Package ‘tidier’

April 27, 2023

Title Enhanced ‘mutate’

Version 0.0.1

Description Provides 'Apache Spark' style window aggregation for R dataframes via 'mutate' in 'dplyr' flavour.

Imports dplyr (>= 1.1.0), tidyr (>= 1.3.0), checkmate (>= 2.1.0),
       rlang (>= 1.0.6), slider (>= 0.2.2), magrittr (>= 1.5), furrr
       (>= 0.3.0),

Suggests lubridate, stringr, testthat,

URL https://github.com/talegari/tidier

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Srikanth Komala Sheshachala [aut, cre]

Maintainer Srikanth Komala Sheshachala <sri.teach@gmail.com>

Repository CRAN

Date/Publication 2023-04-27 12:40:02 UTC

R topics documented:

mutate ................................................................. 2
mutate_ ............................................................... 4

Index 7
**mutate**

*Drop-in replacement for mutate*

**Description**

Provides supercharged version of `mutate` with `group_by`, `order_by` and aggregation over arbitrary window frame around a row.

**Usage**

```r
mutate(x, ..., .by, .order_by, .frame, .index)
```

**Arguments**

- `x` (data.frame)
- `...` expressions to be passed to `mutate`
- `.by` (expression, optional: Yes) columns to group by
- `.order_by` (expression, optional: Yes) columns to order by
- `.frame` (vector, optional: Yes) Vector of length 2 indicating the number of rows to consider before and after the current row. When argument `.index` is provided (typically a column of type date or datetime), before and after can be `interval` objects. See examples.
- `.index` (expression, optional: Yes) index column

**Details**

A window function returns a value for every input row of a dataframe based on a group of rows (frame) in the neighborhood of the input row. This function implements computation over groups (`partition_by in SQL`) in a predefined order (`order_by in SQL`) across a neighborhood of rows (frame) defined by a (up, down) where

- up/down are number of rows before and after the corresponding row
- up/down are interval objects (ex: `c(days(2), days(1))`)

This implementation is inspired by spark's window API.

Implementation Details:

- Iteration per row over the window is implemented using the versatile `slider`.
- Application of a window aggregation can be optionally run in parallel over multiple groups (see argument `.by`) by setting a `future` parallel backend. This is implemented using `furr` package.
- function subsumes regular usecases of `mutate`

**Value**

data.frame
mutate

See Also

mutate_

Examples

library("magrittr")
# example 1
# Using iris dataset,
# compute cumulative mean of column `Sepal.Length`
# ordered by `Petal.Width` and `Sepal.Width` columns
# grouped by `Petal.Length` column
iris %>%
mutate(sl_mean = mean(Sepal.Length),
       .order_by = c(Petal.Width, Sepal.Width),
       .by = Petal.Length,
       .frame = c(Inf, 0),
       ) %>%
dplyr::slice_min(n = 3, Petal.Width, by = Species)

# example 2
# Using a sample airquality dataset,
# compute mean temp over last seven days in the same month for every row
airquality %>%
# create date column
dplyr::mutate(date_col = as.Date(paste("1973",
stringr::str_pad(Month,
     width = 2,
     side = "left",
     pad = "0"
     ),
stringr::str_pad(Day,
     width = 2,
     side = "left",
     pad = "0"
     ),
     sep = "-"
    )
    ) %>%
# create gaps by removing some days
dplyr::slice_sample(prop = 0.8) %>%
# compute mean temperature over last seven days in the same month
mutate(avg_temp_over_last_week = mean(Temp, na.rm = TRUE),
       .order_by = Day,
       .by = Month,
       .frame = c(lubridate::days(7), # 7 days before current row
                 lubridate::days(-1) # do not include current row
                 ),
       .index = date_col
    )
**mutate_**

*Drop-in replacement for mutate*

---

**Description**

Provides supercharged version of `mutate` with `group_by`, `order_by` and aggregation over arbitrary window frame around a row. This function allows some arguments to be passed as strings instead of expressions.

**Usage**

```r
mutate_(x, ..., .by, .order_by, .frame, .index, .desc = FALSE)
```

**Arguments**

- `x` (data.frame)
- `...` expressions to be passed to `mutate`
- `by` (character vector, optional: Yes) columns to group by
- `order_by` (character vector, optional: Yes) columns to order by
- `frame` (vector, optional: Yes) Vector of length 2 indicating the number of rows to consider before and after the current row. When argument `.index` is provided (typically a column of type date or datetime), before and after can be `interval` objects. See examples.
- `index` (string, optional: Yes) name of index column
- `desc` (logical_vector, default: FALSE) bool or logical vector of same length as `.order_by`.

**Details**

A window function returns a value for every input row of a dataframe based on a group of rows (frame) in the neighborhood of the input row. This function implements computation over groups (`partition_by` in SQL) in a predefined order (`order_by` in SQL) across a neighborhood of rows (frame) defined by a (up, down) where

- **up/down** are number of rows before and after the corresponding row
- **up/down** are interval objects (ex: `c(days(2), days(1))`)

This implementation is inspired by spark’s `window API`.

Implementation Details:

- Iteration per row over the window is implemented using the versatile `slider`.
- Application of a window aggregation can be optionally run in parallel over multiple groups (see argument `.by`) by setting a `future` parallel backend. This is implemented using `furr` package.
- function subsumes regular usecases of `mutate`
**mutate_**

**Value**

data.frame

**See Also**

mutate

**Examples**

```r
library("magrittr")
# example 1
# Using iris dataset,
# compute cumulative mean of column `Sepal.Length`
# ordered by `Petal.Width` and `Sepal.Width` columns
# grouped by `Petal.Length` column
iris %>%
  mutate_(sl_mean = mean(Sepal.Length),
            .order_by = c("Petal.Width", "Sepal.Width"),
            .by = "Petal.Length",
            .frame = c(Inf, 0),
  ) %>%
dplyr::slice_min(n = 3, Petal.Width, by = Species)

# example 2
# Using a sample airquality dataset,
# compute mean temp over last seven days in the same month for every row
airquality %>%
  # create date column
dplyr::mutate(date_col = as.Date(paste("1973",
                                        stringr::str_pad(Month,
                                            width = 2,
                                            side = "left",
                                            pad = "0"
                                        ),
                                        stringr::str_pad(Day,
                                            width = 2,
                                            side = "left",
                                            pad = "0"
                                        ),
                                        sep = "-"
  )
  ) %>%
  # create gaps by removing some days
dplyr::slice_sample(prop = 0.8) %>%
  # compute mean temperature over last seven days in the same month
  mutate_(avg_temp_over_last_week = mean(Temp, na.rm = TRUE),
            .order_by = "Day",
            .by = "Month",
            .frame = c(lubridate::days(7), # 7 days before current row

mutate_

lubridate::days(-1) # do not include current row

),
.index = "date_col"
)
Index

mutate, 2, 2, 4
mutate_, 4