Package 'DTSg'

October 14, 2019

Type Package

Title A Class for Working with Time Series Based on 'data.table' and 'R6' with Largely Optional Reference Semantics

Version 0.2.0

Description Basic time series functionalities such as listing of missing values, application of arbitrary aggregation as well as rolling window functions and automatic detection of periodicity. As it is mainly based on 'data.table', it is fast and - in combination with the 'R6' package - offers reference semantics. In addition to its native R6 interface, it provides an S3 interface inclusive an S3 wrapper method generator for those who prefer the latter.

License MIT + file LICENSE

URL https://github.com/gisler/DTSg

BugReports https://github.com/gisler/DTSg/issues

Encoding UTF-8

LazyData true

ByteCompile true

Depends R (>= 3.2.0)

Imports assertive.base, assertive.numbers, assertive.sets, assertive.types, data.table, methods, R6

Suggests covr, dygraphs, fasttime, knitr, magrittr, RColorBrewer, rmarkdown, testthat

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Gerold Hepp [aut, cre]

Maintainer Gerold Hepp <ghepp@iwag.tuwien.ac.at>

Repository CRAN

Date/Publication 2019-10-14 21:40:02 UTC

R topics documented:

`aggregate.DTSg` *Aggregate Values*

Description

Applies a temporal aggregation level function to the *.dateTime* column of a `DTSg` object and aggregates its *values* columnwise to the function's temporal aggregation level utilising a provided summary function. Additionally, it sets the object's *aggregated* field to `TRUE`. See `DTSg` for further information.

Usage

```
## S3 method for class 'DTSg'
aggregate(x, funby, fun, ..., cols = self$cols(class =
  "numeric"), n = FALSE, ignoreDST = FALSE,
  clone = getOption("DTSgClone"))
```

Arguments

`x`	A `DTSg` object (S3 method only).
`funby`	One of the temporal aggregation level functions described in `TALFs` or a user defined temporal aggregation level function. See details for further information.
`fun`	A summary function applied columnwise to all the values of the same temporal aggregation level, for instance, `mean`. Its return value must be of length one.

...	Further arguments passed on to fun.
cols	A character vector specifying the columns to aggregate.
n	A logical specifying if a column named *.n* giving the number of values per temporal aggregation level is added. See details for further information.
ignoreDST	A logical specifying if day saving time is ignored during aggregation. See details for further information.
clone	A logical specifying if the object is modified in place or if a clone (copy) is made beforehand.

Details

User defined temporal aggregation level functions have to return a POSIXct vector of the same length as the time series and accept two arguments: a POSIXct vector as its first and a list with helper data as its second. This list in turn contains the following named elements:

- *timezone:* Same as *timezone* field. See DTSg for further information.
- *ignoreDST:* Same as ignoreDST argument.
- *periodicity:* Same as *periodicity* field. See DTSg for further information.

Depending on the number of columns to aggregate, the *.n* column contains different counts:

- One column: The counts are calculated from the value column without any missing values. This means that missing values are always stripped regardless of the value of a possible na.rm argument.
- More than one column: The counts are calculated from the *.dateTime* column including all missing values.

ignoreDST tells a temporal aggregation level function if it is supposed to ignore day saving time while forming new timestamps. This can be a desired feature for time series strictly following the position of the sun (such as hydrological time series). Doing so ensures that diurnal variations are preserved under all circumstances and that all intervals are of "correct" length. This feature requires that the periodicity of the time series is not unrecognised and is supported by the following temporal aggregation level functions of the package:

- byY_____
- byYQ____
- byYm____
- byYmd___
- by_Q____
- by_m____
- by___H__

Value

Returns an aggregated DTSg object.

See Also

DTSg, TALFs, cols, POSIXct, list

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# mean yearly river flows
## R6 method
x$aggregate(funby = byY_____, fun = mean, na.rm = TRUE)

## S3 method
aggregate(x = x, funby = byY_____, fun = mean, na.rm = TRUE)
```

---

alter.DTSg *Alter Time Series*

Description

Shortens, lengthens and/or changes the periodicity of a DTSg object.

Usage

```
## S3 method for class 'DTSg'
alter(x, from = first(self$values(TRUE)[[1L]]),
  to = last(self$values(TRUE)[[1L]]), by = self$periodicity,
  rollback = TRUE, clone = getOption("DTSgClone"), ...)
```

Arguments

x	A DTSg object (S3 method only).
from	A POSIXct date with the same time zone as the time series or a character string coercible to one. Specifies the new start of the time series.
to	A POSIXct date with the same time zone as the time series or a character string coercible to one. Specifies the new end of the time series.
by	Specifies the new periodicity in one of the ways the by argument of seq.POSIXt can be specified. Must be specified for time series with unrecognised periodicity. Time steps out of sync with the new periodicity are dropped.
rollback	A logical specifying if a call to rollback is made when appropriate.
clone	A logical specifying if the object is modified in place or if a clone (copy) is made beforehand.
...	Not used (S3 method only).

Value

Returns a DTSg object.

See Also

DTSg, values, POSIXct, seq.POSIXt, rollback

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# extract the first two years
## R6 method
x$alter(from = "2007-01-01", to = "2008-12-31")

## S3 method
alter(x = x, from = "2007-01-01", to = "2008-12-31")
```

---

clone.DTSg *Clone Object*

Description

Clones (copies) a DTSg object. Merely assigning a variable representing a DTSg object to a new variable does not result in a copy of the object. Instead, both variables will reference and access the same data in the background, i.e., changing one will also affect the other. This is not an issue when calling methods with the *DTSgClone* option or clone argument set to TRUE, but has to be kept in mind when setting fields, as they are always modified in place. See DTSg for further information.

Usage

```
## S3 method for class 'DTSg'
clone(x, deep = FALSE, ...)
```

Arguments

x	A DTSg object (S3 method only).
deep	A logical specifying if a deep copy is made (for consistency with R6Class the default is FALSE, but should generally be set to TRUE).
...	Not used (S3 method only).

Value

Returns a DTSg object.

See Also

DTSg, R6Class

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# make a deep copy
## R6 method
x$clone(deep = TRUE)

## S3 method
clone(x = x, deep = TRUE)
```

---

colapply.DTSg *Apply Function Columnwise*

Description

Applies an arbitrary function to selected columns of a DTSg object.

Usage

```
## S3 method for class 'DTSg'
colapply(x, fun, ..., cols = self$cols(class =
  "numeric")[1L], clone = getOption("DTSgClone"))
```

Arguments

x	A DTSg object (S3 method only).
fun	A function. Its return value must be of length one.
...	Further arguments passed on to fun.
cols	A character vector specifying the columns to apply fun to.
clone	A logical specifying if the object is modified in place or if a clone (copy) is made beforehand.

Details

In addition to the ... argument, this method hands over a list argument with helper data called .helpers to fun. .helpers contains the following named elements:

- *.dateTime:* A POSIXct vector containing the *.dateTime* column.
- *periodicity:* Same as *periodicity* field. See DTSg for further information.
- *minLag:* A difftime object containing the minimum time difference between two subsequent timestamps.
- *maxLag:* A difftime object containing the maximum time difference between two subsequent timestamps.

Value

Returns a DTSg object.

See Also

DTSg, cols, list, POSIXct, difftime, interpolateLinear

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# linear interpolation of missing values
## R6 method
x$colapply(fun = interpolateLinear)

## S3 method
colapply(x = x, fun = interpolateLinear)
```

---

cols.DTSg *Get Column Names*

Description

Queries all column names of a DTSg object or those of a certain class only.

Usage

```
## S3 method for class 'DTSg'
cols(x, class = "all", ...)
```

Arguments

x	A DTSg object (S3 method only).
class	A character string matched to the most specific class (first element) of each column's class vector or "all" for all column names.
...	Not used (S3 method only).

Value

Returns a character vector.

See Also

DTSg, class

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# get names of numeric columns
## R6 method
x$cols(class = "numeric")

## S3 method
cols(x = x, class = "numeric")
```

DTSg *DTSg Class*

Description

The DTSg class is the working horse of the package. It is an R6Class and offers an S3 interface in addition to its native R6 interface. In the usage sections of the documentation only the S3 interface is shown, however, the examples always show both possibilities. Generally, they are very similar anyway. While the R6 interface always has the object first and the method is selected with the help of the $ operator (for instance, x$cols()), the S3 interface always has the method first and the object as its first argument (for instance, cols(x)). An exception is the new method. It is not an S3 method, but an abused S4 constructor with the character string "DTSg" as its first argument. Regarding the R6 interface, the DTSg class generator has to be used to access the new method with the help of the $ operator.

Usage

```
new(Class, values, ID = "", parameter = "", unit = "", variant = "",
 aggregated = FALSE, fast = FALSE, swallow = FALSE)
```

Arguments

Class	A character string. Must be "DTSg" in order to create a DTSg object. Otherwise a different object may or may not be created (S4 constructor only).
values	A data.frame or object inherited from class data.frame, for instance, data.table. Its first column must be of class POSIXct or coercible to it. It serves as the object's time index and is renamed to *.dateTime*.
ID	A character string specifying the ID (name) of the time series.
parameter	A character string specifying the parameter of the time series.
unit	A character string specifying the unit of the time series.
variant	A character string specifying further metadata of the time series, for instance, "min" to point out that it is a time series of lower bound measurements.
aggregated	A logical signalling how the timestamps of the series have to be interpreted: as snap-shots (FALSE) or as periods between subsequent timestamps (TRUE).

`fast` A logical signalling if all rows (`FALSE`) or only the first 1000 rows (`TRUE`) shall be used to check the object's integrity and for the automatic detection of the time series' periodicity.

`swallow` A logical signalling if the object provided through the `values` argument shall be "swallowed" by the `DTSg` object, i.e., no copy of the data shall be made. This is generally more ressource efficient, but only works if the object provided through the `values` argument is a `data.table`. Be warned, however, that if the creation of the `DTSg` object fails for some reason, the first column of the provided `data.table` might have been coerced to `POSIXct` and keyed (see `setkey` for further information). Furthermore, all references to the "swallowed" `data.table` in the global environment are removed upon successfull creation of a `DTSg` object.

Format

An object of class `R6ClassGenerator` of length 24.

Value

Returns a `DTSg` object.

Methods

A `DTSg` object has the following methods:

- `aggregate`: See `aggregate` for further information.
- `alter`: See `alter` for further information.
- `colapply`: See `colapply` for further information.
- `cols`: See `cols` for further information.
- `merge`: See `merge` for further information.
- `nas`: See `nas` for further information.
- `plot`: See `plot` for further information.
- `refresh`: See `refresh` for further information.
- `rollapply`: See `rollapply` for further information.
- `summary`: See `summary` for further information.
- `values`: See `values` for further information.

Fields

A `DTSg` object has the following fields or properties as they are often called. They are implemented through so called active bindings, which means that they can be accessed and actively set with the help of the `$` operator (for instance, `x$ID` gets the value of the *ID* field and `x$ID <-"River Flow"` sets its value). Please note that fields are always modified in place, i.e., no clone (copy) of the object is made beforehand. See `clone` for further information. Some of the fields are read-only though:

- *aggregated:* Same as `aggregated` argument.

- *fast:* Same as `fast` argument.
- *ID:* Same as `ID` argument. It is used as the title of plots.
- *parameter:* Same as `parameter` argument. It is used as the label of the primary axis of plots.
- *periodicity:* A `difftime` object for a regular and a character string for an irregular `DTSg` object describing its periodicity or containing `"unrecognised"` in case it could not be detected (read-only).
- *regular:* A logical signalling if all lags in seconds between subsequent timestamps are the same (`TRUE`) or if some are different (`FALSE`). A, for instance, monthly time series is considered irregular in this sense (read-only).
- *timestamps:* An integer showing the total number of timestamps of the time series (read-only).
- *timezone:* A character string containing the time zone of the time series (read-only).
- *unit:* Same as `unit` argument. It is added to the label of the primary axis of plots if the *parameter* field is set.
- *variant:* Same as `variant` argument. It is added to the label of the primary axis of plots if the *parameter* field is set.

The *parameter*, *unit* and *variant* fields are especially useful for time series with one variable (value column) only.

Options

The behaviour of `DTSg` objects can be customised with the help of the following option. See `options` for further information:

- *DTSgClone:* A logical specifying if `DTSg` objects are, by default, modified in place (`FALSE`) or if a clone (copy) is made beforehand (`TRUE`).

Note

Due to the `POSIXct` nature of the *.dateTime* column, the same sub-second accuracy, issues and limitations apply to `DTSg` objects. In order to prevent at least some of the possible precision issues, the lags in seconds between subsequent timestamps are rounded to microseconds during integrity checks. This corresponds to the maximum value allowed in `options("digits.secs")`. As a consequence, time series with a sub-second accuracy higher than a microsecond will never work and with a sub-second accuracy lower than a microsecond might work.

Some of the methods which take a function as an argument (`colapply` and `rollapply`) hand over to it an additional `list` argument called `.helpers` containing useful data for the development of user defined functions (see the respective help pages for further information). This can of course be a problem for functions like `sum` which do not expect such a thing. A solution is to wrap it in an anonymous function with a `...` parameter like this: `function(x,...) sum(x)`.

See Also

`R6Class`, `data.frame`, `data.table`, `POSIXct`, `setkey`, `difftime`, `clone`, `options`, `list`

Examples

```
# new DTSg object
## R6 constructor
DTSg$new(values = flow, ID = "River Flow")

## S4 constructor
new(Class = "DTSg", values = flow, ID = "River Flow")
```

`flow` *Daily River Flows*

Description

A dataset containing a fictional time series of daily river flows.

Usage

```
flow
```

Format

A `data.table` with 2169 rows and 2 columns:

date A `POSIXct` vector ranging from the start of the year 2007 to the end of the year 2012.

flow A numeric vector with daily river flows.

`interpolateLinear` *Linear Interpolation*

Description

Linearly interpolates missing values of a numeric vector. For use with the `colapply` method of a `DTSg` object. Other uses are possible, but not recommended. It also serves as an example for writing user defined functions utilising one of the `list`s with helper data as handed over by various methods of `DTSg` objects. See `DTSg` for further information.

Usage

```
interpolateLinear(.col, roll = Inf, rollends = TRUE, .helpers)
```

Arguments

.col	A numeric vector.
roll	A positive numeric specifying the maximum size of gaps whose missing values shall be filled. For time series with unrecognised periodicity it is interpreted as seconds and for time series with recognised periodicity it is multiplied with the maximum time difference between two subsequent time steps in seconds. So for regular time series it is the number of time steps and for irregular it is an approximation of it.
rollends	A logical specifying if missing values at the start and end of the time series shall be filled as well. See data.table for further information.
.helpers	A list with helper data as handed over by colapply. See colapply for further information.

Value

Returns a numeric vector.

See Also

DTSg, colapply, data.table

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# linear interpolation of missing values
## R6 method
x$colapply(fun = interpolateLinear)

## S3 method
colapply(x = x, fun = interpolateLinear)
```

---

merge.DTSg *Merge Two DTSg Objects*

Description

Joins two DTSg objects based on their *.dateTime* column. Their time zones and *aggregated* field must be the same.

Usage

```
## S3 method for class 'DTSg'
merge(x, y, ..., clone = getOption("DTSgClone"))
```

Arguments

x	A DTSg object (S3 method only).
y	A DTSg object or an object coercible to one. See new for further information.
...	Further arguments passed on to merge. As the by, by.x and by.y arguments can endanger the integrity of the object, they are not allowed here.
clone	A logical specifying if the object is modified in place or if a clone (copy) is made beforehand.

Value

Returns a DTSg object.

See Also

DTSg, new, merge

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# merge with data.table
## R6 method
x$merge(y = flow, suffixes = c("_1", "_2"))

## S3 method
merge(x = x, y = flow, suffixes = c("_1", "_2"))
```

---

nas.DTSg *List Missing Values*

Description

Lists the missing values of selected columns of a DTSg object with recognised periodicity.

Usage

```
## S3 method for class 'DTSg'
nas(x, cols = self$cols(), ...)
```

Arguments

x	A DTSg object (S3 method only).
cols	A character vector specifying the columns whose missing values shall be listed.
...	Not used (S3 method only).

Value

Returns a data.table with five columns:

- *.col:* the column name.
- *.group:* the ID of the missing values group within each column.
- *.from:* the start date of the missing values group.
- *.to:* the end date of the missing values group.
- *.n:* the number of missing values in the group.

See Also

DTSg, cols, data.table

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# list missing values
## R6 method
x$nas()

## S3 method
nas(x = x)
```

---

plot.DTSg *Plot Time Series*

Description

Displays an interactive plot of a DTSg object. This method requires **dygraphs** and **RColorBrewer** to be installed. Its main purpose is not to make pretty plots, but rather to offer a possibility to interactively explore time series. The title of the plot and the label of its primary axis are automatically generated out of the object's metadata (fields). See DTSg for further information.

Usage

```
## S3 method for class 'DTSg'
plot(x, from = first(self$values(TRUE)[[1L]]),
  to = last(self$values(TRUE)[[1L]]), cols = self$cols(class =
  "numeric"), secAxisCols = NULL, secAxisLabel = "", ...)
```

Arguments

x	A DTSg object (S3 method only).
from	A POSIXct date with the same time zone as the time series or a character string coercible to one. The time series is plotted from this date on.
to	A POSIXct date with the same time zone as the time series or a character string coercible to one. The time series is plotted up to this date.
cols	A character vector specifying the columns whose values shall be plotted.
secAxisCols	A character vector specifying the columns whose values shall be plotted on a secondary axis. Must be a subset of cols.
secAxisLabel	A character string specifying the label of the secondary axis.
...	Not used (S3 method only).

Value

Returns a DTSg object.

See Also

DTSg, dygraph, POSIXct, cols

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# plot time series
if (requireNamespace("dygraphs", quietly = TRUE) &&
    requireNamespace("RColorBrewer", quietly = TRUE)) {
  ## R6 method
  x$plot()

  ## S3 method
  plot(x = x)
}
```

---

refresh.DTSg *Object Integrity*

Description

Checks the integrity of a DTSg object and tries to automatically (re-)detect its periodicity. Normally, there is no reason for a user to call this method. The only exception is stated in values.

Usage

```
## S3 method for class 'DTSg'
refresh(x, ...)
```

Arguments

x	A DTSg object (S3 method only).
...	Not used (S3 method only).

Value

Returns a DTSg object.

See Also

DTSg, values

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# check object integrity
## R6 method
x$refresh()

## S3 method
refresh(x = x)
```

---

rollapply.DTSg *Rolling Window Function*

Description

Applies an arbitrary function to a rolling window of selected columns of a DTSg object with recognised periodicity.

Usage

```
## S3 method for class 'DTSg'
rollapply(x, fun, ..., cols = self$cols(class =
  "numeric")[1L], before = 1L, after = before,
  weights = c("inverseDistance"), parameters = list(power = 1),
  clone = getOption("DTSgClone"))
```

Arguments

x	A DTSg object (S3 method only).
fun	A function. Its return value must be of length one.
...	Further arguments passed on to fun.
cols	A character vector specifying the columns whose rolling window fun shall be applied to.
before	A numeric specifying the size of the window in time steps before the "center" of the rolling window.
after	A numeric specifying the size of the window in time steps after the "center" of the rolling window.
weights	A character string specifying a method to calculate weights for fun, for instance, weighted.mean. See details for further information.
parameters	A list specifying parameters for weights. See details for further information.
clone	A logical specifying if the object is modified in place or if a clone (copy) is made beforehand.

Details

In addition to the ... argument, this method hands over the weights as a numeric vector (w argument) and a list argument with helper data called .helpers to fun. .helpers contains the following named elements:

- *before:* Same as before argument.
- *after:* Same as after argument.
- *windowSize:* Size of the rolling window (before + 1L + after).
- *centerIndex:* Index of the "center" of the rolling window (before + 1L).

Currently, only one method to calculate weights is supported: "inverseDistance". The distance d of the "center" is one and each time step away from the "center" adds one to it. So, for example, the distance of a timestamp three steps away from the "center" is four. Additionally, the calculation of the weights accepts a power p parameter as a named element of a list provided through the parameters argument: $\frac{1}{d^p}$.

Value

Returns a DTSg object.

See Also

DTSg, cols, list

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# calculate moving average
## R6 method
x$rollapply(fun = mean, na.rm = TRUE, before = 2, after = 2)

## S3 method
rollapply(x = x, fun = mean, na.rm = TRUE, before = 2, after = 2)
```

`rollback` *Rollback Month*

Description

Generating regular sequences of times with the help of `seq.POSIXt` can have undesirable effects. This function "first advances the month without changing the day: if this results in an invalid day of the month, it is counted forward into the next month". Monthly or yearly sequences starting at the end of a month with 30 or 31 days (or 29 in case of a leap year) therefore do not always fall on the end of shorter months. `rollback` reverts this process by counting the days backwards again.

Usage

```
rollback(.dateTime, periodicity)
```

Arguments

`.dateTime`	A `POSIXct` vector.
`periodicity`	A character string specifying a multiple of month(s) or year(s). See `seq.POSIXt` for further information.

Value

Returns a `POSIXct` vector.

See Also

`seq.POSIXt`, `POSIXct`

Examples

```
# rollback monthly time series
by <- "1 month"
rollback(
  .dateTime = seq(
    from = as.POSIXct("2000-01-31", tz = "UTC"),
```

```
    to = as.POSIXct("2000-12-31", tz = "UTC"),
    by = by
  ),
  periodicity = by
)
```

`S3WrapperGenerator` *S3 Wrapper Method Generator*

Description

Generates S3 wrapper methods for public methods of `R6ClassGenerator`s, but can also be used to generate "plain" function wrappers.

Usage

```
S3WrapperGenerator(R6Method, self = "x", dots = TRUE)
```

Arguments

`R6Method`	An `expression` with a public method of an `R6ClassGenerator`.
`self`	A character string specifying the name of the parameter which will take the R6 object.
`dots`	A logical specifying if a `...` parameter shall be added as last parameter in case none already exists. This might be required for S3 generic/method consistency.

Value

Returns a function with the required parameters which captures its own call, reshapes it to the corresponding R6 method call and evaluates it.

See Also

`S3Methods`, `R6Class`, `expression`

Examples

```
# generate S3 wrapper method for alter of DTSg
alter.DTSg <- S3WrapperGenerator(
  R6Method = expression(DTSg$public_methods$alter)
)
```

summary.DTSg *Time Series Summary*

Description

Calculates summary statistics of selected columns of a DTSg object.

Usage

```
## S3 method for class 'DTSg'
summary(object, cols = self$cols(), ...)
```

Arguments

object	A DTSg object (S3 method only).
cols	A character vector specifying the columns whose values shall be summarised.
...	Further arguments passed on to summary.data.frame.

Value

Returns a table.

See Also

DTSg, cols, summary.data.frame, table

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# calculate summary statistics
## R6 method
x$summary()

## S3 method
summary(object = x)
```

TALFs *Temporal Aggregation Level Functions*

Description

Simply specify one of these functions as `funby` argument of `DTSg` objects' `aggregate` method. The method does the rest of the work. See details for further information. Other uses are possible, but not recommended.

Usage

```
byFasttimeY_____(.dateTime, .helpers)

byFasttimeYQ____(.dateTime, .helpers)

byFasttimeYm____(.dateTime, .helpers)

byFasttimeYmd___(.dateTime, .helpers)

byFasttimeYmdH__(.dateTime, .helpers)

byFasttimeYmdHM_(.dateTime, .helpers)

byFasttimeYmdHMS(.dateTime, .helpers)

byFasttime______(.dateTime, .helpers)

byFasttime_Q____(.dateTime, .helpers)

byFasttime_m____(.dateTime, .helpers)

byFasttime___H__(.dateTime, .helpers)

byFasttime____M_(.dateTime, .helpers)

byFasttime_____S(.dateTime, .helpers)

byY_____(.dateTime, .helpers)

byYQ____(.dateTime, .helpers)

byYm____(.dateTime, .helpers)

byYmd___(.dateTime, .helpers)

byYmdH__(.dateTime, .helpers)
```

```
byYmdHM_(.dateTime, .helpers)

byYmdHMS(.dateTime, .helpers)

by______(.dateTime, .helpers)

by_Q____(.dateTime, .helpers)

by_m____(.dateTime, .helpers)

by___H__(.dateTime, .helpers)

by____M_(.dateTime, .helpers)

by_____S(.dateTime, .helpers)
```

Arguments

.dateTime	A POSIXct vector.
.helpers	A list with helper data as handed over by DTSg objects' aggregate method.

Details

There are two families of temporal aggregation level functions. The one family truncates timestamps (truncating family), the other extracts a certain part of them (extracting family). Each family comes in two flavours: one using fastPOSIXct of **fasttime**, the other solely relying on base R. The **fasttime** versions work with UTC time series only and are limited to dates between the years 1970 and 2199, but generally are faster for the extracting family of functions.

The truncating family sets timestamps to the lowest possible time of the corresponding temporal aggregation level:

- *Y_____ truncates to year, e.g., *2000-11-11 11:11:11.1* becomes *2000-01-01 00:00:00.0*
- *YQ____ truncates to quarter, e.g., *2000-11-11 11:11:11.1* becomes *2000-10-01 00:00:00.0*
- *Ym____ truncates to month, e.g., *2000-11-11 11:11:11.1* becomes *2000-11-01 00:00:00.0*
- *Ymd___ truncates to day, e.g., *2000-11-11 11:11:11.1* becomes *2000-11-11 00:00:00.0*
- *YmdH__ truncates to hour, e.g., *2000-11-11 11:11:11.1* becomes *2000-11-11 11:00:00.0*
- *YmdHM_ truncates to minute, e.g., *2000-11-11 11:11:11.1* becomes *2000-11-11 11:11:00.0*
- *YmdHMS truncates to second, e.g., *2000-11-11 11:11:11.1* becomes *2000-11-11 11:11:11.0*

By convention, the extracting family sets the year to 2199 and extracts a certain part of timestamps:

- *______ extracts nothing, i.e., all timestamps become *2199-01-01 00:00:00.0*
- *_Q____ extracts the quarters, e.g., *2000-11-11 11:11:11.1* becomes *2199-10-01 00:00:00.0*
- *_m____ extracts the months, e.g., *2000-11-11 11:11:11.1* becomes *2199-11-01 00:00:00.0*
- *___H__ extracts the hours, e.g., *2000-11-11 11:11:11.1* becomes *2199-01-01 11:00:00.0*
- *____M_ extracts the minutes, e.g., *2000-11-11 11:11:11.1* becomes *2199-01-01 00:11:00.0*
- *_____S extracts the seconds, e.g., *2000-11-11 11:11:11.1* becomes *2199-01-01 00:00:11.0*

Value

All functions return a POSIXct vector with timestamps corresponding to the function's temporal aggregation level.

See Also

DTSg, aggregate, fastPOSIXct

---

values.DTSg *Get Values*

Description

Queries the *values* of a DTSg object.

Usage

```
## S3 method for class 'DTSg'
values(x, reference = FALSE, drop = FALSE,
  class = c("data.table", "data.frame"), ...)
```

Arguments

x	A DTSg object (S3 method only).
reference	A logical specifying if a copy of the *values* or a reference to the *values* is returned. See details for further information.
drop	A logical specifying if the object and all references to it shall be removed from the global environment after successfully querying its values. This feature allows for a ressource efficient destruction of a DTSg object while preserving its *values*.
class	A character string specifying the class of the returned *values*. "data.frame" only works if either a copy of the *values* is returned or the object is dropped.
...	Not used (S3 method only).

Details

A reference to the *values* of a DTSg object can be used to modify them in place. This includes the *.dateTime* column, which serves as the object's time index. Modifying this column can therefore endanger the object's integrity. In case needs to do so ever arise, refresh should be called immediately afterwards in order to check the object's integrity.

Value

Returns a data.table, a reference to a data.table or a data.frame.

Note

The original name of the *.dateTime* column is restored when not returned as a reference or when dropped.

See Also

DTSg, refresh, data.table, data.frame

Examples

```
# new DTSg object
x <- DTSg$new(values = flow)

# get values
## R6 method
x$values()

## S3 method
values(x = x)
```

Index