Package: mutagen (via r-universe)

May 7, 2026

Title Extensions to dplyr's mutate

Version 0.5.0

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 (>= 1.1.0), rlang, stats

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 2025-12-01 01:22:48 UTC

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

RemoteRef v0.5.0

RemoteSha 92ab4ab2561dbc094dc2eea27550686b53430fe2

Contents

`gen_coldiff` *Generate column difference indicator*

Description

This function returns a binary indicator of whether any selected columns are different.

Usage

```
gen_coldiff(data, cols, engine = "identical", ...)
```

Arguments

`data`	A data frame.
`cols`	<`tidy-select`> Columns to search across.
`engine`	A character vector indicating the function used to test for equality. One of either "identical" for `base::identical()` or "all.equal" for `base::all.equal()`.
`...`	Arguments passed onto the function defined in `engine`. For example, the `tolerance` argument can be passed onto `base::all.equal()` to ignore numeric differences smaller than `tolerance`.

Value

An integer vector of value integer 1 (`1L`) for different columns or an integer 0 (`0L`) for identical (or near-equal) columns.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = 1:3,
  y = as.double(1:3),
  z = 1.0001:3.0001
)
```

```
gen_coldiff(a, c(x, y))
b <- a %>%
  mutate(
    q = gen_coldiff(., c(x, y)),
    r = gen_coldiff(., c(x, y), engine = "all.equal"),
    s = gen_coldiff(., c(x, z), engine = "all.equal", tolerance = 1e-4),
    t = gen_coldiff(., c(x, z), engine = "all.equal", tolerance = 1e-5)
  )
b
```

`gen_colpercent` *Generate column percent*

Description

This function calculates a column percent. The `by` argument calculates column percents within unique categories of grouping columns. The `prop` argument calculates a proportion rather than a percent.

Usage

```
gen_colpercent(data, col, by, prop = FALSE)
```

Arguments

`data`	A data frame.
`col`	<tidy-select> A single column with which to calculate a column percent.
`by`	An optional character vector of columns to group by.
`prop`	If `TRUE`, percent will be shown as a proportion between 0-1 rather than a percent between 0-100. Default is `FALSE`.

Value

A double vector totaling 100 within `col`. If grouping columns are specified with `by`, the percent for each unique category of grouping columns will total 100 within `col`. If `prop` is specified, a double vector totaling 1 within `col` (or totaling 1 within unique categories of grouping columns specified with `by`).

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- as_tibble(mtcars)
gen_colpercent(a, gear)
b <-
  a %>%
  select(gear, cyl, carb) %>%
  arrange(gear, cyl, carb) %>%
  mutate(
```

```
    pct1 = gen_colpercent(., gear),
    pct2 = gen_colpercent(., gear, by = "cyl"),
    pct3 = gen_colpercent(., gear, by = c("cyl", "carb")),
    prop1 = gen_colpercent(., gear, prop = TRUE)
  )
b
```

---

gen_listcol_na *Generate list or 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_listcol_na(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_listcol_na))
glimpse(b)
```

gen_rowall *Generate rowwise match of all values*

Description

This function performs a rowwise match of all supplied values across columns in a data frame. If all 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_rowall(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 all supplied values were 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(3, NA, "c", FALSE)
gen_rowall(a, values = val)
b <- a %>%
  mutate(
    q = gen_rowall(., values = val),
    r = gen_rowall(., values = val2)
  )
b
```

gen_rowany *Generate rowwise match of any values*

Description

This function performs a rowwise match of any 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_rowany(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 values were 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_rowany(a, values = val)
b <- a %>%
  mutate(
    q = gen_rowany(., values = val),
    r = gen_rowany(., values = val2)
  )
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 rowwise first nonmissing value*

Description

This function returns the rowwise first 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 rowwise first 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 rowwise last nonmissing value*

Description

This function returns the rowwise last 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 rowwise last 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_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_rowmean *Generate rowwise mean*

Description

This function returns the rowwise arithmetic mean value in a data frame.

Usage

```
gen_rowmean(data, cols)
```

Arguments

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

Details

Parallelization is supported via purrr::in_parallel().

Value

A double vector of the rowwise arithmetic mean value. Missing values are ignored.

Examples

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

gen_rowmedian *Generate rowwise median*

Description

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

Usage

```
gen_rowmedian(data, cols)
```

Arguments

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

Details

Parallelization is supported via purrr::in_parallel().

Value

A double vector of the rowwise median value. Missing values are ignored.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(2, 3, 2),
  z = c(4, NA, 5)
)
gen_rowmedian(a)
gen_rowmedian(a, everything())
gen_rowmedian(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rowmedian(.))
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_rowmiss *Generate rowwise count of missing values*

Description

This function returns the rowwise count of missing values in a data frame.

Usage

```
gen_rowmiss(data, cols)
```

Arguments

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

Details

Parallelization is supported via purrr::in_parallel().

Value

An integer vector of the rowwise count of missing 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_rowmiss(a)
gen_rowmiss(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rowmiss(.))
b
```

gen_rownonmiss *Generate rowwise count of nonmissing values*

Description

This function returns the rowwise count of nonmissing values in a data frame.

Usage

```
gen_rownonmiss(data, cols)
```

Arguments

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

Details

Parallelization is supported via purrr::in_parallel().

Value

An integer vector of the rowwise count of 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_rownonmiss(a)
gen_rownonmiss(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rownonmiss(.))
b
```

---

gen_rownth *Generate rowwise nth nonmissing value*

Description

This function returns the rowwise nth 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 rowwise nth 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 rowwise nth 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>
```

---

gen_rowsd *Generate rowwise standard deviation*

Description

This function returns the rowwise standard deviation in a data frame.

Usage

```
gen_rowsd(data, cols, na.rm = TRUE)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to search across. Non-numeric columns are ignored.
na.rm	Logical vector of length one passed onto stats::sd().

Details

Parallelization is supported via purrr::in_parallel().

If na.rm is TRUE (the default), missing values are removed. If na.rm is FALSE, the presence of an NA or NaN in any row will return NA for that row. As per the documentation for stats::sd(), the standard deviation of a length one vector is NA.

Value

A double vector of the rowwise standard deviation.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5),
  aa = c("a", "b", "c"),
  bb = rep(NA, 3),
  cc = rep(NaN, 3)
)
gen_rowsd(a)
gen_rowsd(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rowsd(.), r = gen_rowsd(., na.rm = FALSE))
b
```

---

gen_rowsum *Generate rowwise sum*

Description

This function returns the rowwise sum in a data frame.

Usage

```
gen_rowsum(data, cols, na.rm = TRUE)
```

Arguments

data	A data frame.
cols	<tidy-select> Columns to sum. Non-numeric columns are ignored.
na.rm	Logical vector of length one passed onto base::sum().

Details

Parallelization is supported via purrr::in_parallel().

If na.rm is TRUE (the default), missing values are removed. If na.rm is FALSE, the presence of an NA or NaN in any row will return NA or NaN for that row.

Value

A vector of the rowwise sum. The vector's type will be of common type to all rowwise numeric values.

Examples

```
library(dplyr, warn.conflicts = FALSE)
a <- tibble(
  x = c(1, NA, 2),
  y = c(NA, 3, NA),
  z = c(4, NA, 5),
  aa = c("a", "b", "c"),
  bb = rep(NA, 3),
  cc = rep(NaN, 3)
)
gen_rowsum(a)
gen_rowsum(a, all_of(letters[25:26]))
b <- a %>% mutate(q = gen_rowsum(.), r = gen_rowsum(., na.rm = FALSE))
b
```

Index