R topics documented:

- community_data ............................................. 2
- length_omit_na ............................................. 3
- richest_in_nigeria .......................................... 3
- str_engleque .................................................... 4
- str_extract_part ............................................. 5
- str_left .......................................................... 6
- str_mid ............................................................ 7
- str_right .......................................................... 7
- str_rm_whitespace_df ........................................... 8
- str_split_extract ............................................. 9
- str_title_case .................................................... 9

Index 11

<table>
<thead>
<tr>
<th>community_data</th>
<th>Data containing whitespaces</th>
</tr>
</thead>
</table>

Description

This survey data was collected using a Google form to demonstrate how the `str_rm_whitespace_df()` function in the forstringr package could be used to eliminate whitespace.

Usage

community_data

Format

A data frame with 32 rows and 8 variables:

- **Date** Form submission date
- **First_name** First name of the respondent
- **Gender** The gender of the respondent
- **State** State or province living
- **Degree** Whether or not the respondent has a degree
- **Year** The year of graduation from a college
- **Use_R** Whether respondent used R for data science or not
- **Community** The data science community the respondent is associated with

Source

*Ezekiel* and *Esther* developed the Google form that was used to collect the data. By clicking the following link, you may also add to the data:

https://docs.google.com/forms/d/e/1FAIpQLSeAhIBaze-pTHgHyIKDZEx5kDuke0oYv0YPq4gtGKijHSaUg/viewform
length_omit_na

Description

length_omitna() counts only non-missing elements of a vector.

Usage

length_omit_na(x)

Arguments

x  Input vector. Either a vector, or something coercible to one.

Value

An integer

See Also

length() counts all the elements in a vector including those that are missing (NAs).

Examples

ethnicity <- c("Hausa", NA, "Yoruba", "Igbo", NA, "Fulani", "Kanuri", "Others")
length_omit_na(ethnicity)
length(ethnicity)

richest_in_nigeria

Description

A dataset containing the list of top ten billionaires in Nigeria.

Usage

richest_in_nigeria
Format

A data frame with 10 rows and 5 variables:

- **Rank**: rank from 1 to 10
- **Name**: full name of the billionaires
- **Net worth**: net worth in billion dollars
- **Age**: the current age of billionaires
- **Source of Wealth**: the origin of the billionaires’ entire body of wealth

Source

https://rnn.ng/richest-men-in-nigeria/

---

`str_englue`  
*Dynamic plot labels using glue operators*

Description

`str_englue()` helps you solve the labeling problem during plotting. For example, any value wrapped in `{ }` will be inserted into the string and it can also understands embracing, `{{ }}`, which automatically inserts a given variable name.

Usage

`str_englue(x, env, error_call, error_arg)`

Arguments

- **x**: A string to interpolate with glue operators.
- **env**: User environment where the interpolation data lives in case you’re wrapping `englue()` in another function.
- **error_call**: The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the call argument of `abort()` for more information.
- **error_arg**: An argument name as a string. This argument will be mentioned in error messages as the input that is at the origin of a problem.

See Also

`rlang::englue()`
library(ggplot2)

histogram_plot <- function(df, var, binwidth) {
  ggplot(df, aes(x = {{ var }})) +
  geom_histogram(binwidth = binwidth) +
  labs(title = str_glue("A histogram of {{var}} with binwidth {binwidth}"))
}

histogram_plot(iris, Sepal.Length, binwidth = 0.1)

---

### str_extract_part

**Extract strings before or after a given pattern**

#### Description

Vectorised over string and pattern.

#### Usage

```r
str_extract_part(string, pattern, before = TRUE)
```

#### Arguments

- `string`: A character vector.
- `pattern`: Pattern to look for.
- `before`: The position in the string to extract from. If TRUE, the extract will occur before the pattern; if FALSE, it will happen after the pattern.

#### Value

A subset of the input vector.

#### See Also

`str_split_extract()` which splits up a string into pieces and extracts the results using a specified index position.
Examples

```r
weekdays <- c("Monday_1", "Tuesday_2", "Wednesday_3", "Thursday_4", "Friday_5", "Saturday_6", "Sunday_7")

str_extract_part(weekdays, before = TRUE, pattern = ".")

str_extract_part(c("$159", "$587", "$897"), before = FALSE, pattern = "$")
```

---

**str_left**

*Returns a substring from the beginning of a specified string*

Description

Given a character vector, `str_left()` returns the left side of a string.

Usage

```r
str_left(string, n = 1)
```

Arguments

- `string` The character from which the left portion will be returned.
- `n` Optional. The number of characters to return from the left side of string

Value

A character vector

See Also

- `str_right()` which extracts characters from the right and `str_mid()` which returns a segment of character strings.

Examples

```r
str_left("Nigeria")
str_left("Nigeria", n = 3)
str_left(c("Female", "Male", "Male", "Female"))
```
**str_mid**

Returns a segment of character strings

**Description**

str_mid() returns a specific number of characters from a text string, starting at the position you specify, based on the number of characters you specify.

**Usage**

str_mid(string, start, n)

**Arguments**

- **string**: The text string containing the characters you want to extract.
- **start**: The position of the first character you want to extract in the text. The first character in text has start = 1, and so on.
- **n**: The length of character to extract.

**Value**

A character vector.

**See Also**

str_left() which extracts characters from the left and str_right() which extracts characters from the right.

**Examples**

str_mid("Super Eagle", 7, 5)

str_mid("Oyo Ibadan", 5, 6)

---

**str_right**

Returns a substring from the end of a specified string

**Description**

Given a character vector, str_right() returns the right side of a string.

**Usage**

str_right(string, n = 1)
**Arguments**

- **string**: The character from which the right portion will be returned.
- **n**: Optional. The number of characters to return from the right side of string.

**Value**

A character vector.

**See Also**

- `str_left()` which extracts characters from the left and `str_mid()` which returns a segment of character strings.

**Examples**

```r
str_right("Sale Price")
str_right("Sale Price", n = 5)
```

---

**str_rm_whitespace_df**  
Remove extra spaces in a data frame

**Description**

`str_rm_whitespace_df()` removes all leading, trailing, and collapses multiple consecutive white spaces in non-numerical variables in a data frame.

**Usage**

```r
str_rm_whitespace_df(df)
```

**Arguments**

- **df**: A data frame or data frame extension (e.g. a tibble) with leading or trailing spaces.

**Value**

A clean data frame with no leading or trailing spaces.

**Examples**

```r
richest_in_nigeria
str_rm_whitespace_df(richest_in_nigeria)
```
**str_split_extract**

Extract the result of a positional split string

**Description**

Split up a string into pieces and extract the results using a specific index position. Mathematically, you can interpret it as follows:

Given a character string, S, extract the element at a given position, k, from the result of splitting S by a given pattern, m.

**Usage**

```r
str_split_extract(string, pattern, position)
```

**Arguments**

- **string**: Input vector. Either a character vector, or something coercible to one.
- **pattern**: Pattern to look for. This may also contain regular expression.
- **position**: Index position to return from the character vector.

**Value**

A character vector.

**Examples**

```r
code <- c("HS-IB-EDB", "OG-OYO-CAS-0121", "NY-ILR-NIG-036")
str_split_extract(code, "-", 1)
str_split_extract(code, "-", 4)
```

**str_title_case**

Convert string to title case

**Description**

`str_title_case()` converts string to title case, capitalizing only the first letter of each word while ignoring articles, prepositions, and conjunctions

**Usage**

```r
str_title_case(string)
```
Arguments

string Input vector. Either a character vector, or something coercible to one.

Details

Please note that `str_title_case()` is different from `stringr::str_to_title()` which converts to title case, where only the first letter of each word is capitalized.

Value

A character vector the same length as the string and in title case.

Examples

```r
words <- "the quick brown fox jumps over a lazy dog"
str_title_case(words)
str_to_title(words)

words <- "A journey through the history of music"
str_title_case(words)
str_to_title(words)
```
Index

* datasets
  community_data, 2
  richest_in_nigeria, 3

abort(), 4

community_data, 2

length(), 3
length_omit_na, 3

NA, 3

richest_in_nigeria, 3
rlang::englue(), 4

strenglue, 4
str_extract_part, 5
str_left, 6
str_left(), 7, 8
str_mid, 7
str_mid(), 6, 8
str_right, 7
str_right(), 6, 7
str_rm_whitespace_df, 8
str_split_extract, 9
str_split_extract(), 5
str_title_case, 9
stringr::str_to_title(), 10