Package ‘ggpage’

October 13, 2022

Type Package

Title Creates Page Layout Visualizations

Version 0.2.3

Description Facilitates the creation of page layout visualizations in which words are represented as rectangles with sizes relating to the length of the words. Which then is divided in lines and pages for easy overview of up to quite large texts.

License MIT + file LICENSE

URL https://github.com/EmilHvitfeldt/ggpage

BugReports https://github.com/EmilHvitfeldt/ggpage/issues

Depends R (>= 3.0.0)

Imports dplyr, ggplot2 (>= 2.0.0), magrittr, purrr, rlang, stringr, tidytext (>= 0.1.0)

Suggests covr, knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre]

Maintainer Emil Hvitfeldt <emilhhvitfeldt@gmail.com>

Repository CRAN

Date/Publication 2019-06-13 20:30:03 UTC

R topics documented:

`break_help` *Repeating of indexes*

Description

Repeating of indexes

Usage

```
break_help(x)
```

Arguments

`x`	Numerical, vector.

Value

Numerical.

Examples

```
break_help(c(1, 2, 3))
break_help(c(6, 8, 23, 50))
```

ggpage_build *Creates a data frame for further analysis and plotting*

Description

This function can be used in combination with `ggpage_plot` to get the same result as `ggpage_quick`. However by splitting the data.frame construction and plotting we are able to do intermediate analysis which can be included in the visualization.

Usage

```
ggpage_build(book, lpp = 25, character_height = 3,
  vertical_space = 1, x_space_pages = 10, y_space_pages = 10,
  nrow = NULL, ncol = NULL, bycol = TRUE, wtl = NULL,
  para.fun = NULL, page.col = NULL, align = "left", line.max = 80,
  ...)
```

Arguments

`book`	Character or data.frame. Can either have each element be a separate line or having each element being separate words.
`lpp`	Numeric. Lines Per Page. Number of lines allocated for each page.
`character_height`	Numeric. Relative size of the height of each letter compared to its width.
`vertical_space`	Numeric. Distance between each lines vertically.
`x_space_pages`	Numeric. Distance between pages along the x-axis.
`y_space_pages`	Numeric. Distance between pages along the y-axis.
`nrow`	Numeric. Number of rows of pages, if omitted defaults to square layout.
`ncol`	Numeric. Number of columns of pages, if omitted defaults to square layout.
`bycol`	Logical. If TRUE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.
`wtl`	logical. If TRUE will convert single word vector into a vector with full lines. (defaults to FALSE).
`para.fun`	Function that generates random numbers to determine number of word in each paragraph.
`page.col`	column to split the pages by.
`align`	Type of line alignment. Must be one of "left", "right" or "both".
`line.max`	Maximal number of characters per line. Defaults to 80.
`...`	Extra arguments.

Details

The text MUST be presented in a column named text.

Value

'data.frame' containing the following columns:

- 'word': Character. The words of the text.
- 'page': Integer. Page number.
- 'line': Integer. Line number within the page.
- 'xmin': Numeric. Border of rectangle, used by `ggpage_plot` do not alter.
- 'xmax': Numeric. Border of rectangle, used by `ggpage_plot` do not alter.
- 'ymin': Numeric. Border of rectangle, used by `ggpage_plot` do not alter.
- 'ymax': Numeric. Border of rectangle, used by `ggpage_plot` do not alter.

Examples

```
library(dplyr)
library(stringr)
library(ggplot2)
library(tidytext)
library(ggpage)
# build and plot
## data.frame with full lines
ggpage_build(tinderbox) %>%
  ggpage_plot()
## vector with full lines
ggpage_build(book = tinderbox %>%
  pull(text)) %>%
  ggpage_plot()
## data.frame with single words
ggpage_build(tinderbox) %>%
  unnest_tokens(text, word) %>%
  ggpage_plot()
## vector with single words
ggpage_build(tinderbox %>%
  unnest_tokens(text, text) %>%
  pull(text)) %>%
  ggpage_plot()

# nrow and ncol
ggpage_build(tinderbox, nrow = 2) %>%
  ggpage_plot()
ggpage_build(tinderbox, ncol = 2) %>%
  ggpage_plot()

# Include analysis within
ggpage_build(tinderbox) %>%
  mutate(word_length = str_length(word)) %>%
  ggpage_plot(aes(fill = word_length))
```

ggpage_plot *Creates a visualization from the ggpage_build output*

Description

Creates a visualization from the ggpage_build output

Usage

```
ggpage_plot(data, mapping = ggplot2::aes(), paper.show = FALSE,
  paper.color = "grey90", paper.alpha = 1, paper.limits = 3,
  page.number = character(1), page.number.x = 3, page.number.y = 3)
```

Arguments

`data`	data.frame. Expects output from `ggpage_build` with optional intermediate analysis.
`mapping`	Default list of aesthetic mappings to use for plot to be handed to internal `ggplot` call.
`paper.show`	Shows the paper underneath the text.
`paper.color`	Color of the pages. Needs to be of length 1 or the same as the number of pages.
`paper.alpha`	Alpha of the pages. Needs to be of length 1 or the same as the number of pages.
`paper.limits`	Numerical. Extends the edges of the paper in all directions.
`page.number`	Position of the page number. Defaults to none.
`page.number.x`	Distance the page number is pushed away from the text along the x-axis.
`page.number.y`	Distance the page number is pushed away from the text along the y-axis.

Value

A ggplot object with the given visualization.

Examples

```
library(dplyr)
library(stringr)
library(ggplot2)
library(tidytext)
library(ggpage)
# build and plot
## data.frame with full lines
ggpage_build(tinderbox) %>%
  ggpage_plot()
## vector with full lines
ggpage_build(book = tinderbox %>%
  pull(text)) %>%
  ggpage_plot()
## data.frame with single words
ggpage_build(tinderbox) %>%
  unnest_tokens(text, word) %>%
  ggpage_plot()
## vector with single words
ggpage_build(tinderbox %>%
  unnest_tokens(text, text) %>%
  pull(text)) %>%
  ggpage_plot()

# nrow and ncol
ggpage_build(tinderbox, nrow = 2) %>%
  ggpage_plot()
ggpage_build(tinderbox, ncol = 2) %>%
  ggpage_plot()

# Include analysis within
ggpage_build(tinderbox) %>%
  mutate(word_length = str_length(word)) %>%
  ggpage_plot(aes(fill = word_length))
```

ggpage_quick *Creates a quick visualization of the page layout*

Description

Creates a quick visualization of the page layout

Usage

```
ggpage_quick(book, lpp = 25, character_height = 3,
  vertical_space = 1, x_space_pages = 10, y_space_pages = 10,
  nrow = NULL, ncol = NULL, bycol = TRUE)
```

Arguments

book	Character or data.frame. Can either have each element be a separate line or having each element being separate words.
lpp	Numeric. Lines Per Page. Number of lines allocated for each page.
character_height	Numeric. Relative size of the height of each letter compared to its width.
vertical_space	Numeric. Distance between each lines vertically.
x_space_pages	Numeric. Distance between pages along the x-axis.
y_space_pages	Numeric. Distance between pages along the y-axis.
nrow	Numeric. Number of rows of pages, if omitted defaults to square layout.

`ncol` Numeric. Number of columns of pages, if omitted defaults to square layout.

`bycol` Logical. If TRUE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.

Value

A ggplot object with the given visualization.

Examples

```
library(dplyr)
library(stringr)
library(ggplot2)
library(tidytext)
library(ggpage)
# quick
## data.frame with full lines
ggpage_quick(tinderbox)
## vector with full lines
ggpage_quick(tinderbox %>%
               pull(text))
## data.frame with single words
ggpage_quick(tinderbox %>%
               unnest_tokens(text, text))
## vector with single words
ggpage_quick(tinderbox %>%
               unnest_tokens(text, text) %>%
               pull(text))

# nrow and ncol
ggpage_quick(tinderbox, nrow = 2)
ggpage_quick(tinderbox, ncol = 2)
```

`line_align` *Adjust lines*

Description

Adjust lines

Usage

```
line_align(line, max_length, type)
```

Arguments

`line`	data.frame
`max_length`	numerical. number of letters allowed on a line.
`type`	Type of line alignment. Must be one of "left", "right" or "both".

Value

data.frame

`nest_paragraphs` *converts paragraph tokens into line tokens*

Description

extents the str_wrap() function from the stringr package to work with longer strings.

Usage

```
nest_paragraphs(data, input, ...)
```

Arguments

`data`	data.frame. With one paragraph per row.
`input`	column that gets split as string or symbol.
`...`	Extra arguments passed to str_wrap.

Value

data.frame.

`page_liner` *Add line number within pages*

Description

Add line number within pages

Usage

```
page_liner(data)
```

Arguments

`data`	data.frame

Value

data.frame

`paper_shape` *Identify the edges of the paper of each page*

Description

Identify the edges of the paper of each page

Usage

```
paper_shape(data)
```

Arguments

`data`	data.frame created by ggpage_build.

Value

data.frame,

Examples

```
paper_shape(ggpage_build(tinderbox))
```

`para_index` *paragraph split*

Description

Converts a word vector into a line vector with variable paragraph lengths.

Usage

```
para_index(n, FUN, ...)
```

Arguments

`n`	Numeric. Numbers of words.
`FUN`	Numeric. how many words to split whole string by.
`...`	Extra arguments.

Details

FUN most be a function that takes in a number n and returns a vector of natural numbers.

Value

Numeric. paragraph indicator.

tinderbox *The tinder-box by H.C. Andersen*

Description

A tidy data.frame containing the entire story of The tinder-box by H.C. Andersen with two columns: `text` which contains the text of the fairy tale divided into elements of up to about 80 characters each and `book` giving the name of the fairy tale in question.

Usage

```
tinderbox
```

Format

A data frame with 211 rows and 2 variables:

text character string up to 80 characters each

book name of the fairy tale ...

tinderbox_paragraph *The tinder-box by H.C. Andersen*

Description

A tidy data.frame containing the entire story of The tinder-box by H.C. Andersen with two columns: `text` which contains the text of the fairy tale divided into paragraphs.

Usage

```
tinderbox_paragraph
```

Format

A data frame with 11 rows and 1 variables:

text character string up to 80 characters each ...

word_to_line *Internal function for converting words to lines*

Description

extents the str_wrap() function from the stringr package to work with longer strings.

Usage

```
word_to_line(words, wot_number = 1000)
```

Arguments

`words`	data.frame. Where each row is a separate word words with the column name text.
`wot_number`	Numeric. how many words to split whole string by.

Value

Character. have each element be a separate line.

Index