Package ‘tidyplus’

October 14, 2022

Title  Additional ‘tidyverse’ Functions
Version  0.0.1
Description  Provides functions such as str_crush(), add_missing_column(),
  coalesce_data() and drop_na_all() that complement ‘tidyverse’ functionality
  or functions that provide alternative behaviors such as if_else2()
  and str_detect2().
License  MIT + file LICENSE
Depends  R (>= 3.6)
Imports  chk, dplyr, rlang, stringi, stringr, tibble, tidyr,
  tidyselect, vctrs
Suggests  covr, readr, sf, testthat (>= 3.0.0)
Config/testthat/edition  3
Encoding  UTF-8
Language  en-US
RoxygenNote  7.2.1
NeedsCompilation  no
Author  Joe Thorley [aut, cre] (<https://orcid.org/0000-0002-7683-4592>),
  Poisson Consulting [cph, fnd]
Maintainer  Joe Thorley <joe@poissonconsulting.ca>
Repository  CRAN
Date/Publication  2022-08-29 09:20:02 UTC

R topics documented:

  add_missing_column .................................................  2
  coalesce_data .......................................................  3
  collapse_comments ..................................................  4
  drop_na_all ...........................................................  5
  drop_uninformative_columns .......................................  5
  if_else2 .............................................................  6
  only .................................................................  7
add_missing_column

This is a convenient way to add one more columns (if not already present) to an existing data frame. It is useful to ensure that all required columns are present in a data frame.

usage

add_missing_column(
  .data,
  ..., 
  .before = NULL, 
  .after = NULL,
  .name_repair = c("check_unique", "unique", "universal", "minimal")
)

arguments

.data  Data frame to append to.
...    <dynamic-dots> Name-value pairs, passed on to tibble(). All values must have the same size of .data or size 1.
.before, .after  One-based column index or column name where to add the new columns, default: after last column.
.name_repair  Treatment of problematic column names:
  • "minimal": No name repair or checks, beyond basic existence,
  • "unique": Make sure names are unique and not empty,
  • "check_unique": (default value), no name repair, but check they are unique,
  • "universal": Make the names unique and syntactic
  • a function: apply custom name repair (e.g., .name_repair = make.names for names in the style of base R).
  • A purrr-style anonymous function, see rlang::as_function()

This argument is passed on as repair to vctrs::vec_as_names(). See there for more details on these terms and the strategies used to enforce them.

details

It is wrapper on tibble::add_column() that doesn’t error if the column is already present.
**Value**
The original data frame with missing columns added if not already present.

**See Also**
`tibble::add_column()`

**Examples**
```r
data <- tibble::tibble(x = 1:3, y = 3:1)
tibble::add_column(data, z = -1:1, w = 0)
add_missing_column(data, z = -1:1, .before = "y")

# add_column errors if already present
try(tibble::add_column(data, x = 4:6))

# add_missing_column silently ignores
add_missing_column(data, x = 4:6)
```

---

**coalesce_data**  

**Coalesce Data**

**Description**
Coalesce values in multiple columns by finding the first non-missing value at each position. Coalesced columns are removed.

**Usage**
```r
coaalesce_data(x, coalesce = list(), quiet = FALSE)
```

**Arguments**
- `x` A data frame.
- `coalesce` A uniquely named list of character vectors where the names are the new column names and the values are the names of the columns to coalesce. If a single value is provided for a column it is treated as a regular expression.
- `quiet` A flag specifying whether to provide messages.

**Details**
Coalescence is performed in the order specified in the coalesce argument such that a column produced by coalescence can be further coalesced.

**Value**
The original data frame with one or more columns coalesced into a new column.
collapse_comments

See Also

dplyr::coalesce()

Examples

data <- data.frame(x = c(1, NA, NA), y = c(NA, 3, NA), z = c(7, 8, 9), a = c(4, 5, 6))
coalesce_data(data, list(b = c("x", "y")), quiet = TRUE)
coalesce_data(data, list(z = c("y", "x"), d = c("z", "a")))

data <- data.frame(x = c(1, NA, NA), y = c(NA, 3, NA), z = c(7, 8, 9), a = c(4, 5, 6))
coalesce_data(data, list(b = c("x", "y")), quiet = TRUE)
coalesce_data(data, list(z = c("y", "x"), d = c("z", "a")))

Description

Collapse comments coercing each element to a string (character scalar) and then collapsing into a single string using the ‘.’ separator.

Usage

collapse_comments(...)

Arguments

... objects to be collapsed into a string.

Value

A string of the collapsed comments.

See Also

unite_str()

Examples

collapse_comments("Saw fish", character(0), "Nice. ", NA_character_)

data <- data.frame(
  visit = c(1,1,2, 2),
  fish = 1:4,
  comment = c("Sunny day. ", "Skinny fish", "Lost boot", NA))
data |> dplyr::group_by(visit) |> dplyr::summarise(comment = collapse_comments(comment)) |> dplyr::ungroup()
**drop_na_all**  
*Drop rows containing all missing values*

**Description**  
This is a convenient way to drop uninformative rows from a data frame.

**Usage**  
drop_na_all(data, ...)

**Arguments**
- `data`  
  A data frame.
- `...`  
  Columns to inspect for missing values. If empty, all columns are used.

**Value**  
The original data frame with rows for which all values are missing dropped.

**See Also**
- `tidyr::drop_na`  
- `drop_uninformative_columns`

**Examples**

data <- tibble::tibble(  
a = c(NA, NA, NA),  
b = c(1, 1, NA),  
c = c(2, NA, NA))

drop_na_all(data)  
drop_na_all(data, a, c)

**drop_uninformative_columns**  
*Drop uninformative columns from a data frame*

**Description**  
This is a convenient way to drop columns which all have one value (missing or not) or if `na_distinct = FALSE` also drop columns which all have one value and/or missing values.

**Usage**  
drop_uninformative_columns(data, na_distinct = TRUE)
if_else2

Arguments

data A data frame.

na_distinct A flag specifying whether to treat missing values as distinct from other values.

Value

The original data frame with only informative columns.

Examples

data <- tibble::tibble(
  a = c(1,1,1), x = c(NA, NA, NA), b = c(1, 1, NA),
  z = c(1, 2, 2), e = c(1, 2, NA))

drop_uninformative_columns(data)
drop_uninformative_columns(data, na_distinct = FALSE)

if_else2 Vectorised if else.

Description

Vectorised if else that if true returns first possibility otherwise returns second possibility (even if the condition is a missing value). When searching character vectors an alternative solution is to use str_detect2().

Usage

if_else2(condition, true, false)

Arguments

c condition Logical vector

t true, false Values to use for TRUE and FALSE values of condition. They must be either the same length as condition, or length 1. They must also be the same type: if_else() checks that they have the same type and same class. All other attributes are taken from true.

Value

Where condition is TRUE, the matching value from true, where it’s FALSE or NA, the matching value from false.

See Also

ifelse() and dplyr::if_else().
Examples

# consider the following data frame
data <- tibble::tibble(
  x = c(TRUE, FALSE, NA),
  y = c("x is false", NA, "hello"))

# with a single vector if_else2() behaves the same as the default call to if_else().
dplyr::mutate(data,
  y1 = dplyr::if_else(y != "x is false", "x is true", y),
  y2 = if_else2(y != "x is false", "x is true", y))

# however in the case of a second vector the use of
# if_else2() does not introduce missing values

dplyr::mutate(data,
  x1 = dplyr::if_else(stringr::str_detect(y, "x is false"), FALSE, x),
  x2 = if_else2(stringr::str_detect(y, "x is false"), FALSE, x))

# in the case of regular expression matching an alternative is to use
# str_detect2()
dplyr::mutate(data,
  x3 = dplyr::if_else(str_detect2(y, "x is false"), FALSE, x))

---

only

Extract the only distinct value from a vector

Description

Extracts the only distinct value from an atomic vector or throws an informative error if no values or multiple distinct values.

Usage

only(x, na_rm = FALSE)

Arguments

x                      An atomic vector.
na_rm                   A flag indicating whether to exclude missing values.

Details

only() is useful when summarizing a vector by group while checking the assumption that it is constant within the group.

Value

The only distinct value from a vector otherwise throws an error.
replace_na_if

Conditional replacement of NAs with specified values

**Description**

Unlike tidyr::replace_na(), it is only defined for vectors.

**Usage**

replace_na_if(x, condition, true)

**Arguments**

- **x**: Vector with missing values to modify.
- **condition**: Logical vector
- **true**: The replacement values where condition is TRUE.

**Details**

replace_na_if() is a wrapper on if_else2(is.na(x) & condition, true, x)

**Value**

A modified version of x that replaces any missing values where condition is TRUE with true.

**See Also**

tidyr::replace_na() and if_else2()
Examples

```r
data <- tibble::tibble(
  x = c(TRUE, FALSE, NA),
  y = c("x is false", NA, "x is false"))
dplyr::mutate(data,
  x1 = tidyr::replace_na(x, FALSE),
  x3 = if_else2(is.na(x) & y == "x is false", FALSE, x),
  x4 = replace_na_if(x, y == "x is false", FALSE))
```

---

**str_crush**

Remove whitespace from a string

Description

str_crush(), which removes all whitespace from a string, is the logical extension to `stringr::str_trim()` and `stringr::str_squish()`.

Usage

```r
str_crush(string)
```

Arguments

- `string`: A character vector.

Details

str_crush() is considered too specialized to be part of stringr.

Value

A character vector.

See Also

`stringr::str_trim()` and `stringr::str_squish()`

Examples

```r
str_crush(" String with trailing, middle, and leading white space\t")
```
str_detect2

Detect the presence or absence of a pattern in a string.

Description

Vectorised over string and pattern. Actually equivalent to `grepl(pattern, x)` as returns FALSE for NAs (unlike `stringr::str_detect()`). This behavior is useful when searching comments many of which are NA to indicate no comments present.

Usage

```r
str_detect2(string, pattern, negate = FALSE)
```

Arguments

- `string`: Input vector. Either a character vector, or something coercible to one.
- `pattern`: Pattern to look for.
  - The default interpretation is a regular expression, as described in `stringi::stringi-search-regex`. Control options with `regex()`.
  - Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate. Generally, for matching human text, you’ll want `coll()` which respects character matching rules for the specified locale.
  - Match character, word, line and sentence boundaries with `boundary()`. An empty pattern, "", is equivalent to `boundary("character")`.
- `negate`: If TRUE, return non-matching elements.

Value

A logical vector.

See Also

`grepl()` and `stringr::str_detect()`

Examples

```r
x <- c("b", NA, "ab")
pattern <- "^a"
grepl(pattern, x)
stringr::str_detect(x, pattern)
str_detect2(x, pattern)
```
**Description**

Convenience function for combining character columns.

**Usage**

```r
unite_str(data, col, ..., sep = ". ", remove = TRUE)
```

**Arguments**

- `data` A data frame.
- `col` The name of the new column, as a string or symbol. This argument is passed by expression and supports **quasiquotation** (you can unquote strings and symbols). The name is captured from the expression with `rlang::ensym()` (note that this kind of interface where symbols do not represent actual objects is now discouraged in the tidyverse; we support it here for backward compatibility).
- `...` <**tidy-select**> Columns to unite
- `sep` Separator to use between values.
- `remove` If TRUE, remove input columns from output data frame.

**Details**

Blank values of "" are converted into missing values.

**Value**

The original data frame with the one or more columns combined as character vectors separated by a period.

**See Also**

`tidyr::unite()` and `collapse_comments()`

**Examples**

```r
data <- tibble::tibble(x = c("good", "Saw fish.", "", NA), y = c("2021", NA, NA, NA))

# unite has poor handling of character vectors
tidyr::unite(data, "new", x, y, remove = FALSE)

unite_str(data, "new", x, y, remove = FALSE)
```
Index

add_missing_column, 2
tidyrr::drop_na, 5
tidyrr::replace_na(), 8
tidyrr::unite(), 11
unite_str, 11
unite_str(), 4
vctrs::vec_as_names(), 2

boundary(), 10

collapse_comments, 4
collapse_comments(), 11
dplyr::coalesce(), 4
dplyr::first(), 8
dplyr::if_else(), 6
drop_na_all, 5
drop_uninformative_columns, 5, 5

fixed(), 10

grepl(), 10

if_else2, 6
if_else2(), 8
ifelse(), 6

only, 7

quasiquotation, 11

regex(), 10
replace_na_if, 8
rlang::as_function(), 2
rlang::ensym(), 11

str_crush, 9
str_detect2, 10
str_detect2(), 6
stringi::stringi-search-regex, 10
stringr::str_detect(), 10
stringr::str_squish(), 9
stringr::str_trim(), 9

tibble(), 2

12