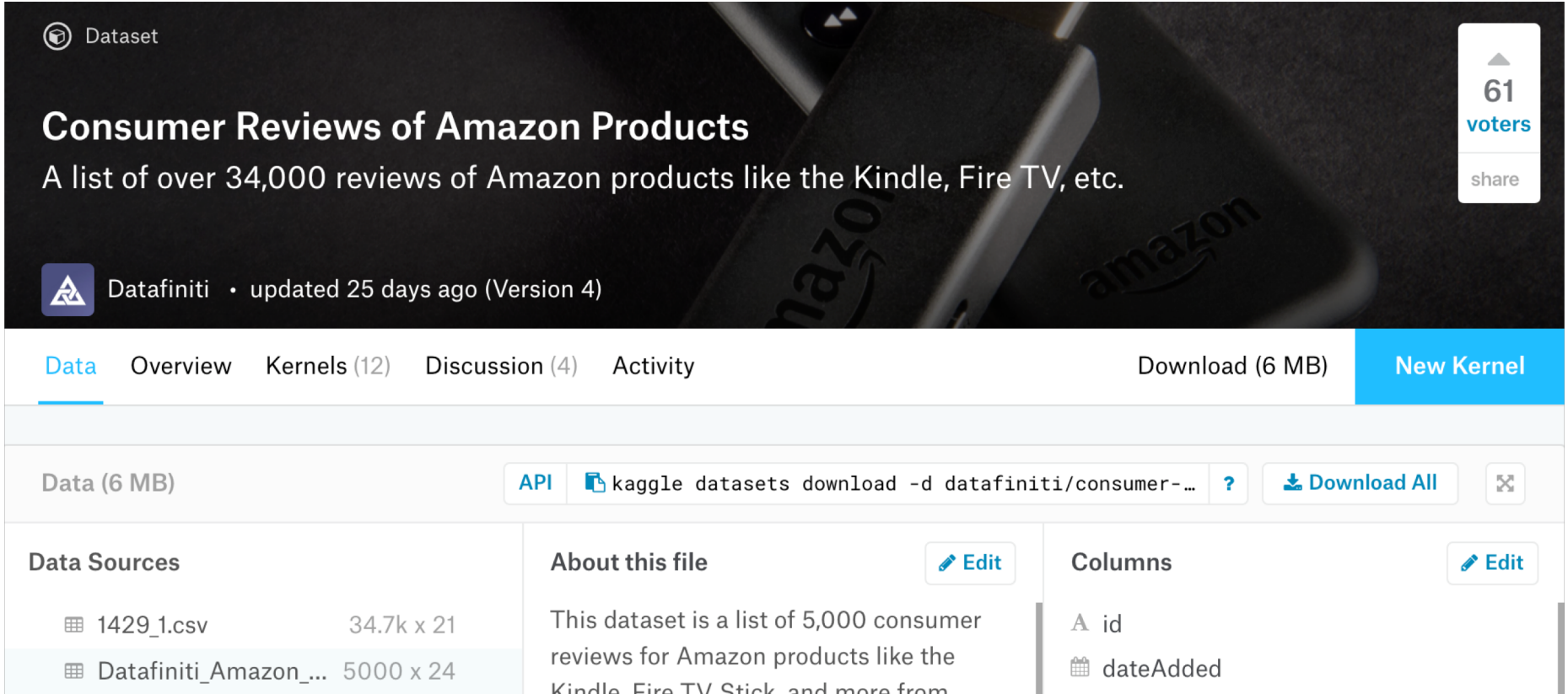
Textbook for this week

- Text Mining with R
- Available free online: https://www.tidytextmining.com/
- I will assign relevant chapters for reading.
- This book will be abbreviated as “TMwR” hereinafter.



Dataset for this week

- https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products



Packages to install

```
install.packages("tidytext")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("tm")
install.packages("quanteda")
install.packages("text2vec")
install.packages("topicmodels")
```

Tidy Text

Reading

- TMwR Chapter 1. The Tidy Text Format

Unstructured data

- In the past weeks, we learned how to describe a dataset in rectangular format.
- Not all data is stored and prepared cleanly.
- Images, videos, and texts are some of those unstructured data.
- Processing and extracting numerical features from these unstructured data are active areas of research in computer science.
- The analysis approach for textual data has been standardized to a great degree.
- Many organizations have accumulated a lot of textual data, so there's a big potential.

Tidy text data

- Tidy data
  - Each row is an observation.
  - Each column is a variable.
  - Each cell is a value.
- Tidy text format
  - Each row is a word (or a term or a token) per document.
  - Punctuation is stripped.
  - Each token is lower-cased.
  - Columns at the other levels (e.g., document) from original file are retained.
- There are other types of text formats.
  - String
  - Corpus: raw strings + metadata
  - Document-term matrix

A text analysis flowchart

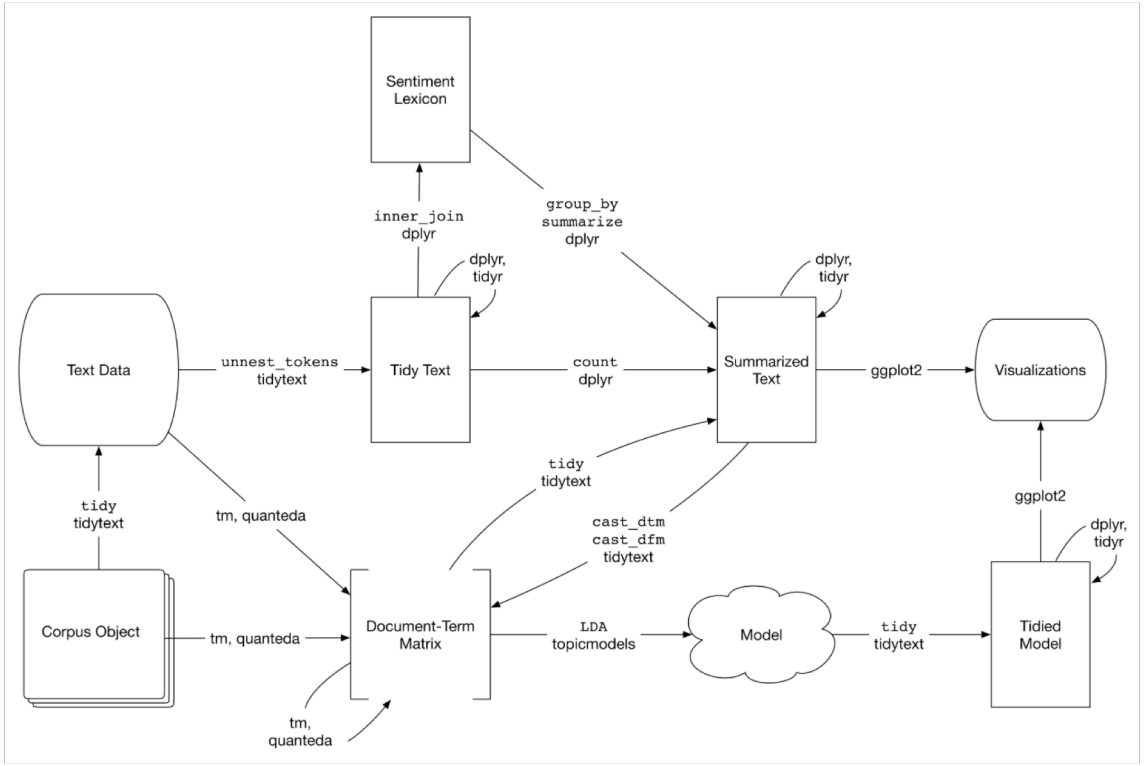


Figure from TMwR, p. 90 (https://www.tidytextmining.com/topicmodeling.html)

Data import

```
amazon <-
read_csv("data/Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv.zip"
)
amz <- amazon %>%
  select(name, reviews.rating, reviews.text) %>%
  mutate(review_num=row_number()) %>%
  filter(reviews.text!="")
amz %>%
  group_by(name) %>%
  summarize(n=n(), rating=mean(reviews.rating)) %>%
  arrange(-n, -rating)
```

```
> amz <- amazon %>%
+   select(name, reviews.rating, reviews.text) %>%
+   mutate(review_num=row_number()) %>%
+   filter(reviews.text!="")
> amz
# A tibble: 5,000 x 4
   name                                reviews.rating reviews.text                                                                  review_num
   <chr>                                        <int> <chr>                                                                          <int>
 1 "Amazon Kindle E-Reader 6\" Wifi (8…             3 I thought it would be as big as small paper but turn out to be just like my palm. I think it is too small to …          1
 2 "Amazon Kindle E-Reader 6\" Wifi (8…             5 This kindle is light and easy to use especially at the beach!!!                                                  2
 3 "Amazon Kindle E-Reader 6\" Wifi (8…             4 Didnt know how much i'd use a kindle so went for the lower end. im happy with it, even if its a little dark     3
 4 "Amazon Kindle E-Reader 6\" Wifi (8…             5 I am 100 happy with my purchase. I caught it on sale at a really good price. I am normally a real book person…  4
 5 "Amazon Kindle E-Reader 6\" Wifi (8…             5 Solid entry level Kindle. Great for kids. Gifted for a kid of my friend and they love to use it to read more …  5
 6 "Amazon Kindle E-Reader 6\" Wifi (8…             5 This make an excellent ebook reader. Don't expect much from this device except to read basic ebooks. The good…  6
 7 "Amazon Kindle E-Reader 6\" Wifi (8…             5 I ordered this for my daughter, as I have the black paperwhite, and I love it. I read quite a bit and the lar…  7
 8 "Amazon Kindle E-Reader 6\" Wifi (8…             4 I bought my Kindle about 2 months ago and the battery is already dead and will not charge                        8
 9 "Amazon Kindle E-Reader 6\" Wifi (8…             5 amazon kindle is always the best ebook, upgrade every new model                                                  9
10 "Amazon Kindle E-Reader 6\" Wifi (8…             5 It's beyond my expectation, and it can even show music score. Not fast turning though.                           10
# ... with 4,990 more rows
```

```
> amazon %>%
+   select(name, reviews.rating, reviews.text) %>%
+   filter(reviews.text!="") %>%
+   group_by(name) %>%
+   summarize(n=n(), rating=mean(reviews.rating)) %>%
+   arrange(-n, -rating)
# A tibble: 23 x 3
   name                                                                                                  n rating
   <chr>                                                                                             <int>  <dbl>
 1 "Amazon Echo Show Alexa-enabled Bluetooth Speaker with 7\" Screen"                                   845   4.66
 2 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 16 GB - Includes Special Offers, Magenta"         797   4.60
 3 Amazon - Echo Plus w/ Built-In Hub - Silver                                                         590   4.75
 4 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Blue Kid-Proof Case                               561   4.58
 5 "Brand New Amazon Kindle Fire 16gb 7\" Ips Display Tablet Wifi 16 Gb Blue"                          467   4.51
 6 Fire Tablet, 7 Display, Wi-Fi, 16 GB - Includes Special Offers, Black                               371   4.46
 7 Amazon Tap - Alexa-Enabled Portable Bluetooth Speaker                                               225   4.51
 8 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Green Kid-Proof Case                              217   4.52
 9 Kindle E-reader - White, 6 Glare-Free Touchscreen Display, Wi-Fi - Includes Special Offers          159   4.58
10 Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Silver Aluminum        106   4.67
# ... with 13 more rows
```

Convert to tidy text format

- unnest_tokens() is the key function to convert raw string to tokens.

```
1  tidy_amz <- amz %>%
2    unnest_tokens(word, reviews.text)
3
4  tidy_amz %>%
5    count(word, sort=T)
```

```
> tidy_amz
# A tibble: 155,258 x 4
   name                                                reviews.rating review_num word
   <chr>                                                        <int>      <int> <chr>
 1 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 i
 2 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 thought
 3 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 it
 4 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 would
 5 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 be
 6 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 as
 7 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 big
 8 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 as
 9 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 small
10 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"              3          1 paper
# ... with 155,248 more rows
```

```
# A tibble: 5,789 x 2
   word      n
   <chr> <int>
 1 the    6747
 2 and    5028
 3 to     5021
 4 it     4819
 5 i      4598
 6 for    3717
 7 a      3489
 8 is     3034
 9 my     2876
10 this   2643
# ... with 5,779 more rows
```

Stop words

- Some tokens are obviously and intuitively not meaningful for further analysis.
- Those tokens are: a, an, the, i, you, he, she, is, are, be, etc.
- These words are called “stop words”. Removing them is a standard step.
- In the natural language processing (NLP) literature, there is a prepared set of words to be removed from text analysis. They come with tidytext package.

```
1  data("stop_words")
2  View(stop_words)
```

```
# A tibble: 1,149 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 1,139 more rows
```

Removing numbers

- Oftentimes, some tokens are only numbers (e.g., “100”) or words containing numbers (e.g., “1st”).
- The code below extracts all tokens that contain a number.
  - http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know

```
nums <- tidy_amz %>%
  filter(str_detect(word, "^[0-9]")) %>%
  select(word) %>% unique()
```

```
> nums %>% arrange(word) %>% print(n=161)
# A tibble: 161 x 1
   word
   <chr>
 1 0
 2 1
 3 1.3
 4 1.5
 5 10
 6 10.00
 7 10.1
 8 100
 9 1000
10 1000s
11 105
12 1080
13 10th
14 10x
15 11
16 11.20
17 11yr
18 12
19 120
20 128
21 128gb
22 129.99
23 13
24 139
```

Stemming

- Another standard, but more optional, step is called stemming.
- Words are inflected depending on the context in a sentence.
- love, loves, loved all share the same root, love.
- There are several stemmers.
- Porter stemmer is one of the oldest and widely used.

Final tidy text data

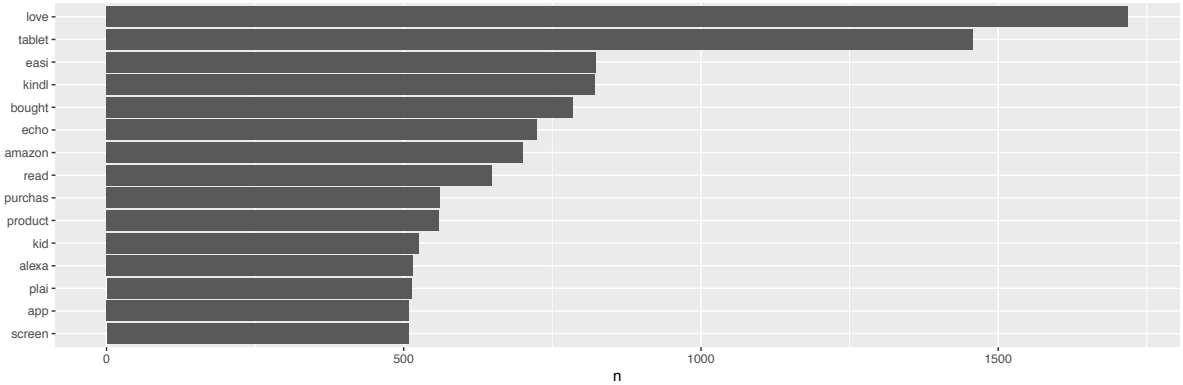
```
1  library(SnowballC)
2  tidy_amz <- tidy_amz %>%
3    anti_join(stop_words) %>%
4    anti_join(nums) %>%
5    mutate(root = wordStem(word))
6  tidy_amz %>%  count(word, sort=T)
7  tidy_amz %>%  count(root, sort=T)
```

```
> tidy_amz %>% count(word, sort=T)
# A tibble: 5,042 x 2
   word       n
   <chr>  <int>
 1 tablet  1309
 2 love    1090
 3 easy     822
 4 bought   785
 5 kindle   764
 6 amazon   694
 7 echo     693
 8 alexa    513
 9 loves    506
10 screen   500
# ... with 5,032 more rows
```

```
> tidy_amz %>% count(root, sort=T)
# A tibble: 3,645 x 2
   root       n
   <chr>  <int>
 1 love    1719
 2 tablet  1458
 3 easi     823
 4 kindl    821
 5 bought   785
 6 echo     724
 7 amazon   701
 8 read     649
 9 purchas  561
10 product  560
# ... with 3,635 more rows
```

Word count

```
1  tidy_amz %>%
2    count(root, sort=T) %>%
3    filter(n>500) %>%
4    mutate(root = reorder(root, n)) %>%
5    ggplot() +
6    geom_col(aes(root, n)) +
7    xlab(NULL) +
8    coord_flip()
```

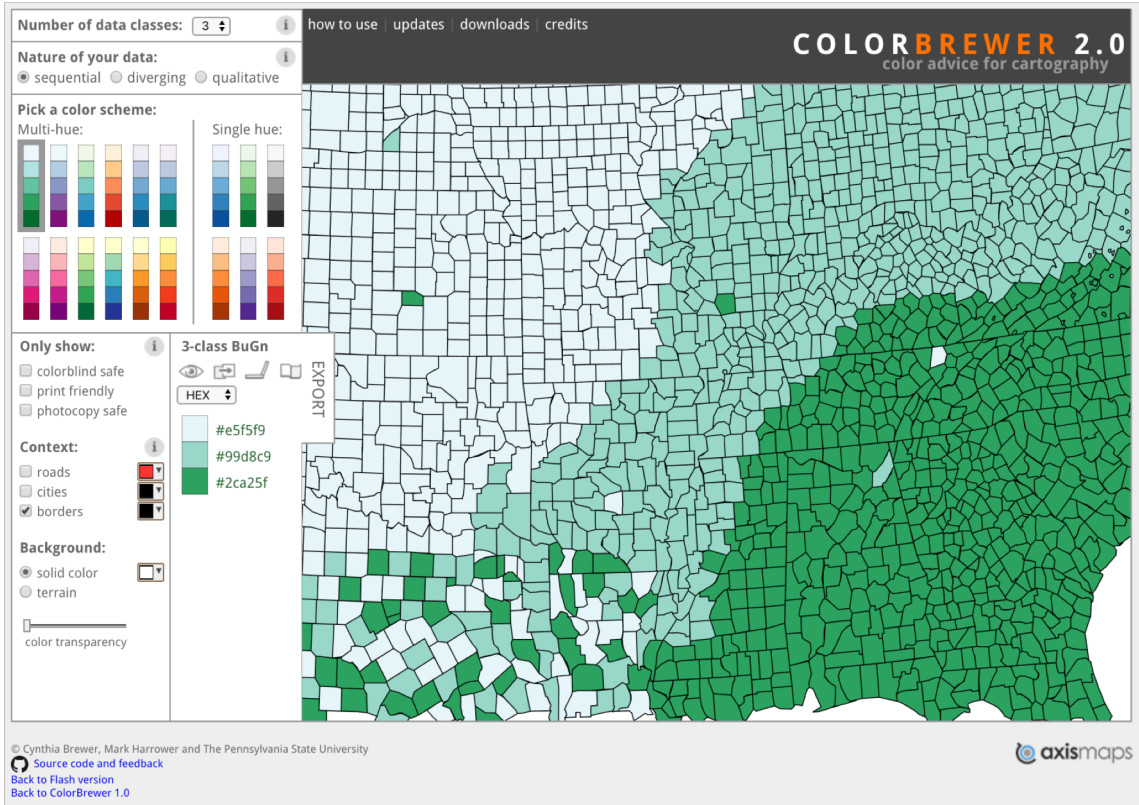


Word cloud

```
1  library(wordcloud)
2  wc <- tidy_amz %>% count(word, sort=T)
3  wordcloud(wc$word, wc$n, min.freq=50, colors=brewer.pal(5, "Dark2"))
   ?wordcloud
```

Color brewer

- Provides some sensible preset color scales
- http://colorbrewer2.org



Document-Term Matrix

Reading

- TMwR Chapter 3. Analyzing Word and Document Frequency: tf-idf
- TMwR Chapter 5. Converting to and from Nontidy Formats

Word count and term frequency (tf)

- Let's start characterizing documents as a collection of words.
- The first step would be to count occurrences of each word in a document.
- Suppose our analysis considers all reviews associated a product as a single document. (i.e., a product = a document)
- Term frequency is word count normalized by document length.

```
> words_amz
# A tibble: 13,001 x 3
   prod_id root       n
     <int> <chr>  <int>
 1       3 tablet   428
 2      10 echo     400
 3      10 love     320
 4      17 tablet   283
 5      17 love     276
 6      10 alexa    253
 7      17 kid      242
 8       6 echo     229
 9      14 tablet   227
10      20 tablet   211
# ... with 12,991 more rows
```

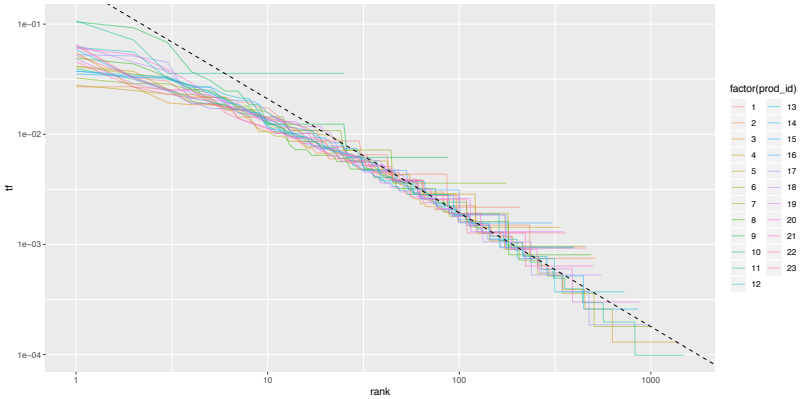
Computing tf

```
1  prods_amz <- tidy_amz %>% count(name) %>% mutate(prod_id = row_number()) %>%
   select(-n)
2  words_amz <- tidy_amz %>% left_join(prods_amz) %>% group_by(prod_id) %>%
   count(root, sort=T) %>% ungroup()
3  totals_amz <- words_amz %>% group_by(prod_id) %>% summarize(total=sum(n))
4  words_amz <- left_join(words_amz, totals_amz)
5  freq_by_rank <- words_amz %>% group_by(prod_id) %>% mutate(rank=row_number(),
   tf=n/total) %>% ungroup()
```

```
> freq_by_rank
# A tibble: 13,001 x 6
# Groups:   prod_id [23]
   prod_id root       n total  rank     tf
     <int> <chr>  <int> <int> <int>  <dbl>
 1       3 tablet   428  7718     1 0.0555
 2      10 echo     400 10131     1 0.0395
 3      10 love     320 10131     2 0.0316
 4      17 tablet   283  5370     1 0.0527
 5      17 love     276  5370     2 0.0514
 6      10 alexa    253 10131     3 0.0250
 7      17 kid      242  5370     3 0.0451
 8       6 echo     229  5559     1 0.0412
 9      14 tablet   227  3865     1 0.0587
10      20 tablet   211  3332     1 0.0633
# ... with 12,991 more rows
```

Zipf's Law

```
1  zipf <- lm(log10(tf)~log10(rank), data=freq_by_rank)
2  ggplot(freq_by_rank) +
3    geom_line(aes(rank, tf, color=factor(prod_id)), alpha=.5) +
4    geom_abline(intercept=zipf$coefficients[1], slope=zipf$coefficients[2],
   linetype=2) +
5    scale_x_log10() + scale_y_log10()
```



$$\text{tf} = \alpha \cdot (rank)^{\beta}$$
$$\log(\text{tf}) = \log(\alpha) + \beta \log(rank)$$

Inverse document frequency (idf)

- Not all terms are equally important when characterizing a document.
- Term frequency is computed for each term within each document.
- Inverse document frequency (idf) is “importance” of a term across all documents.
- Idf is computed as the ratio between # documents containing a term (n_t) and total # documents (N).

$$\mathrm{tf} = f_{t,d} / \sum_{t' \in d} f_{t',d}$$

$$\mathrm{idf} = \log(N/n_t) = -\log(n_t/N)$$

Computing idf

```
1  tidy_amz %>% left_join(prods_amz) %>%
2    group_by(root) %>%
3    summarize(ndocs=n_distinct(prod_id)) %>%
4    arrange(-ndocs) %>%
5    mutate(idf=log(nrow(prods_amz)/ndocs)) %>% print(n=20)
```

```
> tidy_amz %>% left_join(prods_amz) %>%
+   group_by(root) %>%
+   summarize(ndocs=n_distinct(prod_id)) %>%
+   arrange(-ndocs) %>%
+   mutate(idf=log(nrow(prods_amz)/ndocs)) %>% print(n=20)
Joining, by = "name"
# A tibble: 3,645 x 3
   root    ndocs   idf
   <chr>   <int>  <dbl>
 1 easi       23 0
 2 love       23 0
 3 amazon     22 0.0445
 4 book       22 0.0445
 5 bought     22 0.0445
 6 product    22 0.0445
 7 purchas    22 0.0445
 8 tablet     22 0.0445
 9 batteri    21 0.0910
10 devic      21 0.0910
11 enjoi      21 0.0910
12 fire       21 0.0910
13 gift       21 0.0910
14 hand       21 0.0910
15 kindl      21 0.0910
16 life       21 0.0910
17 nice       21 0.0910
18 perfect    21 0.0910
19 price      21 0.0910
20 qualiti    21 0.0910
# ... with 3,625 more rows
```

```
> tfidf_amz %>% filter(root=="amazon") %>% arrange(idf)
# A tibble: 22 x 7
   prod_id root       n total      tf    idf   tf_idf
     <int> <chr>  <int> <int>   <dbl>  <dbl>    <dbl>
 1      10 amazon   192 10131 0.0190  0.0445 0.000842
 2       3 amazon   130  7718 0.0168  0.0445 0.000749
 3       6 amazon    55  5559 0.00989 0.0445 0.000440
 4      17 amazon    49  5370 0.00912 0.0445 0.000406
 5      20 amazon    45  3332 0.0135  0.0445 0.000600
 6      13 amazon    43  2698 0.0159  0.0445 0.000708
 7      14 amazon    42  3865 0.0109  0.0445 0.000483
 8       2 amazon    22  1335 0.0165  0.0445 0.000733
 9       5 amazon    18  1041 0.0173  0.0445 0.000769
10      18 amazon    18  1894 0.00950 0.0445 0.000422
# ... with 12 more rows
```

Tf-idf statistic

- When you multiply term frequency and inverse document frequency, you get tf-idf statistic for each term-document pair. (i.e., tf-idf = tf x idf)
- There are several variant measures for term frequency (tf) and inverse document frequency (idf).
- Tf-idf statistic is a basic but powerful basis for quantitatively comparing two documents, which is discussed in the next section.
- Reading: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Shortcut to computing tf-idf statistic

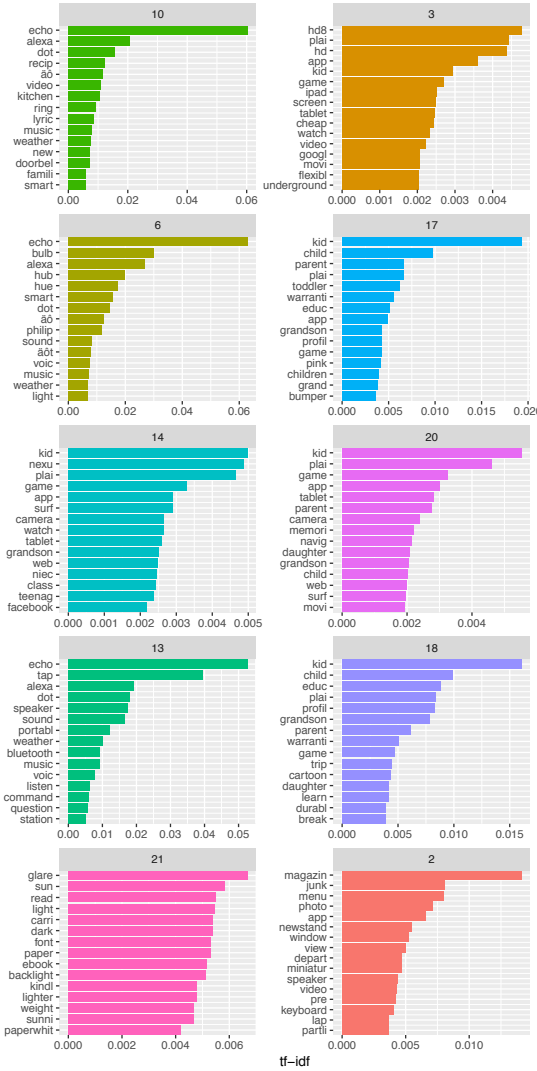
```
1  tfidf_amz <- words_amz %>% bind_tf_idf(root, prod_id, n)
2  tfidf_amz %>% filter(total>1000) %>% arrange(-tf_idf) %>% print(n=20)
3  prods_amz %>% print(n=100)
```

```
> tfidf_amz %>% filter(total>1000) %>% arrange(-tf_idf) %>% print(n=20)
# A tibble: 11,153 x 7
   prod_id root        n total      tf   idf tf_idf
     <int> <chr>   <int> <int>   <dbl> <dbl>  <dbl>
 1       6 echo      229  5559 0.0412  1.53  0.0629
 2      10 echo      400 10131 0.0395  1.53  0.0603
 3      22 oasi       29  1080 0.0269  2.04  0.0547
 4      13 echo       93  2698 0.0345  1.53  0.0526
 5      13 tap       101  2698 0.0374  1.06  0.0395
 6       8 voyag      32  1242 0.0258  1.53  0.0393
 7       6 bulb       68  5559 0.0122  2.44  0.0299
 8       6 alexa     179  5559 0.0322  0.833 0.0268
 9      10 alexa     253 10131 0.0250  0.833 0.0208
10       6 hub        63  5559 0.0113  1.75  0.0198
11       5 magazin    15  1041 0.0144  1.34  0.0194
12      17 kid       242  5370 0.0451  0.427 0.0193
13      13 alexa      62  2698 0.0230  0.833 0.0191
14      13 dot        24  2698 0.00890 2.04  0.0181
15      13 speaker    73  2698 0.0271  0.651 0.0176
16       6 hue        55  5559 0.00989 1.75  0.0173
17      13 sound      90  2698 0.0334  0.496 0.0166
18      18 kid        71  1894 0.0375  0.427 0.0160
19       6 smart      93  5559 0.0167  0.938 0.0157
20      10 dot        77 10131 0.00760 2.04  0.0155
# ... with 1.113e+04 more rows
```

```
> prods_amz %>% print(n=100)
# A tibble: 23 x 2
   name                                                                  prod_id
   <chr>                                                                   <int>
 1 All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Blue        1
 2 All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 32 GB - Includes Special Offers, Blue        2
 3 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 16 GB - Includes Special Offers, Ma…      3
 4 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 32 GB - Includes Special Offers, Bl…      4
 5 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 32 GB - Includes Special Offers, Ma…      5
 6 Amazon - Echo Plus w/ Built-In Hub - Silver                                                 6
 7 Amazon - Kindle Voyage - 4GB - Wi-Fi + 3G - Black                                           7
 8 "Amazon - Kindle Voyage - 6\" - 4GB - Black"                                                8
 9 Amazon 9W PowerFast Official OEM USB Charger and Power Adapter for Fire Tablets and K…      9
10 "Amazon Echo Show Alexa-enabled Bluetooth Speaker with 7\" Screen"                         10
11 Amazon Fire TV with 4K Ultra HD and Alexa Voice Remote (Pendant Design) | Streaming M…     11
12 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"                                   12
13 Amazon Tap - Alexa-Enabled Portable Bluetooth Speaker                                      13
14 "Brand New Amazon Kindle Fire 16gb 7\" Ips Display Tablet Wifi 16 Gb Blue"                 14
15 Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Silver Al…     15
16 "Fire HD 8 Tablet with Alexa, 8\" HD Display, 32 GB, Tangerine - with Special Offers"      16
17 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Blue Kid-Proof Case                      17
18 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Green Kid-Proof Case                     18
19 "Fire Tablet with Alexa, 7\" Display, 16 GB, Magenta - with Special Offers"                19
20 Fire Tablet, 7 Display, Wi-Fi, 16 GB - Includes Special Offers, Black                       20
21 Kindle E-reader - White, 6 Glare-Free Touchscreen Display, Wi-Fi - Includes Special O…     21
22 "Kindle Oasis E-reader with Leather Charging Cover - Black, 6\" High-Resolution Displ…     22
23 Kindle Oasis E-reader with Leather Charging Cover - Merlot, 6 High-Resolution Display…     23
```

Visualizing tf-idf per document

```
1   tfidf_vis_amz <- tfidf_amz %>% filter(total>1300) %>%
2     group_by(prod_id) %>% top_n(15) %>% ungroup() %>%
3     arrange(prod_id, -tf_idf) %>%
4     mutate(order=row_number())
5
6   facet_order <- tidy_amz %>% count(name) %>%
7     mutate(prod_id = row_number()) %>%
8     arrange(-n) %>% head(10)
9
10  tfidf_vis_amz %>%
11    ggplot(aes(order, tf_idf, fill=factor(prod_id))) +
12    geom_col(show.legend=F) +
13    scale_x_reverse(
14      breaks=tfidf_vis_amz$order,
15      labels=tfidf_vis_amz$root,
16      expand=c(0,0)) +
17    labs(x=NULL, y="tf-idf") +
18    facet_wrap(
19      ~factor(prod_id, levels=facet_order$prod_id),
20      ncol=2, scales="free") +
21    coord_flip()
```



Document-Term Matrix

- There are several text mining packages built in R.
- However, they do not directly take tidy text as input.
- The other common text data format is called Document-Term Matrix (DTM).
- Rows are documents; columns are terms; values can be raw count or tf-idf statistic.
- Document-Term Matrix object can be used directly in text mining package `tm`.
- Document-Frequency Matrix object is used in text mining package `quanteda`.

Conversion between tidy text and DTM

```
# tm package
dtm_amz <- tidy_amz %>% left_join(prods_amz) %>%
  group_by(prod_id) %>%
  count(root) %>%
  cast_dtm(term=root, document=prod_id, n)
weightTfIdf(dtm_amz)

# quanteda package
dfm_amz <- tidy_amz %>% left_join(prods_amz) %>%
  group_by(prod_id) %>%
  count(root) %>%
  cast_dfm(term=root, document=prod_id, n)
dfm_tfidf(dfm_amz)
```

```
> dtm_amz
<<DocumentTermMatrix (documents: 23, terms: 3645)>>
Non-/sparse entries: 13001/70834
Sparsity           : 84%
Maximal term length: 51
Weighting          : term frequency (tf)
> dfm_amz
Document-feature matrix of: 23 documents, 3,645 features (84.5% sparse).
```

```
> weightTfIdf(dtm_amz)
<<DocumentTermMatrix (documents: 23, terms: 3645)>>
Non-/sparse entries: 12955/70880
Sparsity           : 85%
Maximal term length: 51
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
> dfm_tfidf(dfm_amz)
Document-feature matrix of: 23 documents, 3,645 features (84.5% sparse).
```

Document Similarity

Similarities between two vectors

- DTM is basically rows of documents characterized as vectors of terms.
- In other words, we numerically represented textual data.
- One benefit of this quantification is that we can compute how similar or dissimilar two documents are.
- There are a few different ways to compute similarity between two vectors.
  - Jaccard similarity
  - Cosine similarity
  - Euclidean distance
- For illustration, suppose two documents are characterized as follows:
  - Doc1 = [0, 1]; Doc1 = “echo”
  - Doc2 = [1, 2]; Doc2 = “amazon echo echo”
- Reading: http://text2vec.org/similarity.html

Prepare toy documents

```
docs <- tribble(
  ~doc_id, ~text,
  1, "echo",
  2, "amazon echo echo"
)

dfm_docs <- docs %>%
  unnest_tokens(word, text) %>%
  group_by(doc_id) %>%
  count(word) %>%
  cast_dfm(word, doc_id, n)
```

```
> docs
# A tibble: 2 x 2
  doc_id text
   <dbl> <chr>
1      1 echo
2      2 amazon echo echo
```

```
> dfm_docs
Document-feature matrix of: 2 documents, 2 features (25.0% sparse).
2 x 2 sparse Matrix of class "dfm"
        features
docs     1 2
  echo   1 2
  amazon 0 1
```

Jaccard similarity

$$\text{Jaccard}(doc_1, doc_2) = \frac{|doc_1 \cap doc_2|}{|doc_1 \cup doc_2|}$$

$$\text{Jaccard}(doc_1, doc_2) = \frac{|\{\text{“}echo\text{”}\}|}{|\{\text{“}amazon\text{”}, \text{“}echo\text{”}\}|} = \frac{1}{2}$$

```
1  library(text2vec)
2  sim2(dfm_docs[1], dfm_docs[2], method="jaccard", norm="none")
```

```
> sim2(dfm_docs[1], dfm_docs[2], method="jaccard", norm="none")
1 x 1 sparse Matrix of class "dgCMatrix"
     amazon
echo    0.5
```

Cosine similarity & cosine distance

$$\text{CosSim}(doc_1, doc_2) = \frac{doc_1 \cdot doc_2}{|doc_1| \cdot |doc_2|}$$

$$\text{CosSim}(doc_1, doc_2) = \frac{0 \cdot 1 + 1 \cdot 2}{\sqrt{0^2 + 1^2} \cdot \sqrt{1^2 + 2^2}} = \frac{2}{\sqrt{5}}$$

$$\text{CosDist}(doc_1, doc_2) = 1 - \text{CosSim}(doc_1, doc_2)$$

```
1  sim2(dfm_docs[1], dfm_docs[2], method="cosine", norm="none")
2  sim2(dfm_docs[1], dfm_docs[2], method="cosine", norm="l2")
```

```
> sim2(dfm_docs[1], dfm_docs[2], method="cosine", norm="none")
1 x 1 sparse Matrix of class "dgCMatrix"
     amazon
echo      2
> sim2(dfm_docs[1], dfm_docs[2], method="cosine", norm="l2")
1 x 1 sparse Matrix of class "dgCMatrix"
        amazon
echo 0.8944272
```

Euclidean distance

```
1  dist2(as.matrix(dfm_docs[1]), as.matrix(dfm_docs[2]), method="euclidean",
   norm="none")
2  sqrt(1^2+1^2)
3  dist2(as.matrix(dfm_docs[1]), as.matrix(dfm_docs[2]), method="euclidean",
   norm="l2")
4  sqrt((1/sqrt(5))^2+(1-2/sqrt(5))^2)
5  dist2(as.matrix(dfm_docs[1]), as.matrix(dfm_docs[2]), method="euclidean",
   norm="l1")
6  sqrt(2*(1/3)^2)
```

Document similarities for the Amazon review corpus

```
1  dfm_tfidf_amz <- dfm_tfidf(dfm_amz)
2  cossims <- sim2(dfm_tfidf_amz, dfm_tfidf_amz, method="cosine", norm="l2")
3  td_sims <- as.tibble(tidy(cossims))
```

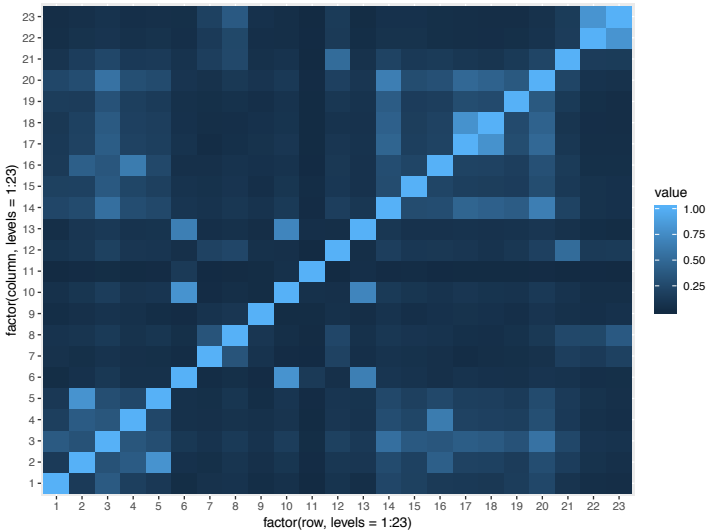
```
> cossims
23 x 23 sparse Matrix of class "dgCMatrix"
   [[ suppressing 23 column names '1', '2', '3' ... ]]

1  1.00000000 0.13056741 0.37757851 0.171593886 0.11604113 0.02464404 0.062823104 0.07397502
2  0.13056741 1.00000000 0.31878455 0.389219076 0.79619494 0.05696667 0.042448100 0.08436595
3  0.37757851 0.31878455 1.00000000 0.347448260 0.28606599 0.10986857 0.073523254 0.12651115
4  0.17159389 0.38921908 0.34744826 1.000000000 0.23657202 0.05486359 0.045053887 0.06693726
5  0.11604113 0.79619494 0.28606599 0.236572021 1.00000000 0.05043056 0.042638554 0.09297917
6  0.02464404 0.05696667 0.10986857 0.054863588 0.05043056 1.00000000 0.019294661 0.04330363
7  0.06282310 0.04244810 0.07352325 0.045053887 0.04263855 0.01929466 1.000000000 0.33527994
8  0.07397502 0.08436595 0.12651115 0.066937263 0.09297917 0.04330363 0.335279942 1.00000000
9  0.02818419 0.04601655 0.07163310 0.054703932 0.02799255 0.01369634 0.070102396 0.09738347
10 0.05437254 0.09294026 0.15426260 0.079647031 0.09063408 0.79683882 0.015123854 0.03400811
11 0.00322072 0.01554392 0.02467021 0.007722206 0.01384641 0.12838205 0.006701997 0.01231439
12 0.07697944 0.10116634 0.18259624 0.099878444 0.09103649 0.04717785 0.193313631 0.22809349
13 0.03343665 0.09445938 0.12233169 0.072596630 0.07569127 0.65495104 0.041243781 0.05170402
14 0.22839980 0.26513580 0.53998299 0.283608268 0.24336763 0.08905993 0.072101733 0.10394272
```

```
> td_sims
# A tibble: 529 x 3
   row   column  value
   <chr> <chr>   <dbl>
 1 1     1       1.000
 2 2     1       0.131
 3 3     1       0.378
 4 4     1       0.172
 5 5     1       0.116
 6 6     1       0.0246
 7 7     1       0.0628
 8 8     1       0.0740
 9 9     1       0.0282
10 10    1       0.0544
# ... with 519 more rows
```

Visualizing document similarities

```
1  td_sims %>% ggplot(
2    aes(factor(row, levels=1:23), factor(column, levels=1:23))) +
3    geom_tile(aes(fill=value))
```



Topic Modeling

Reading

- TMwR Chapter 6. Topic Modeling

Motivation for topic modeling

- Describing documents as a bag of words was a nice starting point.
- But, we often think of a document at another level of abstraction than just a collection of words.
- This is where topic modeling comes into play.
- Topic modeling aims to describe a document as a weighted set of topics and a topic as a weighted set of words.
- It's like inserting the topics layer between the documents layer and the words layer.
- There are a few widely used topic modeling techniques.
  - Latent Semantic Index (LSI) or Latent Semantic Analysis (LSA)
  - Latent Dirichlet Allocation (LDA)
  - Hierarchical Dirichlet Process (HDP)

Latent Dirichlet allocation (LDA) model

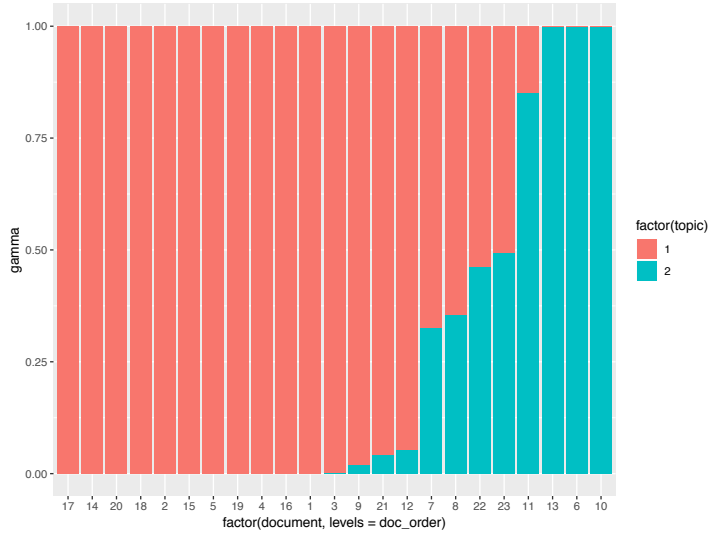
```
1  lda_amz <- LDA(dtm_amz, k=2)
2  lda_amz
3  lda_amz@call
4  lda_amz@Dim
```

```
> lda_amz <- LDA(dtm_amz, k=2)
> lda_amz
A LDA_VEM topic model with 2 topics.
> lda_amz@call
LDA(x = dtm_amz, k = 2)
> lda_amz@Dim
[1]   23 3645
```

Describing documents with topics

```
1  lda_docs <- tidy(lda_amz, matrix="gamma")
2  doc_order <- lda_docs %>% filter(topic==1) %>% arrange(-gamma) %>%
   pull(document)
3  lda_docs %>% arrange(topic, -gamma) %>%
4    ggplot(aes(x=factor(document, levels=doc_order),
5               y=gamma, fill=factor(topic))) + geom_col()
6  prods_amz %>% arrange(factor(prod_id, levels=doc_order)) %>% print(n=23)
```

```
> prods_amz %>% arrange(factor(prod_id, levels=doc_order)) %>% print(n=23)
# A tibble: 23 x 2
   name                                                                prod_id
   <chr>                                                                 <int>
 1 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Blue Kid-Proof Case    17
 2 "Brand New Amazon Kindle Fire 16gb 7\" Ips Display Tablet Wifi 16 Gb Blue"    14
 3 Fire Tablet, 7 Display, Wi-Fi, 16 GB - Includes Special Offers, Black    20
 4 Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16 GB, Green Kid-Proof Case    18
 5 All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 32 GB - Includes Special Offers, Blue     2
 6 Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Silver Alu…    15
 7 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 32 GB - Includes Special Offers, Mag…     5
 8 "Fire Tablet with Alexa, 7\" Display, 16 GB, Magenta - with Special Offers"    19
 9 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 32 GB - Includes Special Offers, Bla…     4
10 "Fire HD 8 Tablet with Alexa, 8\" HD Display, 32 GB, Tangerine - with Special Offers"    16
11 All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Blue     1
12 "All-New Fire HD 8 Tablet, 8\" HD Display, Wi-Fi, 16 GB - Includes Special Offers, Mag…     3
13 Amazon 9W PowerFast Official OEM USB Charger and Power Adapter for Fire Tablets and Ki…     9
14 Kindle E-reader - White, 6 Glare-Free Touchscreen Display, Wi-Fi - Includes Special Of…    21
15 "Amazon Kindle E-Reader 6\" Wifi (8th Generation, 2016)"    12
16 Amazon - Kindle Voyage - 4GB - Wi-Fi + 3G - Black     7
17 "Amazon - Kindle Voyage - 6\" - 4GB - Black"     8
18 "Kindle Oasis E-reader with Leather Charging Cover - Black, 6\" High-Resolution Displa…    22
19 Kindle Oasis E-reader with Leather Charging Cover - Merlot, 6 High-Resolution Display …    23
20 Amazon Fire TV with 4K Ultra HD and Alexa Voice Remote (Pendant Design) | Streaming Me…    11
21 Amazon Tap - Alexa-Enabled Portable Bluetooth Speaker    13
22 Amazon - Echo Plus w/ Built-In Hub - Silver     6
23 "Amazon Echo Show Alexa-enabled Bluetooth Speaker with 7\" Screen"    10
```

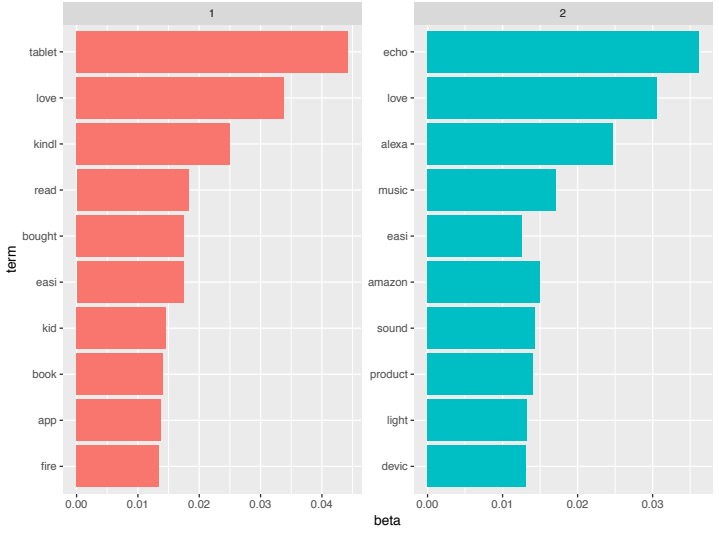


Describing topics with words

```
lda_topics <- tidy(lda_amz, matrix="beta")
top_topics <- lda_topics %>%
  group_by(topic) %>% top_n(10, beta) %>%
  ungroup() %>% arrange(topic, -beta)
top_topics %>%
  mutate(term=reorder(term, beta)) %>% ggplot() +
  geom_col(aes(term, beta, fill=factor(topic)), show.legend=F) +
  facet_wrap(~topic, scale="free") + coord_flip()
```

```
> lda_topics
# A tibble: 7,290 x 3
   topic term           beta
   <int> <chr>         <dbl>
 1     1 account   0.00159
 2     2 account   0.000703
 3     1 adequ     0.0000915
 4     2 adequ     0.0000999
 5     1 advertis  0.000477
 6     2 advertis  0.000618
 7     1 ag        0.00217
 8     2 ag        0.000200
 9     1 amaz      0.00158
10     2 amaz      0.00390
# ... with 7,280 more rows
```

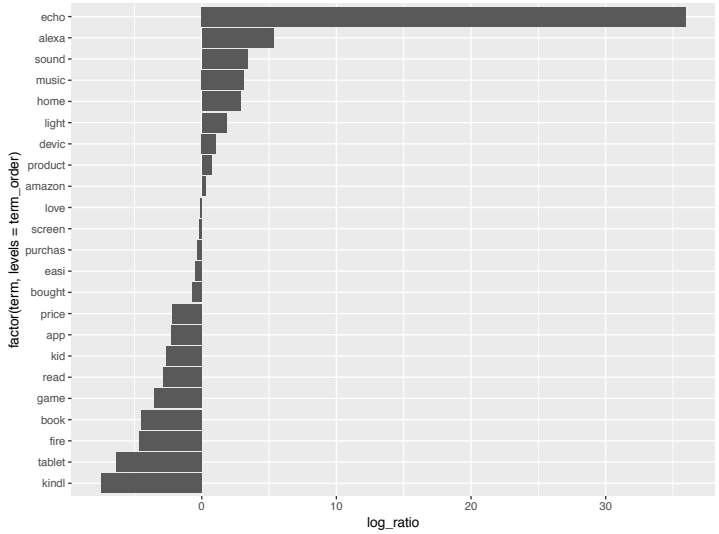
```
> top_topics
# A tibble: 20 x 3
   topic term       beta
   <int> <chr>     <dbl>
 1     1 tablet  0.0442
 2     1 love    0.0338
 3     1 kindl   0.0250
 4     1 read    0.0183
 5     1 bought  0.0175
 6     1 easi    0.0174
 7     1 kid     0.0146
 8     1 book    0.0140
 9     1 app     0.0138
10     1 fire    0.0135
11     2 echo    0.0361
12     2 love    0.0306
13     2 alexa   0.0247
14     2 music   0.0171
15     2 amazon  0.0150
16     2 sound   0.0143
17     2 product 0.0140
18     2 light   0.0133
19     2 devic   0.0131
20     2 easi    0.0126
```



Most differentiating descriptor words for topics

```
beta_spread <- lda_topics %>%
  mutate(topic = paste0("topic", topic)) %>% spread(topic, beta) %>%
  filter(topic1>.01|topic2>.01) %>%
  mutate(log_ratio=log2(topic2/topic1)) %>% arrange(log_ratio)
term_order <- beta_spread$term
beta_spread %>% ggplot() +
  geom_col(aes(factor(term, levels=term_order), log_ratio)) + coord_flip()
```

```
> beta_spread %>% print(n=100)
# A tibble: 23 x 4
   term      topic1   topic2 log_ratio
   <chr>      <dbl>    <dbl>     <dbl>
 1 kindl   2.50e- 2 0.000139     -7.49
 2 tablet  4.42e- 2 0.000554     -6.32
 3 fire    1.35e- 2 0.000540     -4.64
 4 book    1.40e- 2 0.000635     -4.46
 5 game    1.17e- 2 0.00100      -3.55
 6 read    1.83e- 2 0.00255      -2.84
 7 kid     1.46e- 2 0.00240      -2.60
 8 app     1.38e- 2 0.00291      -2.24
 9 price   1.34e- 2 0.00298      -2.17
10 bought  1.75e- 2 0.0106       -0.729
11 easi    1.74e- 2 0.0126       -0.470
12 purchas 1.15e- 2 0.00921      -0.321
13 screen  1.00e- 2 0.00900      -0.153
14 love    3.38e- 2 0.0306       -0.144
15 amazon  1.22e- 2 0.0150        0.292
16 product 8.52e- 3 0.0140        0.718
17 devic   6.34e- 3 0.0131        1.04
18 light   3.69e- 3 0.0133        1.85
19 home    1.55e- 3 0.0113        2.87
20 music   1.92e- 3 0.0171        3.15
21 sound   1.37e- 3 0.0143        3.39
22 alexa   6.19e- 4 0.0247        5.32
23 echo    5.44e-13 0.0361       36.0
```



Sentiment Analysis

Reading

- TMwR Chapter 2. Sentiment Analysis with Tidy Data

Motivation for sentiment analysis

- Text mining exercises so far relied heavily on basically count statistics such as # words, # documents, ratio between them, etc.
- When we, as humans, use natural language, we embed our emotion in our writing by using certain words.
- Of course, most words are sentiment-neutral carrying no specific emotion explicitly.
- Sentiment analysis can be particularly useful for describing, understanding, and analyzing consumer feedback or reviews.
- Then how can we quantify sentiments from natural language documents?

Dictionary-based approach

- The answer is rather simple. We use a dictionary (lexicon) of sentiments. Many people recently developed and validated several useful dictionaries.
- Note that these dictionaries are context-specific.
- The `tidytext` package comes with a set of lexicons by default.

```
1  sentiments
2  get_sentiments("afinn")
3  get_sentiments("bing")
4  get_sentiments("nrc")
```

```
> sentiments
# A tibble: 27,314 x 4
   word        sentiment lexicon score
   <chr>       <chr>     <chr>   <int>
 1 abacus      trust     nrc        NA
 2 abandon     fear      nrc        NA
 3 abandon     negative  nrc        NA
 4 abandon     sadness   nrc        NA
 5 abandoned   anger     nrc        NA
 6 abandoned   fear      nrc        NA
 7 abandoned   negative  nrc        NA
 8 abandoned   sadness   nrc        NA
 9 abandonment anger     nrc        NA
10 abandonment fear      nrc        NA
# ... with 27,304 more rows
```

```
> get_sentiments("afinn")
# A tibble: 2,476 x 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
```

```
> get_sentiments("bing")
# A tibble: 6,788 x 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
```

```
> get_sentiments("nrc")
# A tibble: 13,901 x 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
```

Merge sentiments to tidy text

```
sent_amz <- tidy_amz %>%
  left_join(prods_amz) %>% select(-name) %>%
  inner_join(get_sentiments("afinn"), by="word") %>%
  arrange(prod_id, review_num)
sent_amz <- sent_amz %>%
  group_by(prod_id, review_num) %>%
  summarize(sum_score=sum(score), mean_rating=mean(reviews.rating))
```

```
> sent_amz
# A tibble: 7,836 x 6
   reviews.rating review_num word        root     prod_id score
            <int>      <int> <chr>       <chr>      <int> <int>
 1              4       1620 love        love           1     3
 2              4       1620 gift        gift           1     2
 3              4       1620 glad        glad           1     3
 4              5       1622 pretty      pretti         1     1
 5              5       1622 solid       solid          1     2
 6              5       1622 gift        gift           1     2
 7              5       1657 love        love           1     3
 8              4       1766 opportunity opportun       1     2
 9              4       1766 fair        fair           1     2
10              4       1766 capable     capabl         1     1
# ... with 7,826 more rows
```

```
> sent_amz
# A tibble: 3,829 x 4
# Groups:   prod_id [?]
   prod_id review_num sum_score mean_rating
     <int>      <int>     <int>       <dbl>
 1       1       1620         8           4
 2       1       1622         5           5
 3       1       1657         3           5
 4       1       1766         5           4
 5       1       1823         3           5
 6       1       1850        -3           4
 7       1       1893         0           5
 8       1       1903         4           5
 9       1       1926         3           4
10       1       1934         3           4
# ... with 3,819 more rows
```

Compare sentiment with review ratings

```
1  sent_lm <- lm(mean_rating~sum_score, data=sent_amz)
2  summary(sent_lm)
3  ggplot(sent_amz, aes(sum_score, mean_rating)) +
4    geom_hex() +
5    geom_smooth() +
6    geom_abline(intercept=sent_lm$coefficients[1],
7                slope=sent_lm$coefficients[2], color="red")
```

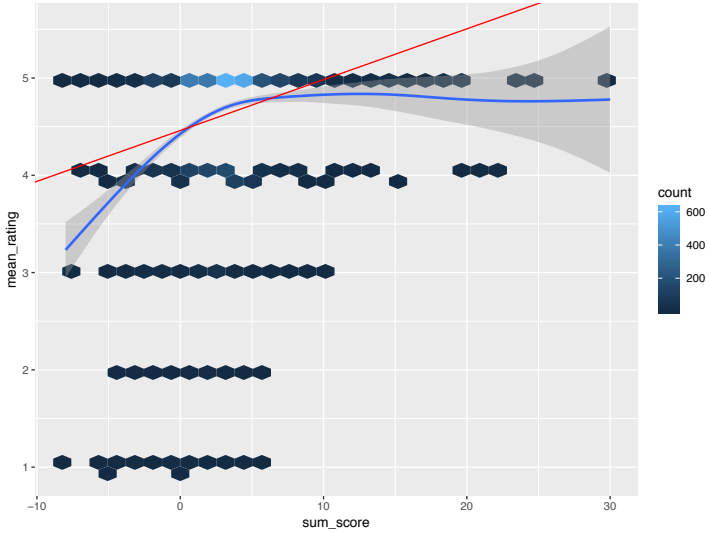
```
> summary(sent_lm)

Call:
lm(formula = mean_rating ~ sum_score, data = sent_amz)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7735 -0.4599  0.3310  0.4355  0.9581

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.459943   0.015781  282.62   <2e-16 ***
sum_score   0.052260   0.003523   14.84   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7095 on 3827 degrees of freedom
Multiple R-squared:  0.05438,   Adjusted R-squared:  0.05413
F-statistic: 220.1 on 1 and 3827 DF,  p-value: < 2.2e-16
```

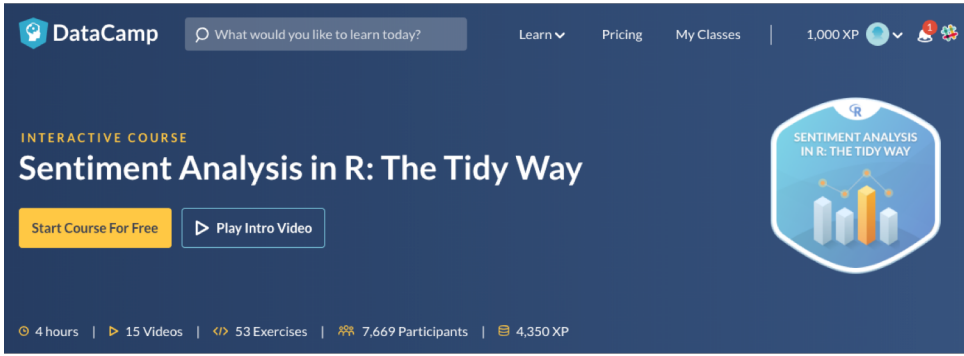


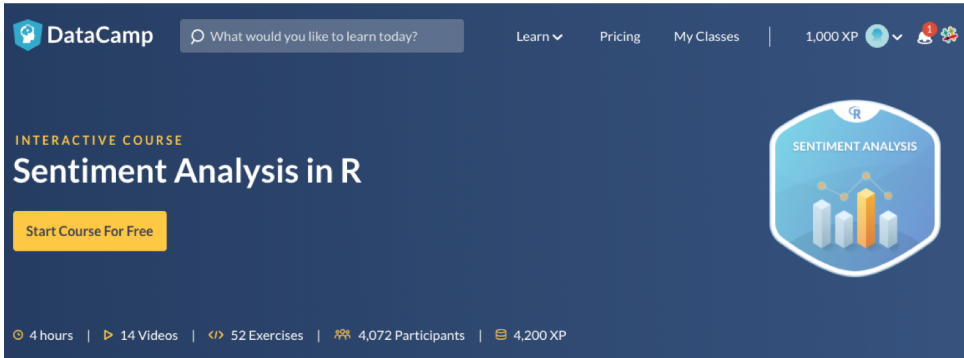
Polarity score

- The sentiment analysis we performed is at the word level, but analyzing real-world sentences involve more thinking.
- Just for instance, think about a sentence like: “I will be never happy.”
  It shouldn’t be interpreted as positive due to negation.
- There are some sentiment analysis packages dedicated to address these issues in sentence-level analysis.
- Look up `qdap` or `sentimentr` packages developed by Tyler Rinker.

Data Camp

- Complete the following two Data Camp courses on sentiment analysis.
  - [3] Sentiment Analysis in R: The Tidy Way
  - [4] Sentiment Analysis in R





Weekly Recap

Things we covered this week

- Tidy Text
  - Import to tidy text format
  - Cleaning (stop words, numbers, stemming)
  - Word count and word cloud
- Document-Term Matrix
  - Term frequency (tf)
  - Inverse document frequency (idf)
  - Tf-idf statistic
  - Conversion between tidy text and DTM
- Document Similarity
  - Jaccard similarity
  - Cosine similarity
  - Euclidean distance
- Topic Modeling
  - Latent Drichlet allocation (LDA)
  - Topic visualization
- Sentiment Analysis
  - Polarity score

Things to do this week

- 5 Quizzes (Due: Thursday, January 24, 11:59pm)
  - Week 3.1. Tidy Text
  - Week 3.2. Term Frequency
  - Week 3.3. Document Similarity
  - Week 3.4. Topic Modeling
  - Week 3.5. Sentiment Analysis
- 1 Weekly Problem (Due: Friday, January 25, 11:59pm)
- 1 Bi-weekly Assignment (Due: Saturday, January 26, 11:59pm)
- 2 DataCamp Courses (Due: Friday, February 1, 11:59pm)
  - Text Mining with R
    - [3] Sentiment Analysis in R: The Tidy Way
    - [4] Sentiment Analysis in R