Package ‘DTSg’

September 27, 2023

**Type** Package

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

**Version** 1.1.3

**Description** Basic time series functionalities such as listing of missing values, application of arbitrary aggregation as well as rolling (asymmetric) 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 for those who prefer the latter. Finally yet importantly, its functional approach allows for incorporating functionalities from many other packages.

**License** MIT + file LICENSE

**URL** [https://gisler.github.io/DTSg/](https://gisler.github.io/DTSg/)

**BugReports** [https://github.com/gisler/DTSg/issues](https://github.com/gisler/DTSg/issues)

**Language** en-GB

**Encoding** UTF-8

**LazyData** true

**ByteCompile** true

**Depends** R (>= 3.2.0)

**Imports** checkmate, data.table, methods, R6

**Suggests** dygraphs, fasttime, knitr, magrittr, RColorBrewer, RcppCCTZ, rmarkdown, runner (>= 0.3.5), tinytest, units

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Gerold Hepp [aut, cre]

**Maintainer** Gerold Hepp <gisler@hepp.cc>

**Repository** CRAN

**Date/Publication** 2023-09-27 21:50:19 UTC
R topics documented:

aggregate.DTSg ............................................. 2
alter.DTSg ..................................................... 6
clone.DTSg ..................................................... 8
colapply.DTSg ................................................. 9
cols.DTSg ....................................................... 14
DTSg .......................................................... 15
flow ............................................................. 19
getCol.DTSg ..................................................... 19
interpolateLinear ............................................. 20
merge.DTSg ..................................................... 21
nas.DTSg ......................................................... 22
plot.DTSg ......................................................... 23
print.DTSg ....................................................... 24
refresh.DTSg ................................................... 25
rollapply.DTSg ................................................ 26
rollback .......................................................... 28
rowaggregate.DTSg .......................................... 29
colapply.DTSg ............................................... 31
S3WrapperGenerator .......................................... 32
setColNames.DTSg .......................................... 33
setCols.DTSg .................................................. 34
subset.DTSg ..................................................... 36
summary.DTSg ................................................ 39
TALFs .......................................................... 40
values.DTSg ..................................................... 42

Index 44

aggregate.DTSg  Aggregate values

Description

Applies a temporal aggregation level function to the .dateTime column of a DTSg object and aggregates its values column-wise to the function's temporal aggregation level utilising one or more provided summary functions. Additionally, it sets the object's aggregated field to TRUE.

Usage

```r
## S3 method for class 'DTSg'
aggregate(
  x,
  funby,
  fun,
  ...,  
  cols = self$scols(class = "numeric"),
```
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 corresponding section for further information.
- **fun**: A summary function, (named) `list` of summary functions or (named) character vector specifying summary functions applied column-wise to all the values of the same temporal aggregation level. The return value(s) must be of length one. See corresponding section for further information.
- **...**: Further arguments passed on to `fun`.
- **cols**: A character vector specifying the columns to aggregate. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".
- **n**: A logical specifying if a column named `.n` giving the number of values per temporal aggregation level shall be added. See corresponding section for further information.
- **ignoreDST**: A logical specifying if day saving time shall be ignored by `funby`. See corresponding section for further information.
- **multiplier**: A positive integerish value “multiplying” the temporal aggregation level of certain TALFs. See corresponding section for further information.
- **funbyHelpers**: An optional `list` with helper data passed on to `funby`. See corresponding section for further information.
- **funbyApproach**: A character string specifying the flavour of the applied temporal aggregation level function. Either “base”, which utilises `as.POSIXct`, or “fasttime”, which utilises `fasttime::fastPOSIXct`, or “RcppCCTZ”, which utilises `RcppCCTZ::parseDatetime` as the main function for transforming timestamps.
- **clone**: A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns an aggregated `DTSg` object.

User defined TALFs, TALFs helper data and multiplier

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. The default elements of this `list` are as follows:
• timezone: Same as the timezone field.
• ignoreDST: Same as the ignoreDST argument.
• periodicity: Same as the periodicity field.
• na.status: Same as the na.status field.
• multiplier: Same as the multiplier argument.
• funbyApproach: Same as the funbyApproach argument.

Any additional element specified in the funbyHelpers argument is appended to the end of the helper data list. In case funbyHelpers contains an ignoreDST, multiplier or funbyApproach element, it takes precedence over the respective method argument. timezone, periodicity and na.status elements are rejected, as they are always taken directly from the object.

The temporal aggregation level of certain TALFs can be adjusted with the help of the multiplier argument. A multiplier of 10, for example, makes byY____ aggregate to decades instead of years. Another example is a multiplier of 6 provided to by_m_____. The function then aggregates all months of all first and all months of all second half years instead of all months of all years separately. This feature is supported by the following TALFs of the package:

• byY____
• byYm____
• byYmdH__ (UTC and equivalent as well as all Etc/GMT time zones only)
• byYmdHM_
• byYmdHMS
• by_m____
• by____H__ (UTC and equivalent as well as all Etc/GMT time zones only)
• by____M_
• by____S

Summary functions

Some examples for fun are as follows:

• mean
• list (min = min, max = max)
• c(sd = "sd", var = "var")

A list or character vector must have names in case more than one summary function is provided. The method can benefit from data.table’s GForce optimisation in case a character vector specifying summary functions is provided.

Number of values per temporal aggregation level

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

• One column: The counts are calculated from the columns’ values disregarding 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.
Ignore day saving time

ignoreDST tells a temporal aggregation level function if it is supposed to ignore day saving time while transforming the 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 by all means and all intervals are of the “correct” length, however, a possible limitation might be that the day saving time shift is invariably assumed to be one hour long. This feature requires that the periodicity of the time series was recognised and is supported by the following TALFs of the package:

- byY_____
- byYQ_____ 
- byYm_____ 
- byYmd___ 
- by_Q_____ 
- by_m___ 
- by___H__

See Also
cols,getOption

Examples

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

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

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

# variance and standard deviation of river flows per quarter
## R6 method
x$aggregate(
  funby = byYQ____,
  fun = c(var = "var", sd = "sd"),
  na.rm = TRUE
)$print()}
# S3 method
print(aggregate(
  x = x,
  funby = byYQ___,
  fun = c(var = "var", sd = "sd"),
  na.rm = TRUE
))

# mean of river flows of all first and all second half years
## R6 method
x$aggregate(
  funby = by_m____,
  fun = "mean",
  na.rm = TRUE,
  multiplier = 6
)$print()

## S3 method
print(aggregate(
  x = x,
  funby = by_m____,
  fun = "mean",
  na.rm = TRUE,
  multiplier = 6
))

---

### alter.DTSg

**Alter time series**

**Description**

Shortens, lengthens, filters for a consecutive range, changes the periodicity and/or the status of missing values of a DTSg object.

**Usage**

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

\[x\] A \texttt{DTSg} object (S3 method only).

\[\text{from}\] A \texttt{POSIXct} timestamp in the same time zone as the time series or a character string coercible to one. Specifies the new start of the time series.

\[\text{to}\] A \texttt{POSIXct} timestamp in the same time zone as the time series or a character string coercible to one. Specifies the new end of the time series.

\[\text{by}\] Specifies the new periodicity in one of the ways the \texttt{by} argument of \texttt{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.

\[\text{rollback}\] A logical specifying if a call to \texttt{rollback} shall be made when appropriate.

\[\text{clone}\] A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

\[\text{na.status}\] A character string. Either "explicit", which makes missing timestamps explicit according to the recognised periodicity, or "implicit", which removes timestamps with missing values on all value columns. Please note that \texttt{DTSg} objects work best with explicitly missing values.

... Not used (S3 method only).

Value

Returns a \texttt{DTSg} object.

See Also

\texttt{getOption, subset, nas}

Examples

\[
\begin{array}{l}
\text{# new DTSg object} \\
x <- \text{DTSg}\$\text{new(values = flow)} \\

\text{# filter for the first two years} \\
\text{## R6 method} \\
x\$\text{alter(} \\
\text{\hspace{1cm} from = "2007-01-01",} \\
\text{\hspace{1cm} to = "2008-12-31"} \\
\text{\hspace{1cm})}\$\text{print()}

\text{## S3 method} \\
\text{print(alter(} \\
\text{\hspace{1cm} x = x,} \\
\text{\hspace{1cm} from = "2007-01-01",} \\
\text{\hspace{1cm} to = "2008-12-31"} \\
\text{\hspace{1cm})})}

\text{# change periodicity to one month} \\
\text{## R6 method} \\
x\$\text{alter(by = "1 month")}\$\text{print()}
\end{array}
\]
clone.DTSg

## S3 method
print(alter(x = x, by = "1 month"))

### 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 under the hood, 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

```r
## 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 shall be made (for consistency with the R6::R6Class the default is FALSE, but should generally be set to TRUE).
- **...**: Not used (S3 method only).

### Value
Returns a cloned DTSg object.

### See Also
options

### Examples

```r
# 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)
```
Description

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

Usage

```r
## S3 method for class 'DTSg'
colapply(
  x, 
  fun, 
  ..., 
  cols = self$cols(class = "numeric")[1L], 
  resultCols = NULL, 
  suffix = NULL, 
  helpers = TRUE, 
  funby = NULL, 
  ignoreDST = FALSE, 
  multiplier = 1L, 
  funbyHelpers = NULL, 
  funbyApproach = self$funbyApproach, 
  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. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".
- **resultCols**: An optional character vector of the same length as `cols` specifying the column names for the return values of `fun`. Another possibility is a character string containing comma separated column names, for example, "x,y,z". Non-existing columns are added and existing columns are overwritten. Columns are matched element-wise between `cols` and `resultCols`.
- **suffix**: An optional character string. The return values of `fun` are added as new columns with names consisting of the columns specified in `cols` and this suffix. Existing columns are never overwritten. Only used when `resultCols` is not specified.
- **helpers**: A logical specifying if helper data shall be handed over to `fun`. See corresponding section for further information.
funby One of the temporal aggregation level functions described in TALFs or a user defined temporal aggregation level function. Can be used to apply functions like cumsum to a certain temporal level. See corresponding section and examples for further information.

ignoreDST A logical specifying if day saving time shall be ignored by funby. See corresponding section for further information.

multiplier A positive integerish value “multiplying” the temporal aggregation level of certain TALFs. See corresponding section for further information.

funbyHelpers An optional list with helper data passed on to funby. See corresponding section for further information.

funbyApproach A character string specifying the flavour of the applied temporal aggregation level function. Either "base", which utilises as.POSIXct, or "fasttime", which utilises fasttime::fastPOSIXct, or "RcppCCTZ", which utilises RcppCCTZ::parseDatetime as the main function for transforming timestamps.

clon A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DTSg object.

Helper data

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

• .dateTime: A POSIXct vector containing the .dateTime column.
• periodicity: Same as the periodicity field.
• minLag: A diffftime object containing the minimum time difference between two subsequent timestamps.
• maxLag: A diffftime object containing the maximum time difference between two subsequent timestamps.

User defined TALFs, TALFs helper data and multiplier

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. The default elements of this list are as follows:

• timezone: Same as the timezone field.
• ignoreDST: Same as the ignoreDST argument.
• periodicity: Same as the periodicity field.
• na.status: Same as the na.status field.
• multiplier: Same as the multiplier argument.
• funbyApproach: Same as the funbyApproach argument.
Any additional element specified in the `funbyHelpers` argument is appended to the end of the helper data list. In case `funbyHelpers` contains an `ignoreDST`, `multiplier` or `funbyApproach` element, it takes precedence over the respective method argument. `timezone`, `periodicity` and `na.status` elements are rejected, as they are always taken directly from the object.

The temporal aggregation level of certain TALFs can be adjusted with the help of the `multiplier` argument. A multiplier of 10, for example, makes `byY_____` aggregate to decades instead of years. Another example is a multiplier of 6 provided to `by_m____`. The function then aggregates all months of all first and all months of all second half years instead of all months of all years separately. This feature is supported by the following TALFs of the package:

- `byY_____`
- `byYm____`
- `byYmdH___` (UTC and equivalent as well as all Etc/GMT time zones only)
- `byYmdHM_`
- `byYmdHMS`
- `by_m____`
- `by_____H__` (UTC and equivalent as well as all Etc/GMT time zones only)
- `by_____M_`
- `by_____S`

### Ignore day saving time

`ignoreDST` tells a temporal aggregation level function if it is supposed to ignore day saving time while transforming the 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 by all means and all intervals are of the “correct” length, however, a possible limitation might be that the day saving time shift is invariably assumed to be one hour long. This feature requires that the periodicity of the time series was recognised and is supported by the following TALFs of the package:

- `byY_____`
- `byYQ_____`
- `byYm____`
- `byYm_____`
- `byYmd___`
- `by_____Q__`
- `by_m____`
- `by_____H__`

### See Also

`cols`, `getOption`
Examples

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

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

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

# daily cumulative sums per month
## R6 method
x$colapply(
  fun = cumsum,
  helpers = FALSE,
  funby = byYm___
)$print()

## S3 method
print(colapply(
  x = x,
  fun = cumsum,
  helpers = FALSE,
  funby = byYm___
))

# calculate moving averages with the help of 'runner' (all four given
# approaches provide the same result with explicitly missing timestamps)
if (requireNamespace("runner", quietly = TRUE) &&
  packageVersion("runner") >= package_version("0.3.5")) {
  wrapper <- function(..., .helpers) {
    runner::runner(..., idx = .helpers["\dateTime"])
  }

  ## R6 method
  x$colapply(
    fun = runner::runner,
    f = mean,
    k = 5,
    lag = -2
  )$print()
  x$colapply(
    fun = wrapper,
    f = mean,
    k = "5 days",
    lag = "-2 days"
  )$print()
}

x$colapply(
  fun = runner::runner,
  f = mean,
  k = "5 days",

lag = "-2 days",
idx = x$getCol(col = ".dateTime")
)print()
x$colapply(
  fun = runner::runner,
  f = mean,
  k = "5 days",
  lag = "-2 days",
  idx = x[['dateTime']]
)print()

## S3 method
print(colapply(
  x = x,
  fun = runner::runner,
  f = mean,
  k = 5,
  lag = -2
))
print(colapply(
  x = x,
  fun = wrapper,
  f = mean,
  k = "5 days",
  lag = "-2 days"
))
print(colapply(
  x = x,
  fun = runner::runner,
  f = mean,
  k = "5 days",
  lag = "-2 days",
  idx = getCol(x = x, col = ".dateTime")
))
print(colapply(
  x = x,
  fun = runner::runner,
  f = mean,
  k = "5 days",
  lag = "-2 days",
  idx = x[['dateTime']]
))

# calculate rolling correlations somewhat inefficiently with the help of
# 'runner'
if (requireNamespace("runner", quietly = TRUE) &&
    packageVersion("runner") >= package_version("0.3.8")) {
  wrapper <- function(x, y, f, k, lag, ...) {
    runner::runner(
      cbind(x, y),
      f = function(x) f(x[, 1], x[, 2]),
      k = k,
      lag = lag,
      idx = idx),
      f = f,
      k = k
    )
  }
cols.DTSg

Get column names

Description

Returns all column names of a DTSg object, those of certain classes, modes, typeofs and/or those matching a certain pattern only.

Usage

## S3 method for class 'DTSg'
cols(x, class = NULL, pattern = NULL, mode = NULL, typeof = NULL, ...)

Arguments

x A DTSg object (S3 method only).

class An optional character vector matched to the most specific class (first element) of each column's class vector. The "special class" .numerary matches the integer and numeric classes.

pattern An optional character string passed on to the pattern argument of grep.

mode An optional character vector matched to each column's mode.

typeof An optional character vector matched to each column's typeof.

... Further arguments passed on to grep. The value argument is rejected.
Value

Returns a character vector.

Examples

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

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

## 'names()' is a "hidden" R6 alias for 'cols()'
x$names(class = "numeric")

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

### Description

The DTSg class is the working horse of the package. It is an `R6::R6Class` and offers an S3 interface in addition to its native R6 interface. In the usage sections of the documentation, unfortunately, only the usage of the S3 methods are displayed, however, the examples always show both ways of calling the respective method. Generally, they are very similar anyway. While the R6 interface always has the object first and the method is then selected with the help of the `$` operator, for instance, `x$cols()`, the S3 interface always has the method first and then 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

```r
new(Class, values, ID = "", parameter = "", unit = "", variant = "",
aggregated = FALSE, fast = getOption("DTSgFast"), swallow = FALSE,
na.status = getOption("DTSgNA.status"), funbyApproach =
getOption("DTSgFunbyApproach"))
```

### 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, e.g. data.table::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 data object.

parameter A character string specifying the parameter name of the time series data.

unit A character string specifying the unit of the time series data.

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 specifying how the timestamps of the series have to be interpreted: as snap-shots (FALSE) or as periods between subsequent timestamps (TRUE).

groups A character string specifying how the aggregations shall be performed. Possible options are "groups" to aggregate on the groups of the data, or "id" to aggregate on the ID of the data.

na.status A character string. Either "explicit", which makes missing timestamps explicit according to the recognised periodicity, or "implicit", which removes timestamps with missing values on all value columns, or "undecided" for no such action. Please note that DTSg objects work best with explicitly missing values.

funbyApproach A character string specifying the default flavour of TALFs used with the created DTSg object. Either "base", which utilises as.POSIXct, or "fasttime", which utilises fasttime::fastPOSIXct, or "RcppCCTZ", which utilises RcppCCTZ::parseDatetime as the main function for transforming timestamps. Custom approaches for user defined temporal aggregation level functions are also possible.

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.
- clone: See clone for further information.
- colapply: See colapply for further information.
• cols: See cols for further information.
• getCol: See getCol for further information.
• merge: See merge for further information.
• nas: See nas for further information.
• plot: See plot for further information.
• print: See print for further information.
• refresh: See refresh for further information.
• rollapply: See rollapply for further information.
• rowaggregate: See rowaggregate for further information.
• rowbind: See rowbind for further information.
• setColNames: See setColNames for further information.
• setCols: See setCols for further information.
• subset: See subset 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 deep clone (copy) of the object is made beforehand. See clone for further information. Some of the fields are read-only though:

• aggregated: Same as the aggregated argument.
• fast: Same as the fast argument.
• funbyApproach: Same as the funbyApproach argument.
• ID: Same as the ID argument. It is used as the title of plots.
• na.status: Same as the na.status argument. When set, the missing values of the object are expanded or collapsed accordingly.
• parameter: Same as the 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. When set, the periodicity of the time series is changed as specified. See the by argument of alter for further information.
• 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 showing the time zone of the time series. When set, the series is converted to the specified time zone. Only names from OlsonNames are accepted.

unit: Same as the unit argument. It is added to the label of the primary axis of plots when the parameter field is set.

variant: Same as the variant argument. It is added to the label of the primary axis of plots when the parameter field is set.

The parameter, unit and variant fields are especially useful for time series of a single variable. For time series of multiple variables with differing units the functionality of the units package may pose a viable alternative.

Options

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

- **DTsClone**: A logical specifying if DTs objects are, by default, modified in place (FALSE) or if a deep clone (copy) shall be made beforehand (TRUE).
- **DTsDeprecatedWarnings**: A logical specifying if warnings are displayed when calling deprecated features.
- **DTsFast**: Default value for the fast argument.
- **DTsFunbyApproach**: Default value for the funbyApproach argument.
- **DTsNA.status**: Default value for the na.status argument.

Note

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

Examples

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

## abused S4 constructor
gen(
Class = "DTs",
values = flow,
ID = "River Flow"
)
```
flow

**Daily river flows**

**Description**

A dataset containing a fictional time series of daily river flows with implicitly missing values.

**Usage**

flow

**Format**

A `data.table::data.table` with 2169 rows and two 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 in cubic metres per second.

---

getCol.DTSg

**Get column vector**

**Description**

Returns the values of a column of a DTSg object.

The extract operator ([) acts as a shortcut for getCol.

**Usage**

```r
## S3 method for class 'DTSg'
getcoll(x, col = self$cols(class = "numeric")[1L], ...)
```

## S3 method for class 'DTSg'

```r
x[...]
```

**Arguments**

- **x** A DTSg object (getCol S3 method only).
- **col** A character string specifying a column name.
- **...** Arguments passed on to getCol (only used by the extract operator).

**Value**

Returns a vector or a list in case of a list column.
See Also
cols

Examples

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

# get the first ten values of the “flow” column
## R6 methods
x$getCol(col = “flow”)[1:10]
x$`\`{“flow”}[1:10]

## S3 methods
getCol(x = x, col = “flow”)[1:10]
x[“flow”][1:10]

interpolateLinear  Linear interpolation

Description

Linearly interpolates missing values of a numeric vector. For use with the colapply method of DTSg objects. Other uses are possible, but not recommended.

This function also serves as an example for writing user defined functions utilising one of the lists with helper data as handed over by some of the methods of DTSg objects.

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 interpolated. For time series with unrecognised periodicity it is interpreted in seconds and for time series with recognised periodicity it is multiplied with the maximum time difference between two subsequent time steps in seconds. Thus, 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. See data.table::data.table for further information.

.helpers  A list with helper data as handed over by colapply. See colapply for further information.
merge.DTSg

Value
Returns a numeric vector.

Examples

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

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

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

merge.DTSg  

Merge two objects

Description
Joins two DTSg objects based on their .dateTime column. Their time zones and aggregated fields must match.

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 data.table::merge. As the by, by.x and by.y arguments can endanger the object’s integrity, they are rejected.
clone  A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DTSg object.

See Also

getOption
Examples

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

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

## S3 method
print(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. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".

...

Not used (S3 method only).

Value

Returns a data.table::data.table with five columns:

• .col: the column name
• .group: the ID of the missing values group within each column
• .from: the first timestamp of the missing values group
• .to: the last timestamp of the missing values group
• .n: the number of missing values per group
**plot.DTSG**

See Also

cols

Examples

```r
# 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 data*

**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 data. The title of the plot and the label of its primary axis are automatically generated from the object’s metadata (fields). See DTSG for further information.

**Usage**

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

**Arguments**

- `x`  
  A DTSG object (S3 method only).

- `from`  
  A POSIXct timestamp in the same time zone as the time series or a character string coercible to one. The data is plotted from this timestamp on.

- `to`  
  A POSIXct timestamp in the same time zone as the time series or a character string coercible to one. The data is plotted up to this timestamp.
cols A character vector specifying the columns whose values shall be plotted. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".

secAxisCols An optional character vector specifying the columns whose values shall be plotted on a secondary axis. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z". Must be a subset of cols.

secAxisLabel A character string specifying the label of the optional secondary axis.

Value

Returns a DTSg object.

See Also

cols

Examples

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

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

  ## S3 method
  plot(x = x)
}

print.DTSg  
Description

Prints a DTSg object.

Usage

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

print object
**refresh.DTSg**

**Arguments**

- **x**: A `DTSg` object (S3 method only).
- **...**: Not used (S3 method only).

**Value**

Returns a `DTSg` object.

**Examples**

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

# print object
## R6 method
dx$print()

## S3 method
print(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**

```r
## 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.
Examples

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

# check the object's integrity
## R6 method
x$refresh()

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

---

rollapply.DTG   
**Rolling window function**

Description

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

Usage

```r
## S3 method for class 'DTG'
rollapply(
x, fun, ...
 cols = self$cols(class = "numeric")[1L],
 before = 1L,
 after = before,
 weights = "inverseDistance",
 parameters = list(power = 1),
 resultCols = NULL,
 suffix = NULL,
 helpers = TRUE,
 memoryOverCPU = TRUE,
 clone =getOption("DTGClone")
)
```

Arguments

- `x` A DTG 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. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".
before  An integerish value specifying the size of the window in time steps before the “center” of the rolling window.

after   An integerish value specifying the size of the window in time steps after the “center” of the rolling window.

weights A character string specifying the method applied to calculate the weights handed over to fun. These are useful for functions like weighted.mean. See corresponding section for further information.

parameters A list specifying parameters for the weight calculation method. See corresponding section for further information.

resultCols An optional character vector of the same length as cols specifying the column names for the return values of fun. Another possibility is a character string containing comma separated column names, for example, "x,y,z". Non-existing columns are added and existing columns are overwritten. Columns are matched element-wise between cols and resultCols.

suffix   An optional character string. The return values of fun are added as new columns with names consisting of the columns specified in cols and this suffix. Existing columns are never overwritten. Only used when resultCols is not specified.

helpers  A logical specifying if helper data shall be handed over to fun. See corresponding section for further information.

memoryOverCPU A logical specifying if memory usage shall be preferred over CPU usage for this method call. The former is generally faster for smaller windows and shorter time series, the latter for bigger windows and longer time series or might even be the only one which works depending on the available hardware.

clone    A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DTSg object.

Weights

Currently, only one method to calculate weights is supported: "inverseDistance". The distance $d$ of the “center” is one and each time step further 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 parameter $p$ as a named element of a list provided through the parameters argument: $d^p$.

Helper data

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

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

Rollback of months

Description
Generating regular sequences of time 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 fixes this by counting the days of affected months backwards again.

Usage
rolloback(.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.

See Also
colsgetOption

Examples

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

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

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

Value

Returns a POSIXct vector.

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
)

rowaggregate.DTSg  Aggregate values row-wise

Description

Applies one or more provided summary functions row-wise to selected columns of a DTSg object.

Usage

## S3 method for class 'DTSg'
rowaggregate(
  x,
  resultCols,
  fun,
  ...
  cols = self$cols(class = "numeric"),
  clone = getOption("DTSgClone")
)

Arguments

x  A DTSg object (S3 method only).
resultCols  A character vector either of length one (names of fun are appended in the case one or more functions are provided) or the same length as fun specifying the column names for the return values of fun.
fun  A summary function, (named) list of summary functions or (named) character vector specifying summary functions applied row-wise to all the values of the specified cols. The return value(s) must be of length one. See corresponding section for further information.
...  Further arguments passed on to fun.
cols

A character vector specifying the columns to apply fun to. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".

clone

A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DT$g object.

Summary functions

Some examples for fun are as follows:

- `mean`
- `list(min = min, max = max)`
- `c(sd = "sd", var = "var")`

See Also

cols,getOption

Examples

# new DT$g object
DT <- data.table::data.table(
  date = flow$date,
  flow1 = flow$flow - abs(rnorm(nrow(flow))),
  flow2 = flow$flow,
  flow3 = flow$flow + abs(rnorm(nrow(flow)))
)
x <- DT$g$new(values = DT)

# mean and standard deviation of multiple measurements per timestamp
## R6 method
x$rowaggregate(
  resultCols = "flow",
  fun = list(mean = mean, sd = sd)
)$print()

## 'raggregate()' is a "hidden" R6 alias for 'rowaggregate()'
x$raggregate(
  resultCols = "flow",
  fun = list(mean = mean, sd = sd)
)$print()

## S3 method
print(rowaggregate(
  x = x,
  resultCols = "flow")
fun = list(mean = mean, sd = sd)
)

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

Arguments

x A DTSg object (S3 method only).
...
Any number of DTSg objects or objects coercible to one (see new for further information). lists of such objects or a mixture of lists and non-lists are also accepted.
clone A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DTSg object.

See Also
cols,getOption

Examples

# new DTSg object
x <- DTSg$new(values = flow[1:500, ])

# combine rows
## R6 method
x$rowbind(
  list(flow[1001:1500, ], DTSg$new(values = flow[501:1000, ])),
  flow[1501:.N, ]
)$print()

## 'rbind()' is a "hidden" R6 alias for 'rowbind()
x$rbind(
  list(flow[1001:1500, ], DTSg$new(values = flow[501:1000, ])),
  flow[1501:.N, ]
)

Description

Combines the rows of DTSg and other suitable objects.

Usage

Combine rows

Arguments

x A DTSg object (S3 method only).
...
Any number of DTSg objects or objects coercible to one (see new for further information). lists of such objects or a mixture of lists and non-lists are also accepted.
clone A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

Value

Returns a DTSg object.

See Also
cols,getOption

Examples

# new DTSg object
x <- DTSg$new(values = flow[1:500, ])

# combine rows
## R6 method
x$rowbind(
  list(flow[1001:1500, ], DTSg$new(values = flow[501:1000, ])),
  flow[1501:.N, ]
)$print()

## 'rbind()' is a "hidden" R6 alias for 'rowbind()
x$rbind(
  list(flow[1001:1500, ], DTSg$new(values = flow[501:1000, ])),
  flow[1501:.N, ]
)
S3WrapperGenerator

S3 wrapper method generator

## Description

Generates S3 wrapper methods for public methods of R6ClassGenerators, but can also be used to generate “plain” function wrappers.

## Usage

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

## Arguments

- **R6Method**: An expression with or a public method (function) 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 an S3 method (function).

## See Also

S3Methods, R6::R6Class

## Examples

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

Changes the column names of DTSg objects.

Usage

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

Arguments

- `x` A DTSg object (S3 method only).
- `cols` A character vector specifying the columns whose names shall be set. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z". The name of the .dateTime column cannot be changed.
- `values` A character vector of the same length as `cols` specifying the desired column names. Another possibility is a character string containing comma separated column names, for example, "x,y,z".
- `clone` A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.
- `...` Not used (S3 method only).

Value

Returns a DTSg object.

See Also

cols, getOption

Examples

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

# rename column "flow" to "River Flow"
## R6 method
```
setCols.DTSG

Set column values

Description

Changes the values of columns, adds columns to and/or removes columns from a DTSG object. The values can optionally be set for certain rows only.

Usage

## S3 method for class 'DTSG'
setCols(
x, 
i, 
cols = self$cols(class = "numeric")[,1L], 
values, 
clone = getOption("DTSGClone"), 
...
)

Arguments

x A DTSG object (S3 method only).
i An integerish vector indexing rows (positive numbers pick and negative numbers omit rows) or a filter expression accepted by the i argument of data.table::data.table. Filter expressions can contain the special symbol .N.
cols A character vector specifying the columns whose values shall be set. Another possibility is a character string containing comma separated column names, for example, "x,y,z". The values of the .dateTime column cannot be changed.
setCols.DTSg

values
A vector, list or list-like object (e.g. data.table::data.table) of replacement and/or new values accepted by the value argument of data.table’s data.table::set function. NULL as a value removes a column.

clone
A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

... Not used (S3 method only).

Value
Returns a DTSg object.

See Also
cols,getOption

Examples

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

# cap river flows to 100
## R6 method
x$setCols(
  i = flow > 100,
  cols = "flow",
  values = 100
)$print()

## 'set()' is a “hidden” R6 alias for 'setCols()' 

x$set(
  i = flow > 100,
  cols = "flow",
  values = 100
)$print()

## S3 method
print(setCols(
  x = x,
  i = flow > 100,
  cols = "flow",
  values = 100
))

# set measurement unit with the help of 'units'
if (requireNamespace("units", quietly = TRUE)) {
  ## R6 method
  x$setCols(
    cols = "flow",
    values = units::set_units(x["flow"], "m^3/s")
  )$print()}
### S3 method
print(setCols(
  x = x,
  cols = "flow",
  values = units::set_units(x["flow"], "m^3/s")
))

**subset.DTSg**  
*Subset time series data*

**Description**

Filters rows and/or selects columns of a *DTSg* object.

**Usage**

```r
## S3 method for class 'DTSg'
subset(
  x,
  i,
  cols = self$cols(),
  funby = NULL,
  ignoreDST = FALSE,
  na.status = "implicit",
  clone = getOption("DTSgClone"),
  multiplier = 1L,
  funbyHelpers = NULL,
  funbyApproach = self$funbyApproach,
  ...
)
```

**Arguments**

- **x** A *DTSg* object (S3 method only).
- **i** An integerish vector indexing rows (positive numbers pick and negative numbers omit rows) or a filter expression accepted by the `i` argument of `data.table::data.table`. Filter expressions can contain the special symbol `.N`.
- **cols** A character vector specifying the columns to select. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z". The `.dateTime` column is always selected and cannot be part of it.
- **funby** One of the temporal aggregation level functions described in *TALFs* or a user defined temporal aggregation level function. Can be used to, for instance, select the last two observations of a certain temporal level. See corresponding section and examples for further information.
ignoreDST  A logical specifying if day saving time shall be ignored by funby. See corresponding section for further information.

na.status  A character string. Either "explicit", which makes missing timestamps explicit according to the recognised periodicity, or "implicit", which removes timestamps with missing values on all value columns. See corresponding section for further information.

cloned  A logical specifying if the object shall be modified in place or if a deep clone (copy) shall be made beforehand.

multiplier  A positive integerish value “multiplying” the temporal aggregation level of certain TALFs. See corresponding section for further information.

funbyHelpers  An optional list with helper data passed on to funby. See corresponding section for further information.

funbyApproach  A character string specifying the flavour of the applied temporal aggregation level function. Either "base", which utilises as.POSIXct, or "fasttime", which utilises fasttime::fastPOSIXct, or "RcppCCTZ", which utilises RcppCCTZ::parseDatetime as the main function for transforming timestamps.

... Further arguments passed on to fun.

Value

Returns a DTSg object.

Status of missing values

Please note that filtering rows and having or making missing timestamps explicit equals to setting the values of all other timestamps to missing. The default value of na.status is therefore "implicit". To simply filter for a consecutive range of a DTSg object while leaving the na.status untouched, alter is probably the better choice.

User defined TALFs, TALFs helper data and multiplier

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. The default elements of this list are as follows:

- timezone: Same as the timezone field.
- ignoreDST: Same as the ignoreDST argument.
- periodicity: Same as the periodicity field.
- na.status: Same as the na.status field.
- multiplier: Same as the multiplier argument.
- funbyApproach: Same as the funbyApproach argument.

Any additional element specified in the funbyHelpers argument is appended to the end of the helper data list. In case funbyHelpers contains an ignoreDST, multiplier or funbyApproach element, it takes precedence over the respective method argument. timezone, periodicity and na.status elements are rejected, as they are always taken directly from the object.
The temporal aggregation level of certain \texttt{TALFs} can be adjusted with the help of the multiplier argument. A multiplier of 10, for example, makes \texttt{byY_____} aggregate to decades instead of years. Another example is a multiplier of 6 provided to \texttt{by_m_____}. The function then aggregates all months of all first and all months of all second half years instead of all months of all years separately. This feature is supported by the following \texttt{TALFs} of the package:

- \texttt{byY_____}
- \texttt{byYm_____}
- \texttt{byYmdH__} (UTC and equivalent as well as all Etc/GMT time zones only)
- \texttt{byYmdHM_}
- \texttt{byYmdHMS}
- \texttt{by_m_____}
- \texttt{by___H__} (UTC and equivalent as well as all Etc/GMT time zones only)
- \texttt{by_____M_}
- \texttt{by_____S}

\textbf{Ignore day saving time}

g\texttt{ignoreDST} tells a temporal aggregation level function if it is supposed to ignore day saving time while transforming the 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 by all means and all intervals are of the “correct” length, however, a possible limitation might be that the day saving time shift is invariably assumed to be one hour long. This feature requires that the periodicity of the time series was recognised and is supported by the following \texttt{TALFs} of the package:

- \texttt{byY_____}
- \texttt{byYQ_____}
- \texttt{byYm_____}
- \texttt{byYmd____}
- \texttt{byYmd___}
- \texttt{by_Q_____}
- \texttt{by_m_____}
- \texttt{by___H__}

\textbf{See Also}

c\texttt{ols}, \texttt{getOption}

\textbf{Examples}

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

# filter for the first six observations
## R6 method
x$subset(i = 1:6)$print()
```
## S3 method
print(subset(x = x, i = 1:6))

# filter for the last two observations per year
## R6 method
x$subset(
  i = (.N - 1):.N,
  funby = function(x, ...) {data.table::year(x)}
)$print()

## S3 method
print(subset(
  x = x,
  i = (.N - 1):.N,
  funby = function(x, ...) {data.table::year(x)}
))

summary.DTSg  

---

**summary.DTSg**  

*Summarise time series data*

---

### Description

Calculates summary statistics of selected columns of a DTSg object.

### Usage

```r
## 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. Another possibility is a character string containing either comma separated column names, for example, "x,y,z", or the start and end column separated by a colon, for example, "x:z".

- **...**  
  Further arguments passed on to `summary.data.frame`.

### Value

Returns a table.

### See Also

- `cols`
Examples

```r
# new DT$g object
x <- DT$g$new(values = flow)

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

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

---

**TALFs**

Temporal Aggregation Level Functions (TALFs)

Description

Simply hand over one of these functions to the funby argument of one of the methods of a DT$g object, which supports it. The method then does the rest of the work. See respective calling method for further information. Other uses are possible, but not recommended.

Usage

```r
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)
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)

Arguments

.dateTime A POSIXct vector.
.helpers A list with helper data as handed over by methods of DT$s objects, which support the funby argument.

Value

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

Families and flavours

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 three flavours: the first relies solely on base R, the second utilises fasttime::fastPOSIXct of fasttime and the third RcppCCTZ::parseDatetime of RcppCCTZ.

The fasttime flavour works with UTC and equivalent as well as all Etc/GMT time zones only (execute grep("^\(Etc/\)?(UCT|UTC)$\|^\(Etc/\)?GMT(\+\-)?\d\d\$", OlsonNames(), ignore.case
values.DTSg = TRUE, value = TRUE) for a full list of supported time zones) and is limited to timestamps between the years 1970 and 2199, but generally is the fastest for the extracting family of functions. For time zones other than UTC and equivalent the ReppCCTZ flavour generally is the fastest.

Use the funbyApproach argument of the respective calling method in order to specify the utilised flavour.

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

- `byY_____` truncates to year, e.g. 2000-11-11 11:11:11.1 becomes 2000-01-01 00:00:00.0
- `byYQ_____` truncates to quarter, e.g. 2000-11-11 11:11:11.1 becomes 2000-10-01 00:00:00.0
- `byYm_____` truncates to month, e.g. 2000-11-11 11:11:11.1 becomes 2000-11-01 00:00:00.0
- `byYmd____` truncates to day, e.g. 2000-11-11 11:11:11.1 becomes 2000-11-11 00:00:00.0
- `byYmdH__` truncates to hour, e.g. 2000-11-11 11:11:11.1 becomes 2000-11-11 11:00:00.0
- `byYmdHM_` truncates to minute, e.g. 2000-11-11 11:11:11.1 becomes 2000-11-11 11:11:00.0
- `byYmdHMS` 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:

- `by_______` extracts nothing, i.e. all timestamps become 2199-01-01 00:00:00.0
- `by_Q_____` extracts the quarters, e.g. 2000-11-11 11:11:11.1 becomes 2199-10-01 00:00:00.0
- `by_m_____` extracts the months, e.g. 2000-11-11 11:11:11.1 becomes 2199-11-01 00:00:00.0
- `by___H__` extracts the hours, e.g. 2000-11-11 11:11:11.1 becomes 2199-01-01 11:00:00.0
- `by_____M_` extracts the minutes, e.g. 2000-11-11 11:11:11.1 becomes 2199-01-01 00:11:00.0
- `by_____S` extracts the seconds, e.g. 2000-11-11 11:11:11.1 becomes 2199-01-01 00:00:11.0

Please note that the byFasttime* versions are deprecated.

See Also

aggregate, colapply, subset

values.DTSg

Get values

Description

Returns the values of a DTSg object.

Usage

```r
## 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 shall be returned. See corresponding section for further information.

- **drop**  
  A logical specifying if the object and all references to it shall be removed from the global (and only the global) environment after successfully returning its values. This feature allows for a resource 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 when either a copy of the values is returned or the object is dropped.

- **...**  
  Not used (S3 method only).

**Value**

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

**Reference to the values**

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.

**Note**

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

**Examples**

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

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

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

* datasets
  flow, 19
  .N, 34, 36
  [.DTSg (getCol.DTSg), 19

  aggregate, 16, 42
  aggregate (aggregate.DTSg), 2
  aggregate.DTSg, 2
  aggregated, 2
  alter, 16, 17, 37
  alter (alter.DTSg), 6
  alter.DTSg, 6
  as.POSIXct, 3, 10, 16, 37

  by______ (TALFs), 40
  by______S, 4, 11, 38
  by______S (TALFs), 40
  by______M_, 4, 11, 38
  by______M (TALFs), 40
  by______H_, 4, 5, 11, 38
  by______H (TALFs), 40
  by______H__ (TALFs), 40
  by______H___, 5, 11, 38
  by______H (TALFs), 40

  byFasttime______ (TALFs), 40
  byFasttime______S (TALFs), 40
  byFasttime______M_ (TALFs), 40
  byFasttime______H__ (TALFs), 40
  byFasttime______m_____ (TALFs), 40
  byFasttime______Q_____ (TALFs), 40
  byFasttime______Y_____ (TALFs), 40
  byFasttime______Ym_____ (TALFs), 40
  byFasttime______Ymd____ (TALFs), 40
  byFasttime______YmdH___ (TALFs), 40
  byFasttime______YmdHM_ (TALFs), 40
  byFasttime______YmdHMS (TALFs), 40
  byFasttime______YQ_____ (TALFs), 40

  byY______, 4, 5, 11, 38
  byY______ (TALFs), 40

  byYm______, 4, 5, 11, 38
  byYm______ (TALFs), 40
  byYmd____, 5, 11, 38
  byYmd____ (TALFs), 40
  byYmdH___, 4, 11, 38
  byYmdH___ (TALFs), 40
  byYmdHM_, 4, 11, 38
  byYmdHM_ (TALFs), 40
  byYmdHMS, 4, 11, 38
  byYmdHMS (TALFs), 40
  byYQ______, 5, 11, 38
  byYQ______ (TALFs), 40

  class, 14
  clone, 16, 17
  clone (clone.DTSg), 8
  clone.DTSg, 8
  colapply, 16, 20, 42
  colapply (colapply.DTSg), 9
  colapply.DTSg, 9
  cols, 5, 11, 17, 20, 23, 24, 28, 30, 31, 33, 35, 38, 39
  cols (cols.DTSg), 14
  cols.DTSg, 14
  cumsum, 10

  data.frame, 16, 43
  data.table::data.table, 16, 19, 20, 22, 34–36, 43
  data.table::merge, 21
  data.table::set, 35
  data.table::setkey, 16
  difftime, 10, 17
  DTSg, 2, 3, 6–10, 14, 15, 19–27, 29–31, 33–37, 39–43

  expression, 32

  fasttime::fastPOSIXct, 3, 10, 16, 37, 41
  flow, 19
INDEX

function, 9, 20, 26, 32
getCol, 17
getcoll(getCol.DTSg), 19
getcoll.DTSg, 19
getOption, 5, 7, 11, 21, 28, 30, 31, 33, 35, 38
GForce, 4
grep, 14
integer, 14
interpolateLinear, 20
list, 3, 4, 10, 11, 19, 20, 27, 29–31, 35, 37, 41
max, 4, 30
mean, 4, 30
merge, 17
merge (merge.DTSg), 21
merge.DTSg, 21
min, 4, 30
mode, 14
na.status, 4, 10, 37
nas, 7, 17
nas(nas.DTSg), 22
nas.DTSg, 22
new, 21, 31
new(DTSg), 15
numeric, 14
OlsonNames, 18
options, 8, 18
periodicity, 4, 10, 37
plot, 17
plot(plot.DTSg), 23
plot.DTSg, 23
POSIXct, 3, 7, 10, 16, 18, 19, 23, 28, 29, 37, 41
print, 17
print(print.DTSg), 24
print.DTSg, 24
R6::R6class, 8, 15, 32
raggregate (raggregate.DTSg), 29
rbind(rbind.DTSg), 31
RcppCCTZ::parseDateTime, 3, 10, 16, 37, 41
refresh, 17, 43
refresh (refresh.DTSg), 25
refresh.DTSg, 25
rollapply (rollapply.DTSg), 17
rollapply.DTSg, 26
rollback, 7, 28
rowaggregate, 17
rowaggregate (rowaggregate.DTSg), 29
rowaggregate.DTSg, 29
rowbind, 17
rowbind (rowbind.DTSg), 31
rowbind.DTSg, 31
S3Methods, 32
S3WrapperGenerator, 32
seq.POSIXt, 7, 28
set(setCols.DTSg), 34
setColNames, 17
setColNames (setColNames.DTSg), 33
setColNames.DTSg, 33
setCols, 17
setCols (setCols.DTSg), 34
setCols.DTSg, 34
setnames (setColNames.DTSg), 33
subset, 7, 17, 42
subset (subset.DTSg), 36
subset.DTSg, 36
summary, 17
summary (summary.DTSg), 39
summary.data.frame, 39
summary.DTSg, 39
table, 39
TALFs, 3–5, 10, 11, 16, 36–38, 40
timezone, 4, 10, 37
typeof, 14
values, 17, 25
values (values.DTSg), 42
values.DTSg, 42
weighted.mean, 27