Package 'dtts'

August 8, 2023

Type Package

Title 'data.table' Time-Series

Version 0.1.1

Date 2023-08-08

Author Dirk Eddelbuettel and Leonardo Silvestri

Maintainer Dirk Eddelbuettel <edd@debian.org>

Description High-frequency time-series support via 'nanotime' and 'data.table'.

License GPL (>= 2)

Imports nanotime, data.table, methods, bit64, Rcpp (>= 0.11.5), RcppCCTZ (>= 0.2.0)

Suggests tinytest

LinkingTo Rcpp, RcppCCTZ, RcppDate, nanotime

BugReports https://github.com/eddelbuettel/dtts/issues

RoxygenNote 7.2.2

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-08-08 13:10:02 UTC

R topics documented:

align *Align a* data.table *onto a* nanotime *vector*

Description

align returns the subset of data.table x that aligns on the temporal vector y

Usage

```
align(x, y, start, end, ...)

## S4 method for signature 'data.table,nanotime,nanoduration,nanoduration'
align(
  x,
  y,
  start = as.nanoduration(0),
  end = as.nanoduration(0),
  sopen = FALSE,
  eopen = TRUE,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,missing,missing'
align(
  x,
  y,
  start = as.nanoduration(0),
  end = as.nanoduration(0),
  sopen = FALSE,
  eopen = TRUE,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,nanoduration,missing'
align(
  x,
  y,
  start = as.nanoduration(0),
  end = as.nanoduration(0),
  sopen = FALSE,
  eopen = TRUE,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,missing,nanoduration'
align(
  x,
```

```
  y,
  start = as.nanoduration(0),
  end = as.nanoduration(0),
  sopen = FALSE,
  eopen = TRUE,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,nanoperiod,nanoperiod'
align(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,nanoperiod,missing'
align(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  func = NULL
)

## S4 method for signature 'data.table,nanotime,missing,nanoperiod'
align(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  func = NULL
)
```

Arguments

x	the data.table time-series to align from
y	the nanotime vector to align to

start	scalar or vector of same length as y of type integer64; start is added to each element in y and it then defines the starting point of the interval under consideration for the alignment on that element of y
end	scalar or vector of same length as y of type integer64; start is added to each element in y and it then defines the ending point of the interval under consideration for the alignment on that element of y
...	further arguments passed to or from methods.
sopen	boolean scalar or vector of same lengths as y that indicates if the start of the interval is open or closed. Defaults to FALSE.
eopen	boolean scalar or vector of same lengths as y that indicates if the end of the interval is open or closed. Defaults to TRUE.
func	a function taking one argument and which provides an arbitrary aggregation of its argument; if NULL then a function which takes the closest observation is used.
tz	scalar or vector of same length as y of type character. Only used when the type of start and end is nanoperiod. It defines the time zone for the definition of the interval.

Details

For each element in y, intervals are created around this element with start and end. All the elements of x that fall within this interval are given as argument to the function func. The function func show reduce this data.frame to one unique row that will be associated with the nanotime value in y.

Value

a data.table time-series of the same length as y; this is a subset of x with the nanotime index of y

Examples

```
## Not run:
y <- nanotime((1:10)*1e9)
x <- data.table(index=nanotime((1:10)*1e9), data=1:10)
align(x, y, as.nanoduration(-1e9), as.nanoduration(1e9), colMeans)

## End(Not run)
```

---

align_idx *Get the index of the alignment of one vector onto another*

Description

align_idx returns the index of the alignment of x on y

Usage

```
align_idx(x, y, start, end, ...)

## S4 method for signature 'nanotime,nanotime,nanoduration,nanoduration'
align_idx(
  x,
  y,
  start,
  end,
  sopen = FALSE,
  eopen = TRUE,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,missing,missing'
align_idx(
  x,
  y,
  start,
  end,
  sopen = FALSE,
  eopen = TRUE,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,missing,nanoduration'
align_idx(
  x,
  y,
  start,
  end,
  sopen = FALSE,
  eopen = TRUE,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,nanoduration,missing'
align_idx(
  x,
  y,
  start,
  end,
  sopen = FALSE,
  eopen = TRUE,
  bypass_x_check = FALSE,
```

```
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,nanoperiod,nanoperiod'
align_idx(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,missing,nanoperiod'
align_idx(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)

## S4 method for signature 'nanotime,nanotime,nanoperiod,missing'
align_idx(
  x,
  y,
  start = as.nanoperiod(0),
  end = as.nanoperiod(0),
  sopen = FALSE,
  eopen = TRUE,
  tz,
  bypass_x_check = FALSE,
  bypass_y_check = FALSE
)
```

Arguments

x	the nanotime vector to align from
y	the nanotime vector to align to
start	scalar or vector of same length as y of type nanoduration or nanoperiod; start is added to each element in y and it then defines the starting point of the

	interval under consideration for the alignment on that element of y
end	scalar or vector of same length as y of type nanoduration or nanoperiod; start is added to each element in y and it then defines the ending point of the interval under consideration for the alignment on that element of y
...	further arguments passed to or from methods.
sopen	boolean scalar or vector of same lengths as y that indicates if the start of the interval is open or closed. Defaults to FALSE.
eopen	boolean scalar or vector of same lengths as y that indicates if the end of the interval is open or closed. Defaults to TRUE.
bypass_x_check	logical indicating if the sorting of x should be bypassed. This can provide a marginal speedup, but should be used carefully.
bypass_y_check	logical indicating if the sorting of y should be bypassed. This can provide a marginal speedup, but should be used carefully.
tz	scalar or vector of same length as y of type character. Only used when the type of start and end is nanoperiod. It defines the time zone for the definition of the interval.

Details

In order to perform the alignment, intervals are created around each elements in y using start and end. For each such interval, the closest element in x is chosen. If no element in x falls in the interval, then NaN is returned.

When only x and y are specified, the default is to close the intervals so that the alignment simply picks up equal points. Note that it is possible to specify meaningless intervals, for instance with a start that is beyond end. In this case, the alignment will simply return NA for each element in y. In principle, the start and end are chosen to define an interval is the past, or around the points in y, but if they are both positive, they can define intervals in the future.

Value

a vector of indices of the same length as y; this vector indexes into x and represent the closest point of x that is in the interval defined around each point in y

Examples

```
## Not run:
align_idx(nanotime(c(10:14, 17:19)), nanotime(11:20))
## [1]  2  3  4  5  NA NA  6  7  8  NA

## End(Not run)
```

`frequency,data.table-method`

Return the number of observations per interval

Description

`frequency` returns the number of observations in `data.table` `x` for each interval specified by `by`.

Usage

```
## S4 method for signature 'data.table'
frequency(
  x,
  by,
  grid_start,
  grid_end,
  tz,
  ival_start = -by,
  ival_end,
  ival_sopen = FALSE,
  ival_eopen = TRUE
)
```

Arguments

`x`	the `data.table` time-series for which to calculate the frequency
`by`	interval specified as a `nanoduration` or `nanoperiod`.
`grid_start`	scalar `nanotime` defining the start of the grid; by default the first element of `x` is taken.
`grid_end`	scalar `nanotime` defining the end of the grid; by default the last element of `x` is taken.
`tz`	scalar of type character. Only used when the type of `by` and `end` is `nanoperiod`. It defines the time zone for the definition of the interval.
`ival_start`	scalar of type `nanoduration` or `nanoperiod`; `ival_start` is added to each element of the grid and it then defines the starting point of the interval under consideration for the alignment onto that element. This defaults to `-by` and most likely does not need to be overriden.
`ival_end`	scalar of type `nanoduration` or `nanoperiod`; `ival_end` is added to each element of the grid and it then defines the ending point of the interval under consideration for the alignment onto that element. This defaults to 0 and most likely does not need to be overriden.
`ival_sopen`	boolean scalar that indicates if the start of the interval is open or closed. Defaults to FALSE.
`ival_eopen`	boolean scalar that indicates if the end of the interval is open or closed. Defaults to TRUE.

Value

a data.table time-series with the number of observations in x that fall withing the intervals defined by the grid interval defined by by.

Examples

```
## Not run:
one_second <- as.nanoduration("00:00:01")
one_minute <- 60 * one_second
x <- data.table(index=nanotime((1:100) * one_second), 1)
setkey(x, index)
frequency(x, one_minute)

## End(Not run)
```

---

grid_align *Align a* data.table *onto a* nanotime *vector grid*

Description

grid_align returns the subset of data.table x that aligns on the grid defined by by, start and end

Usage

```
grid_align(x, by, ...)

## S4 method for signature 'data.table,nanoduration'
grid_align(
  x,
  by,
  func = NULL,
  grid_start = x[[1]][1] + by,
  grid_end = tail(x[[1]], 1),
  ival_start = -by,
  ival_end = as.nanoduration(0),
  ival_sopen = FALSE,
  ival_eopen = TRUE
)

## S4 method for signature 'data.table,nanoperiod'
grid_align(
  x,
  by,
  func = NULL,
  grid_start = plus(x[[1]][1], by, tz),
  grid_end = tail(x[[1]], 1),
```

```
  ival_start = -by,
  ival_end = as.nanoperiod(0),
  ival_sopen = FALSE,
  ival_eopen = TRUE,
  tz
)
```

Arguments

x	the data.table time-series to align from
by	interval specified as a nanoduration or nanoperiod.
...	further arguments passed to or from methods.
func	a function taking one argument and which provides an arbitrary aggregation of its argument; if NULL then a function which takes the closest observation is used.
grid_start	scalar nanotime defining the start of the grid; by default the first element of x is taken.
grid_end	scalar nanotime defining the end of the grid; by default the last element of x is taken.
ival_start	scalar of type nanoduration or nanoperiod; ival_start is added to each element of the grid and it then defines the starting point of the interval under consideration for the alignment onto that element.
ival_end	scalar of type nanoduration or nanoperiod; ival_end is added to each element of the grid and it then defines the ending point of the interval under consideration for the alignment onto that element.
ival_sopen	boolean scalar that indicates if the start of the interval is open or closed. Defaults to FALSE.
ival_eopen	boolean scalar that indicates if the end of the interval is open or closed. Defaults to TRUE.
tz	scalar of type character. Only used when the type of by and end is nanoperiod. It defines the time zone for the definition of the interval.

Details

A grid defined by the parameter by, start and end is created. The function then does a standard alignment of x onto this grid (see the align function)

Value

a data.table time-series of the same length as y with the aggregations computed by func

Examples

```
## Not run:
one_second <- 1e9
x <- data.table(index=nanotime(cumsum(sin(seq(0.001, pi, 0.001)) * one_second)))
x <- x[, V2 := 1:nrow(x)]
setkey(x, index)
```

```
grid_align(x, as.nanoduration("00:01:00"), sum)

## End(Not run)
```

---

ops *Arithmetic operations on two* data.table *time-series*

Description

ops returns the y time-series on which the x time-series values are applied using the specified operator op.

Usage

```
ops(x, y, op_string)

## S4 method for signature 'data.table,data.table,character'
ops(x, y, op_string)
```

Arguments

x	the data.table time-series that determines the left operand
y	the data.table time-series that determines the right operand nanoperiod.
op_string	string defining the operation to apply; the supported values for op are "*", "/", "+", "-".

Details

The n elements of the x time-series operand define a set of n-1 intervals, and the value associated with each interval is applied to all the observations in the y time-series operand that fall in the interval. Note that the interval is closed at the beginning and open at the end. The supported values for op are "*", "/", "+", "-".

There has to be one numeric column in x and y; there has to be either a one to one correspondance between the number of numeric columns in x and y, or there must be only one numeric column in x that will be applied to all numeric columns in y. Non-numeric columns must not appear in x, whereas they will be skipped of they appear in y.

Examples

```
## Not run:
one_second_duration  <- as.nanoduration("00:00:01")
t1 <- nanotime(1:2 * one_second_duration * 3)
t2 <- nanotime(1:4 * one_second_duration)
dt1 <- data.table(index=t1, data1 = 1:length(t1))
setkey(dt1, index)
dt2 <- data.table(index=t2, data1 = 1:length(t2))
setkey(dt2, index)
ops(dt1, dt2, "+")
```

```
## End(Not run)
```

Index