Text mining with Tidy Text

First things first: Instructions

- You can find the exercises – with code to complete - in the RMarkdown file called tidytext_exercises
- The PPT slide titles corresponding to the exercises are marked with # in **BLUE**
- You can find the coding solutions to the exercises in the RMarkdown file called tidytext_codingsolutions
- There are also solution slides, titled tidytext_solutions part [X]. Those include code, but also the output of your analyses, like tibbles and visualizations. Interpretive questions encountered on the slides are also answered here.

Part 1: The tidy text format

- Text mining based on tidy data principles

- The specific structure of tidy data:
  - Each variable is a column
  - Each observation is a row
  - Each type of observational unit is a table

The tidy text format = a table with one-token-per row

What is a token when we text mine?

- A meaningful unit of text, like…
  - A single word
  - An n-gram
  - A sentence
  - A paragraph

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.

Tidy format a text corpus I

Exercise 7

Let's try and turn this poem by Emily Dickinson into a tidy text dataset!

First, install the Tidyverse packages you need during the afternoon session; they are listed in your RStudio environment in the second code block (this might take a while).

7a. Now let's start tidy-texting Dickinson's poem. Run the following lines of code:

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

← The output you will get 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.

Tidy format a text corpus II

7b. Let's call on a package from the Tidyverse that will give us the right data frame: dplyr. You can call on this package by running the following code:

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

← The output you will get is a data frame we call a 'tibble', which is great for use with tidy tools.

Tidy format a text corpus III

We now need to ensure that we can 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.

Tidy format a text corpus IV

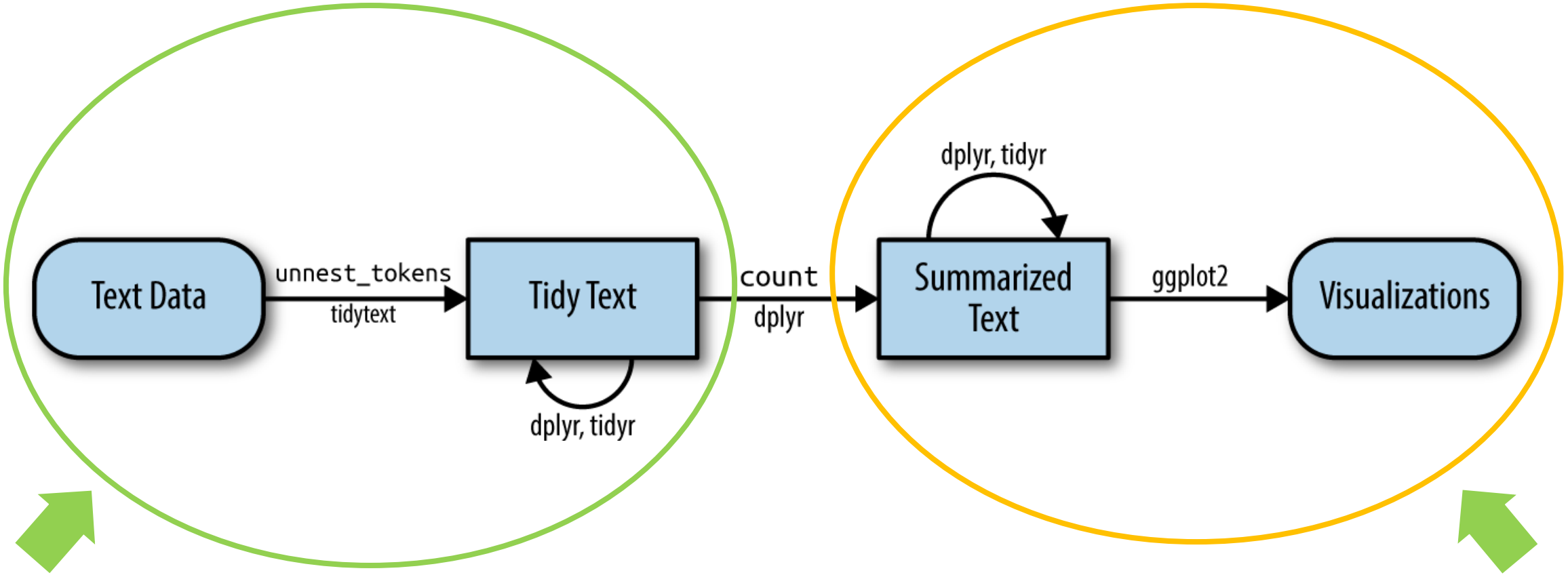
7c. We will now break the text into individual tokens (tokenization) *and* transform it to a tidy data structure. To do this, call on the unnest_tokens() function:

```
library(tidytext)

text_df %>%
  unnest_tokens(word, text)
```

Now check the output! Do you see a simple table with line numbers (left) and words (right)? Then you have tokenized and structured your data for text mining within the Tidyverse 👍

What can we do with Tidy text data?



Been there, done that! …But can you do it again? We will see in a minute!

There is still more fun and useful stuff to learn, like counting words and visualizing your findings. We will get there soon…

Tidying Jane Austen's novels I

Exercise 8

Let's move from one short poem to six novels and transform them into a tidy format. We will use the janeaustenr package, which provides all of 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.

8a. Based on the previous exercises with Dickenson's poem, are you now able to call on the janeaustenr package, as well as the dplyr and stringr packages needed for your analysis?

Tidying Jane Austen's novels II

We will now go on to 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 al the chapters are.

8b. Run the following code… and then challenge yourself with exercise 8c on the next slide!

```
original_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                                     regex("^chapter [\\divxlc]",
                                           ignore_case = TRUE)))) %>%
  ungroup()

original_books
```

Tidying Jane Austen's novels III

8c. You are now ready to tokenize your dataset again! Can you fill in the blanks in this next piece of code so that we will get a **one-token-per-row** format for Austen's novels?

```
library(???)
tidy_books <- original_books %>%
  ???_???(word, text)

tidy_books
```

Tidying Jane Austen's novels IV

Our data is now neatly structured in a one-word-per-row format. Often in text analysis, we will want to remove stop words that are not useful for analysis, typically extremely common words such as "the", "of", "to", and so forth in English. We can remove stop words with an anti_join(), kept in the tidytext dataset stop_words.

8d. This is the first time you call on a dataset from a package. Can you call on the anti_join() function by completing the code below? Note: The code block does not stop at #, so make sure you run the lines of code following " # [sentence]" as well!

```
data(stop_words)

tidy_books <- tidy_books %>%
  ???_???(???_???)

# 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)
```

Tidying Jane Austen's novels V

Now we have removed the stop words, we are ready to go find the most common words in all the books as a whole. We can use dplyr's count() to find those Most Common Words or MCW (also know as Most Frequent Words or MFW).

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.

8e. Let's build ourselves a pipeline! Run this code and see what happens…

```
library(ggplot2)


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

← If all went well, this pipeline just made a beautiful plot appear, showing you the MCW in Austen's novels. Victory!

However, what does all of this counting and plotting tell us? Let's find out during exercise 9!