# ST720 Data Science

## Converting to and from non-tidy formats

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

# Introduction

- Tidy (text) data let us use the popular suite of tidy tools such as `dplyr`, `tidyr`, and `ggplot2` to explore and visualize text data.
- However, most NLP tools in `R` are not compatible. (‘https://cran.r-project.org/web/views/NaturalLanguageProcessing.html’)

# Introduction

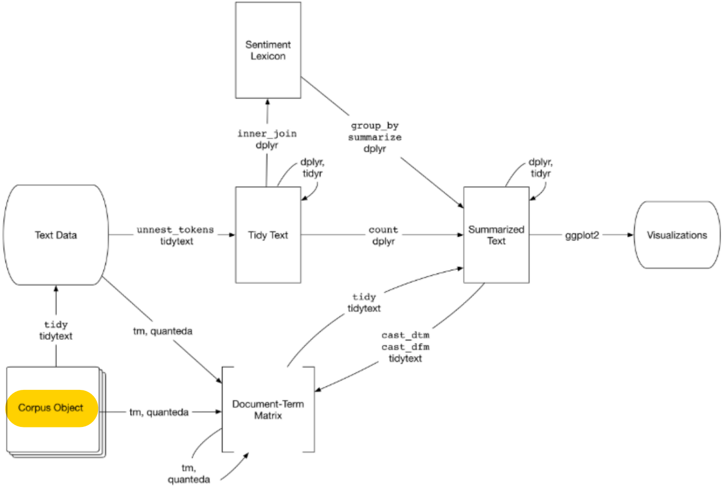



Figure 1: A flowchart of a typical text analysis

# Tidying a document-term matrix

- most common structures that text mining packages work with is the document-term matrix (or DTM).
  - each row represents one document (such as a book or article),
  - each column represents one term, and
  - each value (typically) contains the number of appearances of that term in that document.
- DTMs are usually implemented as sparse matrices, and can be stored in a more efficient format.

# Tidying a document-term matrix

- DTM objects cannot be used directly with tidy tools, and `tidytext` package provides two verbs.
  - `tidy()` turns a document-term matrix into a tidy data frame.
  - `cast()` turns a tidy one-term-per-row data frame into a matrix.
    - `cast_sparse()`: converting to a sparse matrix from `Matrix`.
    - `cast_dtm()`: converting to a DocumentTermMatrix object from `tm`
    - `cast_dfm()`: converting to a dfm object from `quanteda`.

# Tidying DocumentTermMatrix objects

- `DocumentTermMatrix` class in the `tm` package is the most popular.

```
library(tm)

data("AssociatedPress", package = "topicmodels")
AssociatedPress
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

- Dataset contains documents (each of them an AP article) and terms (distinct words).
- This DTM is 99% sparse (99% of document-word pairs are zero).

# Tidying DocumentTermMatrix objects

- Access the terms with `Terms()` function.

```
terms <- Terms(AssociatedPress)
head(terms)
```

```
## [1] "aaron"      "abandon"    "abandoned"  "abandoning" "abbo
## [6] "abboud"
```

# Tidying DocumentTermMatrix objects

- To make it tidy, we use tidy() function.

```
ap_td <- tidy(AssociatedPress)
print(ap_td, n = 5)
```

```
## # A tibble: 302,031 x 3
##   document term      count
##      <int> <chr>     <dbl>
## 1        1 adding        1
## 2        1 adult         2
## 3        1 ago           1
## 4        1 alcohol       1
## 5        1 allegedly     1
## # ... with 3.02e+05 more rows
```

- A tidy three-column tbl_df: document, term, and count. (similar to the melt() {reshape2} for non-sparse matrices.
- Only the non-zero values are included in the tidied output. No 0 for count.

# Sentiment Analysis with tidy data.

```
ap_sentiments <- ap_td %>%
  inner_join(get_sentiments("bing"), by = c(term = "word"))

ap_sentiments
```

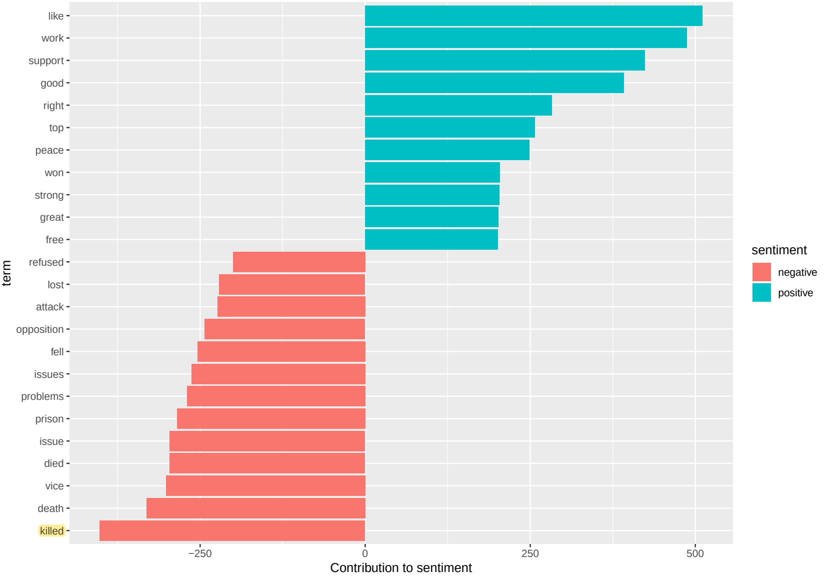
```
## # A tibble: 30,094 x 4
##    document term     count sentiment
##       <int> <chr>    <dbl> <chr>
##  1        1 assault      1 negative
##  2        1 complex      1 negative
##  3        1 death        1 negative
##  4        1 died         1 negative
##  5        1 good         2 positive
##  6        1 illness      1 negative
##  7        1 killed       2 negative
##  8        1 like         2 positive
##  9        1 liked        1 positive
## 10        1 miracle      1 positive
## # ... with 30,084 more rows
```

# Sentiment Analysis with tidy data.

- Visualize which words from the AP articles most often contributed to positive or negative sentiment.

```
ap_sentiments %>%
  count(sentiment, term, wt = count) %>%
  ungroup() %>%
  filter(n >= 200) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  ylab("Contribution to sentiment") +
  coord_flip()
```

# Sentiment Analysis with tidy data.



# Tidying `dfm` objects

- Alternative implementations of document-term matricesis `dfm` (document-feature matrix) class from the `quanteda` package.
- Example in `quanteda`: presidential inauguration speeches

```
data("data_corpus_inaugural", package = "quanteda")
inaug_dfm <- quanteda::dfm(data_corpus_inaugural,
                           verbose = FALSE)
inaug_dfm
```

```
## Document-feature matrix of: 58 documents, 9,357 features (91.
```

# Tidying dfm objects

```
inaug_td <- tidy(inaug_dfm)
inaug_td
```

```
## # A tibble: 44,709 x 3
##    document        term            count
##    <chr>           <chr>           <dbl>
##  1 1789-Washington fellow-citizens     1
##  2 1797-Adams      fellow-citizens     3
##  3 1801-Jefferson  fellow-citizens     2
##  4 1809-Madison    fellow-citizens     1
##  5 1813-Madison    fellow-citizens     1
##  6 1817-Monroe     fellow-citizens     5
##  7 1821-Monroe     fellow-citizens     1
##  8 1841-Harrison   fellow-citizens    11
##  9 1845-Polk       fellow-citizens     1
## 10 1849-Taylor     fellow-citizens     1
## # ... with 44,699 more rows
```

# Tidying dfm objects

- Find the words most specific to each of the inaugural speeches.

```
inaug_tf_idf <- inaug_td %>%
  bind_tf_idf(term, document, count) %>%
  arrange(desc(tf_idf))

print(inaug_tf_idf, n = 5)
```

```
## # A tibble: 44,709 x 6
##   document        term        count      tf   idf tf_idf
##   <chr>           <chr>       <dbl>   <dbl> <dbl>  <dbl>
## 1 1793-Washington arrive          1 0.00680  4.06 0.0276
## 2 1793-Washington upbraidings     1 0.00680  4.06 0.0276
## 3 1793-Washington violated        1 0.00680  3.37 0.0229
## 4 1793-Washington willingly       1 0.00680  3.37 0.0229
## 5 1793-Washington incurring       1 0.00680  3.37 0.0229
## # ... with 4.47e+04 more rows
```

# Tidying `dfm` objects

# Tidying dfm objects

- Pick several words and visualize how they changed in frequency over time.

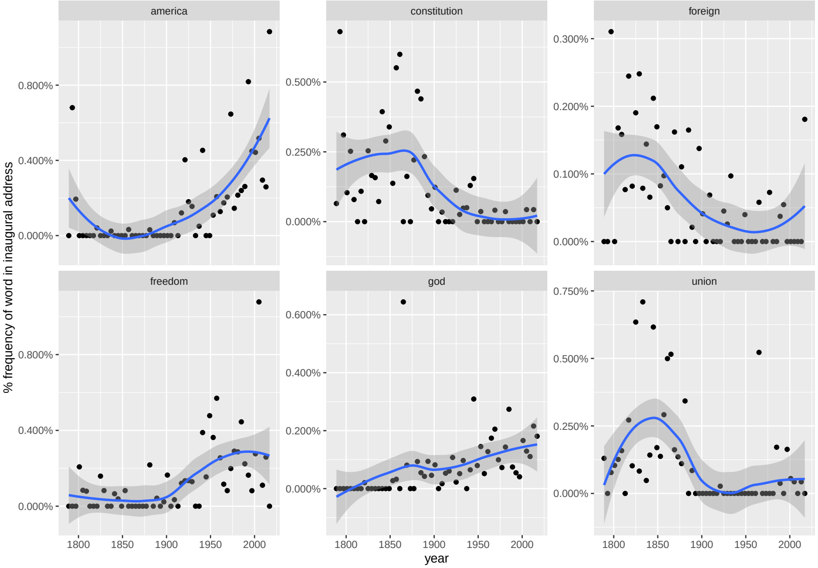
```
year_term_counts <- inaug_td %>%
  extract(document, "year", "(\\d+)", convert = TRUE) %>%
  complete(year, term, fill = list(count = 0)) %>%
  group_by(year) %>%
  mutate(year_total = sum(count))
print(year_term_counts, n = 5)
```

```
## # A tibble: 542,706 x 4
## # Groups:   year [58]
##    year term  count year_total
##   <int> <chr> <dbl>      <dbl>
## 1  1789 -         2       1538
## 2  1789 ,        70       1538
## 3  1789 ;         8       1538
## 4  1789 :         1       1538
## 5  1789 !         0       1538
## # ... with 5.427e+05 more rows
```

# Tidying dfm objects

```
year_term_counts %>%
  filter(term %in% c("god", "america", "foreign",
                     "union", "constitution", "freedom")) %>%
  ggplot(aes(year, count / year_total)) +
  geom_point() +
  geom_smooth() +
  facet_wrap(~ term, scales = "free_y") +
  scale_y_continuous(labels = scales::percent_format()) +
  ylab("% frequency of word in inaugural address")
```

# Tidying `dfm` objects



# Casting tidy text data into a DTM

- Tidied AP dataset and cast it back into a document-term matrix using the `cast_dtm()` function.

```
ap_td %>%
  cast_dtm(document, term, count)
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

# Casting tidy text data into a DFM

- Similarly, we could cast the table into a dfm object from quanteda's dfm with cast_dfm().

```r
ap_td %>%
  cast_dfm(document, term, count)
```

```
## Document-feature matrix of: 2,246 documents, 10,473 features
```

# Casting tidy text data into a matrix

- Some tools simply require a sparse matrix:

```r
library(Matrix)

m <- ap_td %>%
  cast_sparse(document, term, count)

class(m)
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

# DTM of Jane Austen's books

```r
library(janeaustenr)

austen_dtm <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word) %>%
  cast_dtm(book, word, n)

austen_dtm
```

```
## <<DocumentTermMatrix (documents: 6, terms: 14520)>>
## Non-/sparse entries: 40379/46741
## Sparsity           : 54%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

# Tidying corpus objects with metadata

- Some data structures are designed to store document collections before tokenization, often called a "corpus".
- acq corpus in tm package contains 50 articles from Reuters.

```
data("acq")
acq
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 50
```

- A corpus object is a list, with each item containing both text and metadata.

# Tidying corpus objects with metadata

- Let's tidy.

```
acq_td <- tidy(acq)
acq_td
```

```
## # A tibble: 50 x 16
##    author datetimestamp       description heading id    langu
##    <chr>  <dttm>              <chr>       <chr>   <chr> <chr>
##  1 <NA>   1987-02-27 00:18:06 ""          COMPUT~ 10    en
##  2 <NA>   1987-02-27 00:19:15 ""          OHIO M~ 12    en
##  3 <NA>   1987-02-27 00:49:56 ""          MCLEAN~ 44    en
##  4 By Ca~ 1987-02-27 00:51:17 ""          CHEMLA~ 45    en
##  5 <NA>   1987-02-27 01:08:33 ""          <COFAB~ 68    en
##  6 <NA>   1987-02-27 01:32:37 ""          INVEST~ 96    en
##  7 By Pa~ 1987-02-27 01:43:13 ""          AMERIC~ 110   en
##  8 <NA>   1987-02-27 01:59:25 ""          HONG K~ 125   en
##  9 <NA>   1987-02-27 02:01:28 ""          LIEBER~ 128   en
## 10 <NA>   1987-02-27 02:08:27 ""          GULF A~ 134   en
## # ... with 40 more rows, and 9 more variables: topics <chr>,
## #   lewissplit <chr>, cgisplit <chr>, oldid <chr>, places <na
## #   people <lgl>, orgs <lgl>, exchanges <lgl>, text <chr>
```

# Tidying corpus objects with metadata

- Then use `unnest_tokens()` to find the most common words or the ones most specific to each article.

```
acq_tokens <- acq_td %>%
  select(-places) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word")
```

# Tidying corpus objects with metadata

- Most common words

```
acq_tokens %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 1,566 x 2
##    word         n
##    <chr>    <int>
##  1 dlrs       100
##  2 pct         70
##  3 mln         65
##  4 company     63
##  5 shares      52
##  6 reuter      50
##  7 stock       46
##  8 offer       34
##  9 share       34
## 10 american    28
## # ... with 1,556 more rows
```

# Tidying corpus objects with metadata

- tf-idf

```
acq_tokens %>%
  count(id, word) %>%
  bind_tf_idf(word, id, n) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 2,853 x 6
##    id    word          n     tf   idf tf_idf
##    <chr> <chr>     <int>  <dbl> <dbl>  <dbl>
##  1 186   groupe        2 0.133   3.91  0.522
##  2 128   liebert       3 0.130   3.91  0.510
##  3 474   esselte       5 0.109   3.91  0.425
##  4 371   burdett       6 0.103   3.91  0.405
##  5 442   hazleton      4 0.103   3.91  0.401
##  6 199   circuit       5 0.102   3.91  0.399
##  7 162   suffield      2 0.1     3.91  0.391
##  8 498   west          3 0.1     3.91  0.391
##  9 441   rmj           8 0.121   3.22  0.390
## 10 467   nursery       3 0.0968  3.91  0.379
## # ... with 2,843 more rows
```

# Summary

- Text analysis requires working with a variety of tools, many of which have inputs and outputs that aren't in a tidy form.
- Should know how to convert between a tidy text data frame and sparse document-term matrices, as well as how to tidy a Corpus object containing document metadata.
- This conversion tools are an essential part of text analysis as shown in the next chapter, Tompic model.