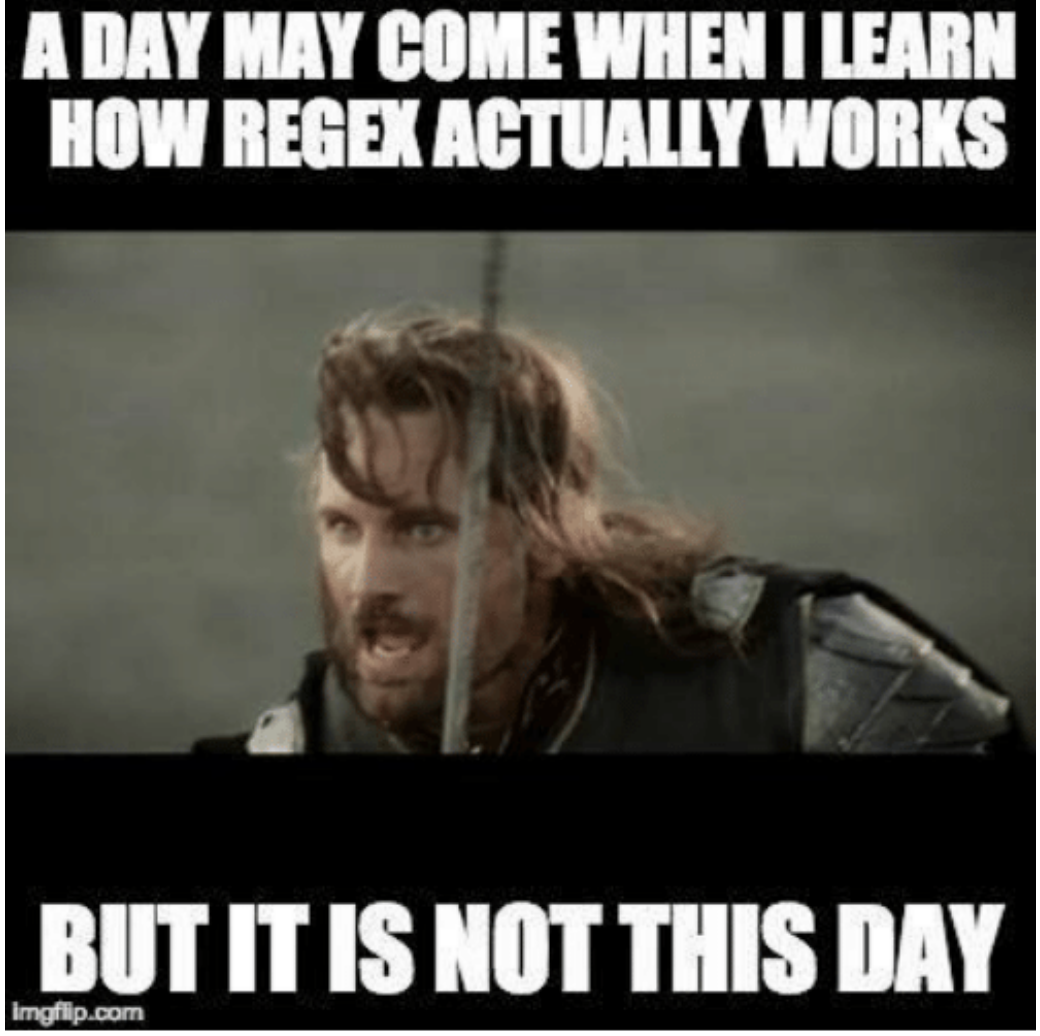
Lab Section 4: String Manipulation & Text Preprocessing in R

Napon Jatusripitak

4/25/2019

String Manipulation in R



There's a Stack Overflow post for every Regex I've ever needed

Figure 1:

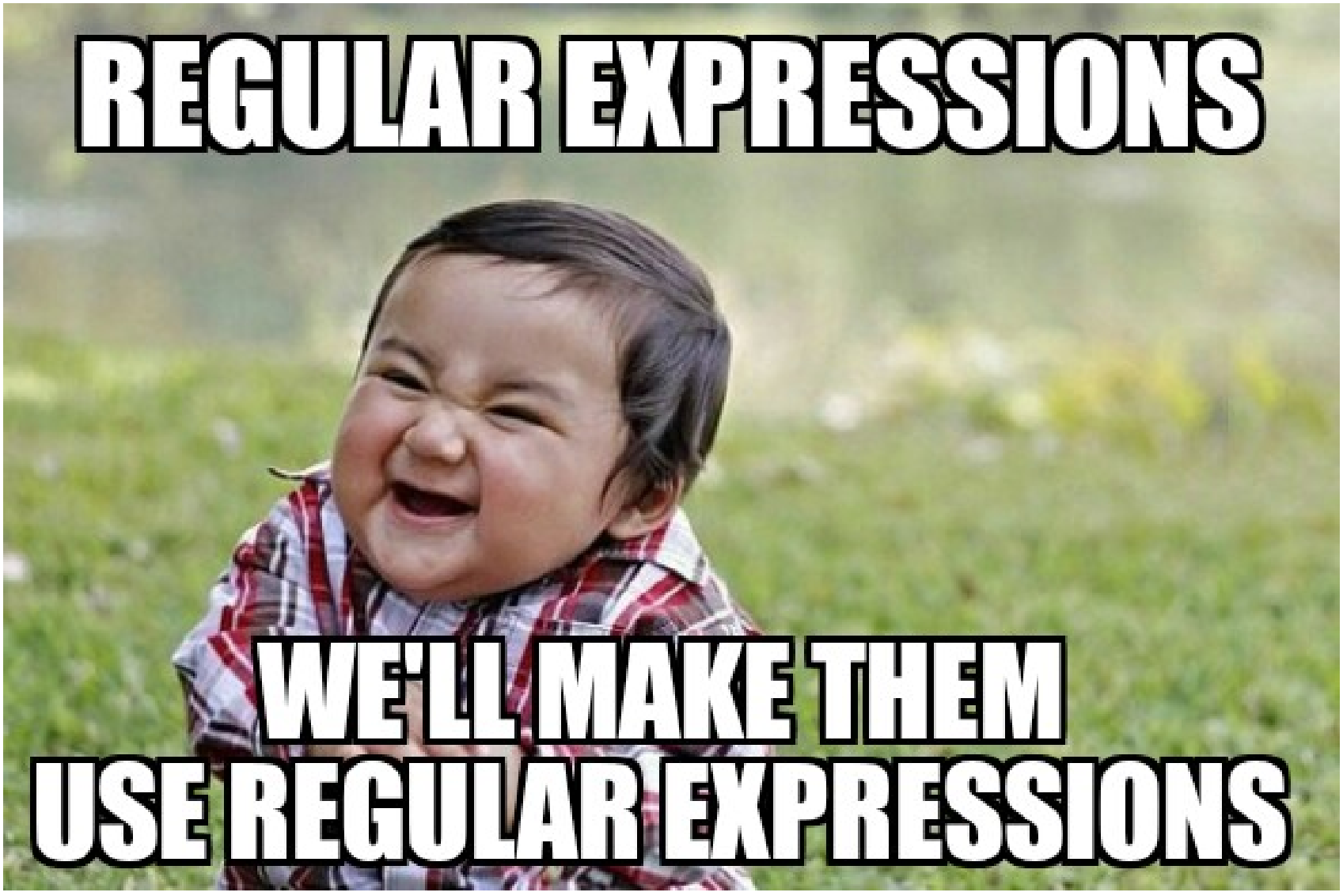

REGULAR EXPRESSIONS
WE'LL MAKE THEM
USE REGULAR EXPRESSIONS

Figure 2:

Regular Expression

Regular expressions are a set of pattern matching commands that determine how string searches are performed. They combine boolean search components as well as quantifiers.

- | indicates OR
- ( ) are used to group
- ^ looks for a starting position
- $ looks for the end of the string
- . searches for a single character
- [ ] searches for a single character in a set
- ? indicates that an element may or may not exist ($\leqslant 1$ occurrence)
- * indicates $\geqslant 0$ occurrences
- + indicates $\geqslant 1$ occurrences
- / indicates a delimiter (special character)

Examples: Working with Regex in R

In `R`, you will often use regular expressions data organization and cleaning. First, you might want to familiarize yourself with `grep`, `grepl`, `sub` and `gsub` commands.

Find matches using `grep` and `grepl`

- `grep` searches a specific pattern (case sensitive) in x where x is a character vector and returns a numeric vector that gives the position of each member of x in the pattern
- `grepl` works in a similar manner but returns a logical vector instead

```
# grep(pattern, x)
text <- c("Luke Skywalker", "Yoda", "Anakin Skywalker", "Obi-wan Kenobi")
pattern <- "Skywalker"
grep(pattern, text)
```

```
## [1] 1 3
```

```
# grepl(pattern, x)
grepl(pattern, text)
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

Find and replace first match

```
text <- "I went to the library but the library was closed."
pattern <- "library"
sub(pattern, "restaurant", text) # sub(pattern, replacement, x)
```

```
## [1] "I went to the restaurant but the library was closed."
```

Find and replace all matches

```
text <- "I went to the library but the library was closed."
pattern <- "library"
gsub(pattern, "restaurant", text) # gsub(pattern, replacement, x)
```

```
## [1] "I went to the restaurant but the restaurant was closed."
```

Now with regex operators

```
starttime <- c("1/21/2015 8:48:53", "1/23/2015 8:48:51", "1/20/2015 17:46:47")
pattern <- "^.* |:.*$"
gsub(pattern, "", starttime) # gsub(pattern, replacement, x)
```

```
## [1] "8"  "8"  "17"
```

Extracting digits from a string

```
text <- "My phone number is 987-654-3210."
pattern <- "[^0-9]" # using ^ within [] excludes whatever is inside the brackets
gsub(pattern, "", text) # gsub(pattern, replacement, x)
```

```
## [1] "9876543210"
```

Removing special characters

```
text <- "Zheng He (Chinese: ; 1371  1433 or 1435) was a Chinese mariner, explorer, diplomat, fleet admiral

pattern <- "\\(.*\\)"
gsub(pattern, "", text) # gsub(pattern, replacement, x)
```

```
## [1] "Zheng He  was a Chinese mariner, explorer, diplomat, fleet admiral, and court eunuch during China
```

Removing urls

```
text <- "Thank you @StanleyPJohnson for bringing #wildlife + #biodiversity loss into
@BBCNewsnight debate on #environment https://t.co/r0PKh7lYX1"

pattern <- "http.*"
gsub(pattern, "", text) # gsub(pattern, replacement, x)
```

```
## [1] "Thank you @StanleyPJohnson for bringing #wildlife + #biodiversity loss into \n@BBCNewsnight debate
```

I would strongly urge you to:
1. Learn different commands for string manipulation in R (base r + stringr)
2. Familiarize yourself with Regex

Additional Resources

Here are some online resources that might be helpful for you.

- Christina Maimone's GitHub Repository
- Cheat sheet
- Quick tutorial

Text Preprocessing

Preprocessing with tm

Documentation Import the data as a corpus

```
library(tm)
```

```
## Loading required package: NLP
```

```
sources <- file.path("~/Documents/GitHub/MMSS_311_2/Lab Exercises/Week 4/treaties")
names <- list.files("~/Documents/GitHub/MMSS_311_2/Lab Exercises/Week 4/treaties")

docs<-VCorpus(DirSource(sources), readerControl=list(language="fre"))
summary(docs)
```

```
##                                  Length Class             Mode
## 1998 France Model BIT edited.txt 2      PlainTextDocument list
## Argentina clean.txt              2      PlainTextDocument list
## Tajikistan clean.txt             2      PlainTextDocument list
## Zambia clean.txt                 2      PlainTextDocument list
```

Using tm_map()

```
docs<-VCorpus(DirSource(sources), readerControl=list(language="fre"))
summary(docs)
```

```
##                                  Length Class             Mode
## 1998 France Model BIT edited.txt 2      PlainTextDocument list
## Argentina clean.txt              2      PlainTextDocument list
## Tajikistan clean.txt             2      PlainTextDocument list
## Zambia clean.txt                 2      PlainTextDocument list
```

```
#writing over the corpus with a version without punctuation
docs <- tm_map(docs, removePunctuation)
#remove numbers
docs <- tm_map(docs, removeNumbers)
#make lowercase
docs <- tm_map(docs, tolower)
#removing stopwords
docs <- tm_map(docs, removeWords, stopwords("french"))
#stem the documents
docs <- tm_map(docs, stemDocument)
#remove whitespace
docs <- tm_map(docs, stripWhitespace)
#make sure it's still plain text
docs <- tm_map(docs, PlainTextDocument)
```

Creating Document-Term Matrix

```
#creating a document-term matrix, which contains the frequency of each word in the corpus
docsTDM <- DocumentTermMatrix(docs)

docsTDM.mat <- as.matrix(docsTDM)
rownames(docsTDM.mat) <- names
#print the matrix to the console
docsTDM.mat
```

```
##                                  Terms
## Docs                              autr organis aboutiss absenc
##   1998 France Model BIT edited.txt    0        0        0      0
##   Argentina clean.txt                 1        1        0      1
##   Tajikistan clean.txt                0        0        0      0
##   Zambia clean.txt                    0        0        1      1
##                                  Terms
## Docs                              accord activit affect agrment
##   1998 France Model BIT edited.txt      0        0      0        0
##   Argentina clean.txt                   1        0      1        0
##   Tajikistan clean.txt                  0        0      0        0
##   Zambia clean.txt                      2        1      2        1
##                                  Terms
## Docs                              alina amiabl applic appliqu
##   1998 France Model BIT edited.txt      0      0      0       0
##   Argentina clean.txt                   0      0      1       1
##   Tajikistan clean.txt                  0      0      0       0
##   Zambia clean.txt                      1      1      3       2
##                                  Terms
## Docs                              approb arbitrag articl autr gard
##   1998 France Model BIT edited.txt      0        0      1    0    0
```

Word Frequency

```
# word frequencies
freq <- colSums(docsTDM.mat)
ord <- order(freq)
freq[head(ord)]
```

```
##     autr organis aboutiss activit agrment   alina
##        1        1        1        1        1       1
```

```
freq[tail(ord)]
```

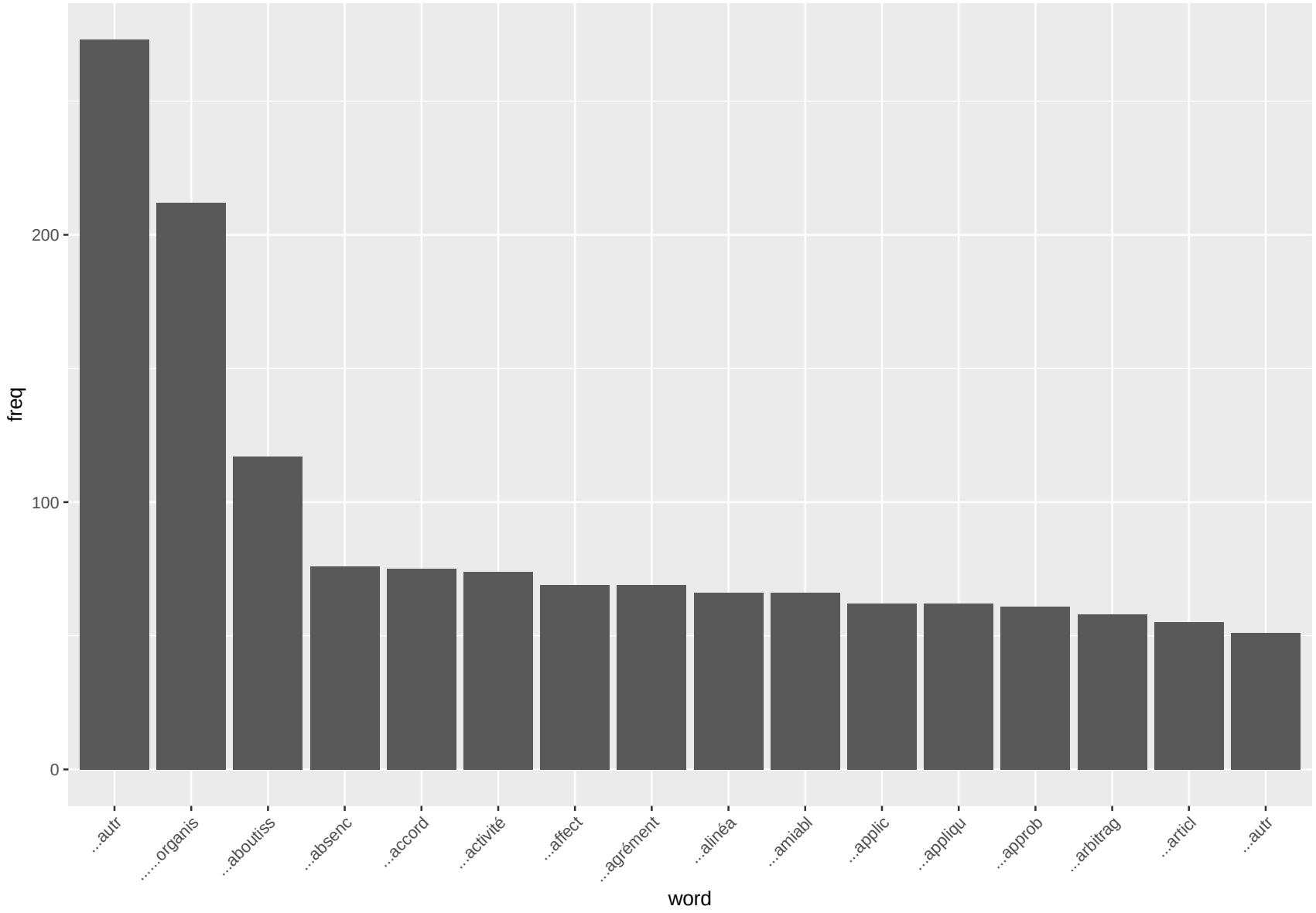
```
##        zone      accord   territoir   investiss contractant       parti
##          74          75          76         117         212         273
```

```
freq2 <- sort(colSums(docsTDM.mat), decreasing=TRUE)
head(freq2, 20)
```

```
##       parti contractant   investiss   territoir      accord        zone
##         273         212         117          76          75          74
##       droit       lautr     maritim   nationaux     socit        tout
##          69          69          66          66          62          62
##        lune      articl     prsent   diffrend        entr        autr
##          61          58          55          51          45          44
##   rpubliqu     gouvern
##          44          39
```

```
# or use a built-in function
findFreqTerms(docsTDM, 10)
```

Visualizing Word Frequency



Remove Sparse Terms

```
trms <- removeSparseTerms(docsTDM,.1)
```

Text analysis with `tidy`

Using data from faculty profiles

Text analysis with `tidy`

Tidying will arrange the text into one row per word to make analysis easier.

```
library(tidyverse)
library(tm) # for stripping whitespace
library(tidytext)

tidy_dat <- dat %>%
  mutate(text = stripWhitespace(text)) %>%
  unnest_tokens(word, text)

tidy_dat %>% select(fac_names, word) %>% .[1000:1010, ]
```

```
##       fac_names        word
## 3.28 Ana Arjona       latin
## 3.29 Ana Arjona     america
## 3.30 Ana Arjona    subfield
## 3.31 Ana Arjona specialties
## 3.32 Ana Arjona    conflict
## 3.33 Ana Arjona     studies
## 3.34 Ana Arjona         ana
## 3.35 Ana Arjona    arjonas
## 3.36 Ana Arjona    academic
## 3.37 Ana Arjona   interests
## 3.38 Ana Arjona     include
```

Describing the data

Now you can get some basic statistics:

```
tidy_dat %>%
  group_by(fac_names) %>%
  count() %>%
  arrange(-n)
```

```
## # A tibble: 45 x 2
## # Groups:   fac_names [45]
##    fac_names                  n
##    <chr>                  <int>
##  1 Zekeria Ahmed Salem      624
##  2 Wendy Pearlman           605
##  3 Jeffrey A. Winters       506
##  4 Galya Ben-Arieh          393
##  5 Elizabeth Shakman Hurd   357
##  6 Karen J. Alter           347
##  7 Jordan Gans-Morse        288
##  8 Anthony S. Chen          268
##  9 Laurel Harbridge-Yong    242
## 10 Jacqueline Stevens       227
## # ... with 35 more rows
```

Describing the data

```
tidy_dat %>%
  group_by(word) %>%
  count() %>%
  arrange(-n)
```

```
## # A tibble: 1,982 x 2
## # Groups:   word [1,982]
##    word              n
##    <chr>         <int>
##  1 and             494
##  2 the             464
##  3 of              361
##  4 in              253
##  5 political       199
##  6 politics        185
##  7 research        135
##  8 s               125
##  9 a               116
## 10 international   110
## # ... with 1,972 more rows
```

Preprocessing with `tidy`

Example: removing stopwords

```
library(gutenbergr)

hgwells <- gutenberg_download(c(35, 36, 5230, 159))

tidy_hgwells <- hgwells %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

# note that you can also use filter()
# for a smaller set of stopwords
```

Tidy Word Freq

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

```
## # A tibble: 11,769 x 2
##    word       n
##    <chr>  <int>
##  1 time     454
##  2 people   302
##  3 door     260
##  4 heard    249
##  5 black    232
##  6 stood    229
##  7 white    222
##  8 hand     218
##  9 kemp     213
## 10 eyes     210
## # ... with 11,759 more rows
```

TF-IDF

tf-idf refers to term frequency–inverse document frequency. It is a numeric measure that evaluates the importance of a given word in a given document based on how often the word appears in that document and a collection of document.

TF-IDF score for term i in document j

$$tfidf(i,j) = tf(i,j) \times idf(i)$$

$$tf(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$idf(i) = log(\frac{\text{Total documents}}{\text{docs containing term i}})$$

Tidy TF-IDF

```
book_words <- hgwells %>%
 unnest_tokens(word, text) %>%
 anti_join(stop_words) %>%
 count(gutenberg_id, word, sort = TRUE) %>%
 ungroup()

total_words <- book_words %>%
 group_by(gutenberg_id) %>%
 summarize(total = sum(n))

book_words <- left_join(book_words, total_words)

book_words
```

```
## # A tibble: 20,611 x 4
##    gutenberg_id word           n total
##           <int> <chr>      <int> <int>
##  1         5230 kemp         213 17592
##  2           35 time         200 11111
##  3         5230 invisible    180 17592
##  4          159 montgomery   179 15448
##  5         5230 door         169 17592
##  6           36 martians     163 22583
##  7           36 people       159 22583
##  8         5230 hall         149 17592
##  9          159 moreau       124 15448
## 10           36 black        122 22583
## # ... with 20,601 more rows
```

Tidy TF-IDF

```
book_words <- book_words %>%
  bind_tf_idf(word, gutenberg_id, n)
book_words
```

```
## # A tibble: 20,611 x 7
##    gutenberg_id word           n total      tf   idf  tf_idf
##           <int> <chr>      <int> <int>   <dbl> <dbl>   <dbl>
##  1         5230 kemp         213 17592 0.0121  1.39  0.0168
##  2           35 time         200 11111 0.0180  0     0
##  3         5230 invisible    180 17592 0.0102  0     0
##  4          159 montgomery   179 15448 0.0116  1.39  0.0161
##  5         5230 door         169 17592 0.00961 0     0
##  6           36 martians     163 22583 0.00722 1.39  0.0100
##  7           36 people       159 22583 0.00704 0     0
##  8         5230 hall         149 17592 0.00847 0.288 0.00244
##  9          159 moreau       124 15448 0.00803 1.39  0.0111
## 10           36 black        122 22583 0.00540 0     0
## # ... with 20,601 more rows
```

A Caveat

Note that until now, we've been using unigrams as our unit of analysis. You can also change the *ngram*.

```
bigram.docs <- all.docs %>%
unnest_tokens(ngram, text, token = "ngrams", n = 2)
```