

Package ‘safejoin’

July 23, 2025

Title Perform ``Safe" Table Joins

Version 0.2.0

Description The goal of 'safejoin' is to guarantee that when performing joins extra rows are not added to your data. 'safejoin' provides a wrapper around 'dplyr::left_join' that will raise an error when extra rows are unexpectedly added to your data. This can be useful when working with data where you expect there to be a many to one relationship but you are not certain the relationship holds.

License MIT + file LICENSE

Encoding UTF-8

Suggests testthat, knitr, rmarkdown

Imports dplyr, glue, lifecycle

RoxygenNote 7.2.3

URL <https://github.com/SamEdwardes/safejoin>

BugReports <https://github.com/SamEdwardes/safejoin/issues>

NeedsCompilation no

Author Sam Edwardes [aut, cre]

Maintainer Sam Edwardes <edwardes.s@gmail.com>

Repository CRAN

Date/Publication 2024-06-02 21:10:03 UTC

Contents

safe_left_join	2
Index	4

safe_left_join

Validate extra rows are added not added to the left hand side

Description

[Deprecated] Perform a "safe" left join where it is guaranteed that no additional rows are added to the left hand side table. For more information on left joins see ([dplyr::left_join](#)).

Usage

```
safe_left_join(..., action = "error", relationship = "x:1")
```

Arguments

...

Arguments passed on to [dplyr::left_join](#)

x, y A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.

by A join specification created with [join_by\(\)](#), or a character vector of variables to join by.

If NULL, the default, `*_join()` will perform a natural join, using all variables in common across x and y. A message lists the variables so that you can check they're correct; suppress the message by supplying `by` explicitly. To join on different variables between x and y, use a [join_by\(\)](#) specification. For example, `join_by(a == b)` will match `x$a` to `y$b`.

To join by multiple variables, use a [join_by\(\)](#) specification with multiple expressions. For example, `join_by(a == b, c == d)` will match `x$a` to `y$b` and `x$c` to `y$d`. If the column names are the same between x and y, you can shorten this by listing only the variable names, like `join_by(a, c)`.

[join_by\(\)](#) can also be used to perform inequality, rolling, and overlap joins. See the documentation at [?join_by](#) for details on these types of joins. For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, `by = c("a", "b")` joins `x$a` to `y$a` and `x$b` to `y$b`. If variable names differ between x and y, use a named character vector like `by = c("x_a" = "y_a", "x_b" = "y_b")`.

To perform a cross-join, generating all combinations of x and y, see [cross_join\(\)](#).

copy If x and y are not from the same data source, and `copy` is TRUE, then y will be copied into the same src as x. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.

suffix If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.

keep Should the join keys from both x and y be preserved in the output?

- If NULL, the default, joins on equality retain only the keys from x, while joins on inequality retain the keys from both inputs.
- If TRUE, all keys from both inputs are retained.

- If FALSE, only keys from x are retained. For right and full joins, the data in key columns corresponding to rows that only exist in y are merged into the key columns from x. Can't be used when joining on inequality conditions.

action	What should happen when the number of rows changes from a join? Options include: 'error', 'warning', or 'message'. By default 'error'.
relationship	What is the expected relationship between x and y? At this time the only available option is '*:1', indicating a many to one relationship between x and y. In the future more options may be added.

Value

An object of the same type as x. The order of the rows and columns of x is preserved as much as possible. The output has the following properties:

Examples

```
# The relationship between `x` and `y` is `*:1`. No extra rows will be added
# to the left hand side.
x <- data.frame(key = c("a", "a", "b"), value_x = c(1, 4, 2))
y <- data.frame(key = c("a", "b"), value_y = c(1, 1))
safe_left_join(x, y)

# The relationship between `x` and `y` is `1:*`. An error should be raised
# because additional rows will be added to the left hand side.
## Not run: x <- data.frame(key = c("a", "b"), value_x = c(1, 2))
y <- data.frame(key = c("a", "a"), value_y = c(1, 1))
safe_left_join(x, y)
## End(Not run)

# Alternatively instead of raising an error a warning or message can be
# outputted.
x <- data.frame(key = c("a", "b"), value_x = c(1, 2))
y <- data.frame(key = c("a", "a"), value_y = c(1, 1))
safe_left_join(x, y, action = "warning")
safe_left_join(x, y, action = "message")
```

Index

`?join_by`, [2](#)

`cross_join()`, [2](#)

`dplyr::left_join`, [2](#)

`join_by()`, [2](#)

`safe_left_join`, [2](#)