

Text Mining with R

A Tidy Approach

Julia Silge and David Robinson

2021-12-26

Tartalomjegyzék

Welcome to Text Mining with R	5
Preface	7
Outline	7
Topics this book does not cover	8
About this book	9
Using code examples	9
1. The tidy text format	11
1.1. Contrasting tidy text with other data structures	12
1.2. The <code>unnest_tokens</code> function	12
1.3. Tidying the works of Jane Austen	14
1.4. The <code>gutenbergr</code> package	17
1.5. Word frequencies	18
1.6. Summary	19
2. Thidy text format	21
3. Thidy text format	23
4. Thidy text format	25
5. Thijhhdy text formatnnn	27
6. Thijaahdy text formatnnnsss	29
7. Thijaashdy text formatnnnsss	31
8. Thijaashdsdfsy text formatnnnsss	33
9. Thijaashdsdfsy text formatnnnsss	35
10.Thijaashdsdfsys text formatnnnsss	37
11.Thijaashdswdfsys text formatnnnsss	39

Welcome to Text Mining with R

This is the website¹ for *Text Mining with R*! Visit the GitHub repository for this site², find the book at O'Reilly³, or buy it on Amazon⁴.

This work by Julia Silge⁵ and David Robinson⁶ is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States License.

¹ <http://tidytextmining.com/>

² <https://github.com/dgrtwo/tidy-text-mining>

³ http://www.jdoqocy.com/click-4428796-11290546?url=http%3A%2F%2Fshop.oreilly.com%2Fproduct%2F0636920067153.do%3Fcmp%3Daf-strata-books-video-product_cj_0636920067153_%25zp&cjsku=0636920067153

⁴ <http://amzn.to/2tZkmxG>

⁵ <http://juliasilge.com/>

⁶ <http://varianceexplained.org/>

Preface

If you work in analytics or data science, like we do, you are familiar with the fact that data is being generated all the time at ever faster rates. (You may even be a little weary of people pontificating about this fact.) Analysts are often trained to handle tabular or rectangular data that is mostly numeric, but much of the data proliferating today is unstructured and text-heavy. Many of us who work in analytical fields are not trained in even simple interpretation of natural language.

We developed the `tidytext`⁷ (?) R package because we were familiar with many methods for data wrangling and visualization, but couldn't easily apply these same methods to text. We found that using tidy data principles can make many text mining tasks easier, more effective, and consistent with tools already in wide use. Treating text as data frames of individual words allows us to manipulate, summarize, and visualize the characteristics of text easily and integrate natural language processing into effective workflows we were already using.

This book serves as an introduction of text mining using the `tidytext` package and other tidy tools in R. The functions provided by the `tidytext` package are relatively simple; what is important are the possible applications. Thus, this book provides compelling examples of real text mining problems.

Outline

We start by introducing the tidy text format, and some of the ways `dplyr`, `tidyr`, and `tidytext` allow informative analyses of this structure.

- **Chapter 1** outlines the tidy text format and the `unnest_tokens()` function. It also introduces the `gutenbergr` and `janeaustenr` packages, which provide useful literary text datasets that we'll use throughout this book.
- **Chapter ??** shows how to perform sentiment analysis on a tidy text dataset, using the `sentiments` dataset from `tidytext` and `inner_join()` from `dplyr`.

⁷ <https://github.com/juliasilge/tidytext>

- **Chapter ??** describes the tf-idf statistic (term frequency times inverse document frequency), a quantity used for identifying terms that are especially important to a particular document.
- **Chapter ??** introduces n-grams and how to analyze word networks in text using the `widyr` and `ggraph` packages.

Text won't be tidy at all stages of an analysis, and it is important to be able to convert back and forth between tidy and non-tidy formats.

- **Chapter ??** introduces methods for tidying document-term matrices and corpus objects from the `tm` and `quanteda` packages, as well as for casting tidy text datasets into those formats.
- **Chapter ??** explores the concept of topic modeling, and uses the `tidy()` method to interpret and visualize the output of the `topicmodels` package.

We conclude with several case studies that bring together multiple tidy text mining approaches we've learned.

- **Chapter ??** demonstrates an application of a tidy text analysis by analyzing the authors' own Twitter archives. How do Dave's and Julia's tweeting habits compare?
- **Chapter ??** explores metadata from over 32,000 NASA datasets (available in JSON) by looking at how keywords from the datasets are connected to title and description fields.
- **Chapter ??** analyzes a dataset of Usenet messages from a diverse set of newsgroups (focused on topics like politics, hockey, technology, atheism, and more) to understand patterns across the groups.

Topics this book does not cover

This book serves as an introduction to the tidy text mining framework along with a collection of examples, but it is far from a complete exploration of natural language processing. The CRAN Task View on Natural Language Processing⁸ provides details on other ways to use R for computational linguistics. There are several areas that you may want to explore in more detail according to your needs.

- **Clustering, classification, and prediction:** Machine learning on text is a vast topic that could easily fill its own volume. We introduce one method of unsupervised clustering (topic modeling) in Chapter ?? but many more machine learning algorithms can be used in dealing with text.
- **Word embedding:** One popular modern approach for text analysis is to map words to vector representations, which can then be used to examine linguistic relationships between words and to classify text. Such representations of words are not tidy in the sense that we consider here, but have found powerful applications in machine learning algorithms.

⁸ <https://cran.r-project.org/web/views/NaturalLanguageProcessing.html>

- **More complex tokenization:** The tidytext package trusts the tokenizers package (?) to perform tokenization, which itself wraps a variety of tokenizers with a consistent interface, but many others exist for specific applications.
- **Languages other than English:** Some of our users have had success applying tidytext to their text mining needs for languages other than English, but we don’t cover any such examples in this book.

About this book

This book is focused on practical software examples and data explorations. There are few equations, but a great deal of code. We especially focus on generating real insights from the literature, news, and social media that we analyze.

We don’t assume any previous knowledge of text mining. Professional linguists and text analysts will likely find our examples elementary, though we are confident they can build on the framework for their own analyses.

We do assume that the reader is at least slightly familiar with dplyr, ggplot2, and the `%>%` „pipe” operator in R, and is interested in applying these tools to text data. For users who don’t have this background, we recommend books such as R for Data Science⁹. We believe that with a basic background and interest in tidy data, even a user early in their R career can understand and apply our examples.

Using code examples

This book was written in RStudio¹⁰ using bookdown¹¹. The website¹² is hosted via Netlify¹³, and automatically built after every push by GitHub Actions¹⁴. While we show the code behind the vast majority of the analyses, in the interest of space we sometimes choose not to show the code generating a particular visualization if we’ve already provided the code for several similar graphs. We trust the reader can learn from and build on our examples, and the code used to generate the book can be found in our public GitHub repository¹⁵. We generated all plots in this book using ggplot2¹⁶ and its light theme (`theme_light()`).

This version of the book was built with R version 4.1.2 (2021-11-01) and the following packages:

⁹ <http://r4ds.had.co.nz/>

¹⁰ <http://www.rstudio.com/ide/>

¹¹ <http://bookdown.org/>

¹² <https://www.tidytextmining.com/>

¹³ <http://netlify.com/>

¹⁴ <https://help.github.com/actions>

¹⁵ <https://github.com/dgrtwo/tidy-text-mining>

¹⁶ <https://ggplot2.tidyverse.org/>

package	version	source
bookdown	0.24	CRAN (R 4.1.1)
dplyr	1.0.7	CRAN (R 4.1.1)
forcats	0.5.1	CRAN (R 4.1.0)
ggforce	0.3.3	CRAN (R 4.1.0)
ggplot2	3.3.5	CRAN (R 4.1.1)
ggraph	2.0.5	CRAN (R 4.1.0)
gutenbergr	0.2.1	CRAN (R 4.1.0)
igraph	1.2.9	CRAN (R 4.1.2)
janeaustenr	0.1.5	CRAN (R 4.1.0)
jsonlite	1.7.2	CRAN (R 4.1.0)
lubridate	1.8.0	CRAN (R 4.1.2)
mallet	1.0	CRAN (R 4.1.0)
Matrix	1.3-4	CRAN (R 4.1.2)
quantada	3.2.0	CRAN (R 4.1.2)
readr	2.1.1	CRAN (R 4.1.2)
reshape2	1.4.4	CRAN (R 4.1.0)
sessioninfo	1.2.1	CRAN (R 4.1.2)
stringr	1.4.0	CRAN (R 4.1.0)
styler	1.6.2	CRAN (R 4.1.1)
textdata	0.4.1	CRAN (R 4.1.0)
tidyverse	1.1.4	CRAN (R 4.1.1)
tm	0.3.2	CRAN (R 4.1.2)
topicmodels	0.2-12	CRAN (R 4.1.0)
widyr	NA	NA
wordcloud	NA	NA
XML	3.99-0.7	CRAN (R 4.1.1)

1. fejezet

The tidy text format



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

We thus define the tidy text format as being **a table with one-token-per-row**. A token is a meaningful unit of text, such as a word, that we are interested in using for analysis, and tokenization is the process of splitting text into tokens. This one-token-per-row structure is in contrast to the ways text is often stored in current analyses, perhaps as strings or in a document-term matrix. For tidy text mining, the **token** that is stored in each row is most often a single word, but can also be an n-gram, sentence, or paragraph. In the tidytext package, we provide functionality to tokenize by commonly used units of text like these and convert to a one-term-per-row format.

Tidy data sets allow manipulation with a standard set of „tidy” tools, including popular packages such as dplyr (?), tidyr (?), ggplot2 (?), and broom (?). By keeping the input and output in tidy tables, users can transition fluidly between

these packages. We've found these tidy tools extend naturally to many text analyses and explorations.

At the same time, the tidytext package doesn't expect a user to keep text data in a tidy form at all times during an analysis. The package includes functions to `tidy()` objects (see the broom package [Robinson et al cited above]) from popular text mining R packages such as `tm` (?) and `quanteda` (?). This allows, for example, a workflow where importing, filtering, and processing is done using `dplyr` and other tidy tools, after which the data is converted into a document-term matrix for machine learning applications. The models can then be re-converted into a tidy form for interpretation and visualization with `ggplot2`.

1.1. Contrasting tidy text with other data structures

As we stated above, we define the tidy text format as being a table with **one-token-per-row**. Structuring text data in this way means that it conforms to tidy data principles and can be manipulated with a set of consistent tools. This is worth contrasting with the ways text is often stored in text mining approaches.

- **String**: Text can, of course, be stored as strings, i.e., character vectors, within R, and often text data is first read into memory in this form.
- **Corpus**: These types of objects typically contain raw strings annotated with additional metadata and details.
- **Document-term matrix**: This is a sparse matrix describing a collection (i.e., a corpus) of documents with one row for each document and one column for each term. The value in the matrix is typically word count or tf-idf (see Chapter ??).

Let's hold off on exploring corpus and document-term matrix objects until Chapter ??, and get down to the basics of converting text to a tidy format.

1.2. The `unnest_tokens` function

Emily Dickinson wrote some lovely text in her time.

```
text <- c("Because I could not stop for Death -",
       "He kindly stopped for me -",
       "The Carriage held but just Ourselves -",
       "and Immortality")

text
#> [1] "Because I could not stop for Death -"
#> [2] "He kindly stopped for me -"
#> [3] "The Carriage held but just Ourselves -"
#> [4] "and Immortality"
```

This is a typical character vector that we might want to analyze. In order to turn it into a tidy text dataset, we first need to put it into a data frame.

```
library(dplyr)
text_df <- tibble(line = 1:4, text = text)

text_df
#> # A tibble: 4 x 2
#>   line    text
#>   <int> <chr>
#> 1     1 Because I could not stop for Death -
#> 2     2 He kindly stopped for me -
#> 3     3 The Carriage held but just Ourselves -
#> 4     4 and Immortality
```

What does it mean that this data frame has printed out as a „tibble”? A tibble is a modern class of data frame within R, available in the dplyr and tibble packages, that has a convenient print method, will not convert strings to factors, and does not use row names. Tibbles are great for use with tidy tools.

Notice that this data frame containing text isn't yet compatible with tidy text analysis, though. We can't filter out words or count which occur most frequently, since each row is made up of multiple combined words. We need to convert this so that it has **one-token-per-document-per-row**.



A token is a meaningful unit of text, most often a word, that we are interested in using for further analysis, and tokenization is the process of splitting text into tokens.

In this first example, we only have one document (the poem), but we will explore examples with multiple documents soon.

Within our tidy text framework, we need to both break the text into individual tokens (a process called *tokenization*) and transform it to a tidy data structure. To do this, we use tidytext's `unnest_tokens()` function.

```
library(tidytext)

text_df %>%
  unnest_tokens(word, text)
#> # A tibble: 20 x 2
#>   line    word
#>   <int> <chr>
#> 1     1 because
#> 2     1 i
#> 3     1 could
#> 4     1 not
```

```
#> 5    1 stop
#> 6    1 for
#> 7    1 death
#> 8    2 he
#> 9    2 kindly
#> 10   2 stopped
#> # ... with 10 more rows
```

The two basic arguments to `unnest_tokens` used here are column names. First we have the output column name that will be created as the text is unnested into it (`word`, in this case), and then the input column that the text comes from (`text`, in this case). Remember that `text_df` above has a column called `text` that contains the data of interest.

After using `unnest_tokens`, we've split each row so that there is one token (`word`) in each row of the new data frame; the default tokenization in `unnest_tokens()` is for single words, as shown here. Also notice:

- Other columns, such as the line number each word came from, are retained.
- Punctuation has been stripped.
- By default, `unnest_tokens()` converts the tokens to lowercase, which makes them easier to compare or combine with other datasets. (Use the `to_lower = FALSE` argument to turn off this behavior).

Having the text data in this format lets us manipulate, process, and visualize the text using the standard set of tidy tools, namely `dplyr`, `tidyr`, and `ggplot2`, as shown in Figure ??.

1.3. Tidying the works of Jane Austen

Let's use the text of Jane Austen's 6 completed, published novels from the `janeaustenr`¹ package (?), and transform them into a tidy format. The `janeaustenr` package provides these texts in a one-row-per-line format, where a line in this context is analogous to a literal printed line in a physical book. Let's start with that, and also use `mutate()` to annotate a `linenumber` quantity to keep track of lines in the original format and a `chapter` (using a regex) to find where all the chapters are.

```
library(janeaustenr)
library(dplyr)
library(stringr)

original_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
```

¹ <https://cran.r-project.org/package=janeaustenr>

```

chapter = cumsum(str_detect(text,
                           regex("^chapter [\\\\divxlc]"),
                           ignore_case = TRUE))) %>%
ungroup()

original_books
#> # A tibble: 73,422 x 4
#>   text                  book      linenumber chapter
#>   <chr>                <fct>     <int>     <int>
#> 1 "SENSE AND SENSIBILITY" Sense & Sensibility     1       0
#> 2 ""                     Sense & Sensibility     2       0
#> 3 "by Jane Austen"      Sense & Sensibility     3       0
#> 4 ""                     Sense & Sensibility     4       0
#> 5 "(1811)"              Sense & Sensibility     5       0
#> 6 ""                     Sense & Sensibility     6       0
#> 7 ""                     Sense & Sensibility     7       0
#> 8 ""                     Sense & Sensibility     8       0
#> 9 ""                     Sense & Sensibility     9       0
#> 10 "CHAPTER 1"          Sense & Sensibility    10      1
#> # ... with 73,412 more rows

```

To work with this as a tidy dataset, we need to restructure it in the **one-token-per-row** format, which as we saw earlier is done with the `unnest_tokens()` function.

```

library(tidytext)
tidy_books <- original_books %>%
  unnest_tokens(word, text)

tidy_books
#> # A tibble: 725,055 x 4
#>   book      linenumber chapter word
#>   <fct>     <int>     <int> <chr>
#> 1 Sense & Sensibility     1       0 sense
#> 2 Sense & Sensibility     1       0 and
#> 3 Sense & Sensibility     1       0 sensibility
#> 4 Sense & Sensibility     3       0 by
#> 5 Sense & Sensibility     3       0 jane
#> 6 Sense & Sensibility     3       0 austen
#> 7 Sense & Sensibility     5       0 1811
#> 8 Sense & Sensibility    10      1 chapter
#> 9 Sense & Sensibility    10      1 1
#> 10 Sense & Sensibility   13      1 the
#> # ... with 725,045 more rows

```

This function uses the `tokenizers`² package to separate each line of text in the original data frame into tokens. The default tokenizing is for words, but other options include characters, n-grams, sentences, lines, paragraphs, or separation around a regex pattern.

Now that the data is in one-word-per-row format, we can manipulate it with tidy tools like `dplyr`. Often in text analysis, we will want to remove stop words; stop words are words that are not useful for an analysis, typically extremely common words such as „the”, „of”, „to”, and so forth in English. We can remove stop words (kept in the `tidytext` dataset `stop_words`) with an `anti_join()`.

```
data(stop_words)

tidy_books <- tidy_books %>%
  anti_join(stop_words)
```

The `stop_words` dataset in the `tidytext` package contains stop words from three lexicons. We can use them all together, as we have here, or `filter()` to only use one set of stop words if that is more appropriate for a certain analysis.

We can also use `dplyr`'s `count()` to find the most common words in all the books as a whole.

```
tidy_books %>%
  count(word, sort = TRUE)
#> # A tibble: 13,914 x 2
#>   word     n
#>   <chr> <int>
#> 1 miss    1855
#> 2 time    1337
#> 3 fanny   862
#> 4 dear    822
#> 5 lady    817
#> 6 sir     806
#> 7 day     797
#> 8 emma    787
#> 9 sister   727
#> 10 house   699
#> # ... with 13,904 more rows
```

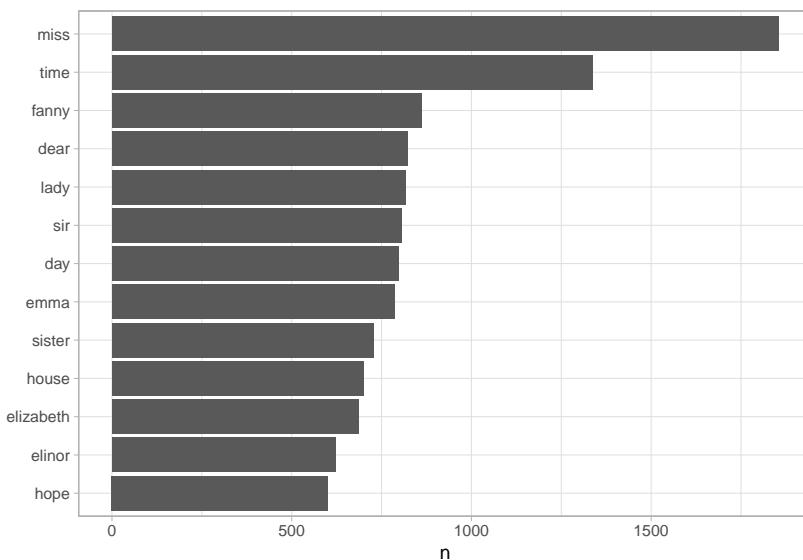
Because we've been using tidy tools, our word counts are stored in a tidy data frame. This allows us to pipe this directly to the `ggplot2` package, for example to create a visualization of the most common words (Figure 1.1).

```
library(ggplot2)

tidy_books %>%
```

² <https://github.com/ropensci/tokenizers>

```
count(word, sort = TRUE) %>%
filter(n > 600) %>%
mutate(word = reorder(word, n)) %>%
ggplot(aes(n, word)) +
geom_col() +
labs(y = NULL)
```



1.1. ábra. The most common words in Jane Austen’s novels

Note that the `austen_books()` function started us with exactly the text we wanted to analyze, but in other cases we may need to perform cleaning of text data, such as removing copyright headers or formatting. You’ll see examples of this kind of pre-processing in the case study chapters, particularly Chapter ??.

1.4. The gutenbergr package

Now that we’ve used the `janeaustenr` package to explore tidying text, let’s introduce the `gutenbergr`³ package (?). The `gutenbergr` package provides access to the public domain works from the Project Gutenberg⁴ collection. The package includes tools both for downloading books (stripping out the unhelpful header/footer information), and a complete dataset of Project Gutenberg metadata that can be used to find works of interest. In this book, we will mostly use the function `gutenberg_download()` that downloads one or more works

³ <https://github.com/ropensci/gutenbergr>

⁴ <https://www.gutenberg.org/>

from Project Gutenberg by ID, but you can also use other functions to explore metadata, pair Gutenberg ID with title, author, language, etc., or gather information about authors.

Szint 1



- kjdshflksdhjksdhf
- sdfaj adqweqweqwe akjlkjf

To learn more about `gutenbergr`, check out the package's documentation at rOpenSci⁵, where it is one of rOpenSci's packages for data access.



To learn more about `gutenbergr`, check out the package's documentation at rOpenSci⁶, where it is one of rOpenSci's packages for data access.



To learn more about `gutenbergr`, check out the package's documentation at rOpenSci⁷, where it is one of rOpenSci's packages for data access.



To learn more about `gutenbergr`, check out the package's documentation at rOpenSci⁸, where it is one of rOpenSci's packages for data access.



To learn more about `gutenbergr`, check out the package's documentation at rOpenSci⁹, where it is one of rOpenSci's packages for data access.

1.5. Word frequencies

A common task in text mining is to look at word frequencies, just like we have done above for Jane Austen's novels, and to compare frequencies across different texts. We can do this intuitively and smoothly using tidy data principles. We already have Jane Austen's works; let's get two more sets of texts to compare to. First, let's look at some science fiction and fantasy novels by H.G. Wells, who lived in the late 19th and early 20th centuries. Let's get *The Time Machine*¹⁰, *The War of the Worlds*¹¹, *The Invisible Man*¹², and *The Island of Doctor*

¹⁰ <https://www.gutenberg.org/ebooks/35>

¹¹ <https://www.gutenberg.org/ebooks/36>

¹² <https://www.gutenberg.org/ebooks/5230>

*Moreau*¹³. We can access these works using `gutenberg_download()` and the Project Gutenberg ID numbers for each novel.

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

1.6. Summary

In this chapter, we explored what we mean by tidy data when it comes to text, and how tidy data principles can be applied to natural language processing. When text is organized in a format with one token per row, tasks like removing stop words or calculating word frequencies are natural applications of familiar operations within the tidy tool ecosystem. The one-token-per-row framework can be extended from single words to n-grams and other meaningful units of text, as well as to many other analysis priorities that we will consider in this book.

¹³ <https://www.gutenberg.org/ebooks/159>

2. fejezet

Thidy text format



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

3. fejezet

Thidy text format

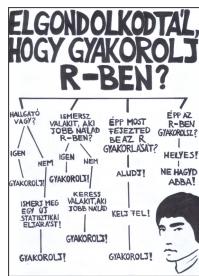


Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

4. fejezet

Thidy text format



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

5. fejezet

Thijhhdy text formatnnn



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

6. fejezet

Thijaaahdy text formatnnnsss



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

7. fejezet

Thijaashdy text formatnnnsss



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

8. fejezet

Thijaashdsdfsy text formatnnnsss

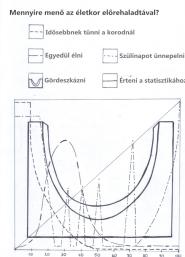
Számolj mintaadatlagot Excel-ben!	
Számolj mintaadatlagot SPSS-ben!	
Számolj mintaadatlagot R-ben!	
Számolj mintaadatlagot jamovi-ban!	
Valójában a jamovi is R-ben számol!	

Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

9. fejezet

Thijaashdsdfsy text formatnnnsss

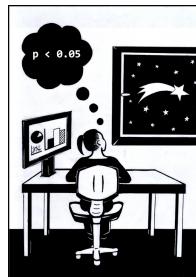


Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

10. fejezet

Thijaashdsdfs text formatnnnsss

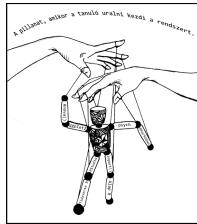


Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

11. fejezet

Thijaashdswdfsystext formatnnnsss



Using tidy data principles is a powerful way to make handling data easier and more effective, and this is no less true when it comes to dealing with text. As described by Hadley Wickham (?), tidy data has a specific structure:

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table