Package: mutagen (via r-universe)

October 5, 2025

Title Extensions to dplyr's mutate

Version 0.1.0

Description Extensions to dplyr's mutate.

License MIT + file LICENSE

URL https://github.com/gvelasq/mutagen, https://gvelasq.github.io/mutagen/

BugReports https://github.com/gvelasq/mutagen/issues

Depends R (>= 3.2)

Imports purrr, rlang

Suggests carrier, covr, dplyr, mirai, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository https://r-multiverse.r-universe.dev

Date/Publication 2025-09-12 22:42:27 UTC

RemoteUrl https://github.com/gvelasq/mutagen

RemoteRef v0.1.0

RemoteSha 2f5b5f43845e26db9aabdf7dc1c7d0c965ca1405

Contents

gen_na_listcol 2
gen_rowcount 3
gen_rowfirst 4
gen_rowlast 5
gen_rowmatch 6
gen_rowmax 7
gen_rowmin 8
gen_rownth 9

Index **10**

`gen_na_listcol` *Generate a list-column with NULLs replaced with NAs*

Description

This function takes a list and replaces all `NULL` values with `NA`. It is useful for working with list-columns in a data frame.

Usage

```
gen_na_listcol(x)
```

Arguments

`x`	A list or list-column to modify.

Details

Parallelization is supported via `purrr::in_parallel()`.

Value

A list with all `NULL` values replaced with `NA`.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <-
  mtcars %>%
  select(cyl, vs, am) %>%
  slice(1:6) %>%
  as_tibble() %>%
  mutate(listcol = list(NULL, "b", "c", "d", "e", "f"))
glimpse(a)
b <-
  a %>%
  mutate(across(starts_with("listcol"), gen_na_listcol))
glimpse(b)
```

gen_rowcount *Generate rowwise count of columns matching a set of values*

Description

This function performs a rowwise count of columns in a data frame that match a set of supplied values.

Usage

```
gen_rowcount(data, cols, values)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.
values	A list of values to match.

Details

Parallelization is supported via purrr::in_parallel().

Value

An integer vector with the number of matched values.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = 1:3,
  y = rep(NA, 3),
  z = letters[1:3],
  aa = rep(FALSE, 3)
)
val <- list(1, NA, "a", FALSE)
gen_rowcount(a, values = val)
gen_rowcount(a, everything(), values = val)
gen_rowcount(a, starts_with(letters[25:26]), values = val)
b <- a %>% mutate(q = gen_rowcount(., values = val))
b
```

gen_rowfirst *Generate first rowwise nonmissing value*

Description

This function returns the first rowwise nonmissing value in a data frame.

Usage

```
gen_rowfirst(data, cols)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.

Details

Parallelization is supported via `purrr::in_parallel()`.

Value

A vector of the first rowwise nonmissing value. The vector's type will be of common type to all rowwise nonmissing values.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5)
)
gen_rowfirst(a)
gen_rowfirst(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rowfirst(.))
b
c <-
  a %>%
  mutate(w = c("a", TRUE, NA), .before = "x") %>%
  mutate(q = gen_rowfirst(.))
c # note that q is of type <chr>
```

gen_rowlast *Generate last rowwise nonmissing value*

Description

This function returns the last rowwise nonmissing value in a data frame.

Usage

```
gen_rowlast(data, cols)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.

Details

Parallelization is supported via `purrr::in_parallel()`.

Value

A vector of the last rowwise nonmissing value. The vector's type will be of common type to all rowwise nonmissing values.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5)
)
gen_rowlast(a)
gen_rowlast(a, all_of(letters[24:25]))
b <- a %>% mutate(q = gen_rowlast(.))
b
c <-
  a %>%
  mutate(aa = c("a", TRUE, NA), .after = "z") %>%
  mutate(q = gen_rowlast(.))
c # note that q is of type <chr>
```

gen_rowmatch *Generate rowwise match of a set of values*

Description

This function performs a rowwise match of a set of supplied values across columns in a data frame. If any of the row values equal one of the supplied values, this function returns an integer 1 (`1L`) for that row, otherwise it returns an integer 0 (`0L`).

Usage

```
gen_rowmatch(data, cols, values)
```

Arguments

`data`	A data frame.
`cols`	<tidy-select> Columns to search across.
`values`	A list of values to match.

Details

Parallelization is supported via `purrr::in_parallel()`.

Value

A binary integer vector indicating whether any supplied value was matched with an integer 1 (`1L`), otherwise it returns an integer 0 (`0L`).

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = 1:3,
  y = rep(NA, 3),
  z = letters[1:3],
  aa = rep(FALSE, 3)
)
val <- list(1, NA, "a", FALSE)
val2 <- list(5, NaN, "d", Inf)
gen_rowmatch(a, values = val)
b <- a %>%
  mutate(
    q = gen_rowmatch(., values = val),
    r = gen_rowmatch(., values = val2)
  )
b
```

gen_rowmax *Generate rowwise maximum value*

Description

This function returns the rowwise maximum value in a data frame.

Usage

```
gen_rowmax(data, cols)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.

Details

Parallelization is supported via purrr::in_parallel().

Value

A vector of the rowwise maximum value.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5)
)
gen_rowmax(a)
gen_rowmax(a, everything())
gen_rowmax(a, starts_with(letters[24:25]))
b <- a %>% mutate(q = gen_rowmax(.))
b
```

gen_rowmin *Generate rowwise minimum value*

Description

This function returns the rowwise minimum value in a data frame.

Usage

```
gen_rowmin(data, cols)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.

Details

Parallelization is supported via purrr::in_parallel().

Value

A vector of the rowwise minimum value.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5)
)
gen_rowmin(a)
gen_rowmin(a, everything())
gen_rowmin(a, starts_with(letters[25:26]))
b <- a %>% mutate(q = gen_rowmin(.))
b
```

gen_rownth *Generate nth rowwise nonmissing value*

Description

This function returns the nth rowwise nonmissing value in a data frame.

Usage

```
gen_rownth(data, cols, n)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across.
n	An integer vector of length 1 that specifies the position of the nth rowwise nonmissing value to search for. A negative integer will index from the end.

Details

Parallelization is supported via purrr::in_parallel().

Value

A vector of the nth rowwise nonmissing value. The vector's type will be of common type to all rowwise nonmissing values.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5)
)
gen_rownth(a, n = 1)
gen_rownth(a, n = 2)
gen_rownth(a, all_of(letters[25:26]), n = 1)
b <- a %>% mutate(q = gen_rownth(., n = 1), r = gen_rownth(., n = 2))
b
c <-
  a %>%
  mutate(w = c("a", TRUE, NA), .before = "x") %>%
  mutate(q = gen_rownth(., n = 1), r = gen_rownth(., n = 2))
c # note that q and r are of type <chr>
```

Index

gen_na_listcol, 2
gen_rowcount, 3
gen_rowfirst, 4
gen_rowlast, 5
gen_rowmatch, 6
gen_rowmax, 7
gen_rowmin, 8
gen_rownth, 9