Package ‘dedupewider’

April 7, 2025

Title Deduplication Across Multiple Columns

Version 0.1.1

Description Duplicated data can exist in different rows and columns and user may need to treat observations (rows) connected by duplicated data as one observation, e.g. companies can belong to one family (and thus: be one company) by sharing some telephone numbers. This package allows to find connected rows based on data on chosen columns and collapse it into one row.

License MIT + file LICENSE

URL https://github.com/gsmolinski/dedupewider

BugReports https://github.com/gsmolinski/dedupewider/issues

Depends R (>= 3.2.0)

Imports methods, data.table (>= 1.12.4)

Suggests rmarkdown, knitr, kableExtra, magrittr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Grzegorz Smoliński [aut, cre]

Maintainer Grzegorz Smoliński <g.smolinski1@gmail.com>

Repository CRAN

Date/Publication 2025-04-07 17:20:01 UTC

Contents

`dedupe_wide` *Dedupe across multiple columns*

Description

Collapse many rows connected by duplicated data (which can exist in different rows and columns) into one, based on data in chosen columns, optionally putting non-consistent data into newly created additional columns.

Usage

```
dedupe_wide(
  x,
  cols_dedupe,
  cols_expand = NULL,
  max_new_cols = NULL,
  enable_drop = TRUE
)
```

Arguments

`x`	A data.frame without column named '....idx' and any column which ends by four dots and number (e.g. 'column....2').
`cols_dedupe`	A character vector of length min. 2 of columns' names in `x` used to dedupe. Deduplicated data from these columns will be saved into new columns, number of which is control by `max_new_cols`.
`cols_expand`	A character vector of columns' names in `x` or `NULL` (means: none except those used to dedupe) indicating columns with data to keep in case of non-consistent data, i.e. unique data from these columns will be saved into new columns, number of which is control by `max_new_cols`.
`max_new_cols`	A numeric vector length 1 or `NULL` (means: without limit) indicating how many new columns can be created to store unique data from columns passed to `cols_dedupe` and each column passed to `cols_expand`. Cannot be lower than 1.
`enable_drop`	A logical vector length 1: should given column be dropped if (after deduplication) contains only missing data (`NA`)? Applicable only to columns used to dedupe.

Details

Columns passed to `cols_dedupe` must be atomic.

Row names will always be removed. If you want to preserve row names, simply put in into separate column. Note that if this column won't be passed to `cols_expand` argument, only the one row name for duplicated rows will be preserved (row name closest to the top of the table).

Although `duplicated` or `unique` treats missing data (`NA`) as duplicated data, this function do not do this (see second example below).

Type of columns passed to `cols_dedupe` will be coerced to the most general type.

Value

If duplicated data found - data.frame with changed columns' names and optionally additional columns (in some cases less columns, depends on `enable_drop` argument). Otherwise data.frame without changes (except row names removed).

Note

To enable parallel computation, call `setDTthreads` before calling `dedupe_wide` function.

Examples

```
x <- data.frame(tel_1 = c(111, 222, 444, 555),
                tel_2 = c(222, 666, 666, 555),
                name = paste0("name", 1:4))
# rows 1, 2, 3 share the same phone numbers

dedupe_wide(x,
           cols_dedupe = c("tel_1", "tel_2"),
           cols_expand = "name")
# first three collapsed into one, for name4 kept only one phone number (555)
# 'name1', 'name2', 'name3' kept in new columns

y <- data.frame(tel_1 = c(777, 888, NA, NA),
                tel_2 = c(888, 777, NA, NA),
                name = paste0("name", 5:8))
# rows 3 and 4 has only missing data

dedupe_wide(y,
           cols_dedupe = c("tel_1", "tel_2"),
           cols_expand = "name")
# first two rows collapsed into one, nothing change for the rest of rows
```

`na_move` *Move* `NA` *across columns or rows*

Description

For chosen columns, move `NA` to right or left (i.e. across columns) or to top or bottom (i.e. across rows).

Usage

```
na_move(data, cols = names(data), direction = "right")
```

Arguments

`data`	A data.frame without column named "....idx".
`cols`	A character vector of columns' names in `data` across which function will be performed. If `NULL`, first column in `data` will be used. By default all columns will be used.
`direction`	A character vector of length 1 indicating where to move `NA`. Can be one of `"top"`, `"right"`,`"bottom"`, `"left"`. If `NULL` and also by default, `"right"` direction will be used.

Value

A data.frame with only these attributes preserved, which are returned by `attributes` function used on object passed to `data` parameter.

Type of columns passed to `cols` will be coerced to the most general type, although sometimes when column will contain only `NA`, that column will be of type logical.

Note

To enable parallel computation, call `setDTthreads` before calling `na_move` function.

Examples

```
data <- data.frame(col1 = c(1, 2, 3),
                   col2 = c(NA, NA, 4),
                   col3 = c(5, NA, NA),
                   col4 = c(6, 7, 8))
data
na_move(data, c("col2", "col3", "col4"), direction = "right")
```

Index