

Package ‘DTSg’

July 12, 2020

Type Package

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

Version 0.6.0

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 inclusive an S3 wrapper method generator for those who prefer the latter. Finally yet importantly, its functional approach allows incorporating functionalities from many other packages.

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 checkmate, data.table, methods, R6

Suggests covr, dygraphs, fasttime, knitr, magrittr, RColorBrewer, rmarkdown, runner (>= 0.3.5), tinytest

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Gerold Hepp [aut, cre]

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

Repository CRAN

Date/Publication 2020-07-12 07:00:02 UTC

R topics documented:

aggregate.DTSg	2
alter.DTSg	4
clone.DTSg	6
colapply.DTSg	7
cols.DTSg	9
DTSg	10
flow	14
getCol.DTSg	14
interpolateLinear	15
merge.DTSg	16
nas.DTSg	17
plot.DTSg	18
print.DTSg	19
refresh.DTSg	20
rollapply.DTSg	21
rollback	23
S3WrapperGenerator	24
setCols.DTSg	24
subset.DTSg	26
summary.DTSg	27
TALFs	28
values.DTSg	30
Index	32

aggregate.DTSg	<i>Aggregate Values</i>
----------------	-------------------------

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. 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, list of summary functions or character vector specifying summary functions applied column-wise to all the values of the same temporal aggregation level, for instance, mean , list (min = min , max = max) or <code>c(sd = "sd", var = "var")</code> . 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 <i>GForce</i> optimisation in case a character vector is used. The return value(s) 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 <i>.n</i> 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.
- *na.status*: Same as *na.status* 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 and all intervals are of "correct" length, however, a possible limitation might be that the

day saving time shift is invariably assumed to be exactly one hour long. This feature requires that the periodicity of the time series is recognised 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](#), [list](#), [GForce](#), [cols](#), [POSIXct](#)

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)

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

## S3 method
aggregate(x = x, funby = byYQ_____, fun = c(sd = "sd", var = "var"), na.rm = TRUE)
```

alter.DTSg

Alter Time Series

Description

Shortens, lengthens, subsets a complete range, changes the periodicity and/or the status of missing values of a [DTSg](#) object.

Usage

```
## 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 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.
na.status	A character string. Either "explicit", which makes missing timestamps according to the recognised periodicity explicit, or "implicit", which removes timestamps with missing values on all value columns. Please note that DTSg objects work best with explicit missing values.
...	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)

# subset 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")

# change periodicity to one month
## R6 method
x$alter(by = "1 month")

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

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

<code>x</code>	A DTSg object (S3 method only).
<code>deep</code>	A logical specifying if a deep copy is made (for consistency with R6Class the default is <code>FALSE</code> , but should generally be set to <code>TRUE</code>).
<code>...</code>	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 Column-wise

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"),
  resultCols = NULL,
  suffix = NULL,
  funby = NULL,
  ignoreDST = FALSE
)
```

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.
resultCols	An optional character vector of the same length as cols. Non-existing columns specified in this argument are added and existing columns are overwritten by the return values of fun. Columns are matched element-wise between resultCols and cols.

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.
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 examples and aggregate for further information.
ignoreDST	A logical specifying if day saving time is ignored during formation of the temporal level. See aggregate for further information.

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](#), [function](#), [cols](#), [TALFs](#), [aggregate](#), [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)

# daily cumulative sums per month
## R6 method
x$colapply(fun = function(x, ...) {cumsum(x)}, funby = byYm____)

## S3 method
colapply(x = x, fun = function(x, ...) {cumsum(x)}, 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") >= numeric_version("0.3.5")) {
  wrapper <- function(..., .helpers) {
    runner::runner(..., idx = .helpers[[".dateTime"]])
  }

  ## R6 method
  x$colapply(fun = runner::runner, f = mean, k = 5, lag = -2)
  x$colapply(fun = wrapper, f = mean, k = "5 days", lag = "-2 days")
  x$colapply(
    fun = runner::runner,
    f = mean,
    k = "5 days",
    lag = "-2 days",
    idx = x$getCol(col = ".dateTime")
  )
  x$colapply(
    fun = runner::runner,
    f = mean,
    k = "5 days",
    lag = "-2 days",
    idx = x[[".dateTime"]]
  )

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

```

Description

Queries all column names of a [DTSg](#) object, those of certain [classes](#) and/or those matching a certain pattern only.

Usage

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

Arguments

<code>x</code>	A DTSg object (S3 method only).
<code>class</code>	An optional character vector matched to the most specific class (first element) of each column's class vector.
<code>pattern</code>	An optional character string passed on to the <code>pattern</code> argument of grep .
<code>...</code>	Further arguments passed on to grep . The value argument is rejected.

Value

Returns a character vector.

See Also

[DTSg](#), [class](#), [grep](#)

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")
```

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, na.status = c("explicit",
     "implicit", "undecided"))
```

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 <code>.dateTime</code> .
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 resource 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 (and only the global) environment are removed upon successful creation of a DTSg object.
na.status	A character string. Either "explicit", which makes missing timestamps according to the recognised periodicity explicit, 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.

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.
- `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 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.
- *na.status*: Same as `na.status` argument. When set, the *values* of the object are expanded or collapsed accordingly.
- *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. When set, the periodicity of the time series is changed as specified. See 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 containing 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 `unit` argument. It is added to the label of the primary axis of plots when the `parameter` field is set.
- *variant*: Same as `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 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 for `options("digits.secs")`. As a consequence, time series with a sub-second accuracy higher than a microsecond will never 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`, `OlsonNames`, `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	<i>Daily River Flows</i>
------	--------------------------

Description

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

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 in cubic metres per second.

getCol.DTSg	<i>Get Column Vector</i>
-------------	--------------------------

Description

Queries the values of a column of a [DTSg](#) object. The extract operator (`[]`) acts as a shortcut for `getCol`.

Usage

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

```
## S3 method for class 'DTSg'
x[...]
```

Arguments

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

Value

Returns a vector or a [list](#) in case of a [list](#) column.

See Also

[DTSg](#), [cols](#), [list](#)

Examples

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

# get values of "flow" column
## R6 methods
x$getCol(col = "flow")
x$`["flow"]`

## S3 methods
getCol(x = x, col = "flow")
x["flow"]
```

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 [lists](#) with helper data as handed over by some methods of [DTSg](#) objects. See [DTSg](#) for further information.

Usage

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

Arguments

<code>.col</code>	A numeric vector.
<code>roll</code>	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. Thus, for regular time series it is the number of time steps and for irregular it is an approximation of it.
<code>rollends</code>	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.
<code>.helpers</code>	A list with helper data as handed over by colapply . See colapply for further information.

Value

Returns a numeric vector.

See Also

[DTSg](#), [colapply](#), [function](#), [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* fields 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 <i>by</i> , <i>by.x</i> and <i>by.y</i> arguments can endanger the integrity of the object, they are rejected.
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$values(reference = TRUE)[[".dateTime"]]),
  to = last(self$values(reference = TRUE)[[".dateTime"]]),
  cols = self$cols(class = "numeric"),
  secAxisCols = NULL,
  secAxisLabel = "",
  ...
)
```

Arguments

x	A DTSg object (S3 method only).
from	A POSIXct 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	An optional 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 optional 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)
}
```

print.DTSg

Print Time Series

Description

Prints a [DTSg](#) object.

Usage

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

Arguments

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

Value

Returns a [DTSg](#) object.

See Also

[DTSg](#)

Examples

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

# print object
## R6 method
x$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

```
## 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"),
  resultCols = NULL,
  suffix = NULL,
  memoryOverCPU = TRUE
)
```

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	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 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.

resultCols	An optional character vector of the same length as cols. Non-existing columns specified in this argument are added and existing columns are overwritten by the return values of fun. Columns are matched element-wise between resultCols and cols.
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.
memoryOverCPU	A logical specifying if memory usage is preferred over CPU usage for this method. 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 way that works depending on the available hardware.

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`, `function`, `cols`, `list`

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)

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

rollback	<i>Rollback Month</i>
----------	-----------------------

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

<code>.dateTime</code>	A POSIXct vector.
<code>periodicity</code>	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 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 <code>...</code> 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](#), [R6Class](#), [expression](#), [function](#)

Examples

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

setCols.DTSg *Set Columns' Values*

Description

Set the values of columns, add columns to and/or remove 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 <i>DTSg</i> 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 <code>i</code> argument of <code>data.table</code> . Filter expressions can contain the special symbol <code>.N</code> .
cols	A character vector specifying the columns whose values shall be set. The values of the <code>.dateTime</code> column cannot be set.
values	A vector, <code>list</code> or list-like object (e.g. <code>data.table</code>) of replacement and/or new values accepted by the <code>value</code> argument of <code>data.table</code> 's <code>set</code> function. <code>NULL</code> as a value removes a column.
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](#), [data.table](#), [.N](#), [cols](#), [list](#), [set](#)

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)

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

subset.DTSg

*Subset Time Series***Description**

Filter rows and/or select columns of a [DTSg](#) object.

Usage

```
## S3 method for class 'DTSg'
subset(
  x,
  i,
  cols = self$cols(),
  funby = NULL,
  ignoreDST = FALSE,
  na.status = "implicit",
  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 <code>i</code> argument of data.table . Filter expressions can contain the special symbol <code>.N</code> .
cols	A character vector specifying the columns to select. The <code>.dateTime</code> 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 examples and aggregate for further information.
ignoreDST	A logical specifying if day saving time is ignored during formation of the temporal level. See aggregate for further information.
na.status	A character string. Either "explicit", which makes missing timestamps according to the recognised periodicity explicit, or "implicit", which removes timestamps with missing values on all value columns. Please note that filtering rows and having or making missing timestamps explicit equals to setting the values of all other timestamps to missing. Its default value is therefore "implicit". To simply subset a complete range of a DTSg object while leaving <code>na.status</code> untouched, alter is probably the better choice.
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](#), [data.table](#), [.N](#), [cols](#), [TALFs](#), [alter](#)

Examples

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

# filter for the last six rows
## R6 method
x$subset(i = (.N - 5):.N)

## S3 method
subset(x = x, i = (.N - 5):.N)

# filter for the first two observations per year
## R6 method
x$subset(i = 1:2, funby = function(x, ...) {data.table::year(x)})

## S3 method
subset(x = x, i = 1:2, funby = function(x, ...) {data.table::year(x)})
```

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 hand over one of these functions to the `funby` argument of one of the methods (e.g. [aggregate](#)) of [DTsg](#) objects which support it. 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](#), [colapply](#), [subset](#), [fastPOSIXct](#), [list](#), [POSIXct](#)

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 <i>values</i> or a reference to the <i>values</i> 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 (and only the global) environment after successfully querying its values. This feature allows for a resource efficient destruction of a DTSg object while preserving its <i>values</i> .
class	A character string specifying the class of the returned <i>values</i> . "data.frame" only works when either a copy of the <i>values</i> 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

* datasets

- flow, 14
- .N, 25–27
- [.DTSg (getCol.DTSg), 14

- aggregate, 8, 12, 26, 28–30
- aggregate (aggregate.DTSg), 2
- aggregate.DTSg, 2
- alter, 12, 13, 26, 27
- alter (alter.DTSg), 4
- alter.DTSg, 4

- by_____ (TALFs), 28
- by_____S (TALFs), 28
- by_____M_ (TALFs), 28
- by___H__, 4
- by___H__ (TALFs), 28
- by_m____, 4
- by_m____ (TALFs), 28
- by_Q____, 4
- by_Q____ (TALFs), 28
- byFasttime_____ (TALFs), 28
- byFasttime_____S (TALFs), 28
- byFasttime_____M_ (TALFs), 28
- byFasttime___H__ (TALFs), 28
- byFasttime_m____ (TALFs), 28
- byFasttime_Q____ (TALFs), 28
- byFasttimeY_____ (TALFs), 28
- byFasttimeYm____ (TALFs), 28
- byFasttimeYmd___ (TALFs), 28
- byFasttimeYmdH__ (TALFs), 28
- byFasttimeYmdHM_ (TALFs), 28
- byFasttimeYmdHMS (TALFs), 28
- byFasttimeYQ_____ (TALFs), 28
- byY_____, 4
- byY_____ (TALFs), 28
- byYm____, 4
- byYm____ (TALFs), 28
- byYmd____, 4
- byYmd___ (TALFs), 28

- byYmdH__ (TALFs), 28
- byYmdHM_ (TALFs), 28
- byYmdHMS (TALFs), 28
- byYQ_____, 4
- byYQ_____ (TALFs), 28

- class, 10
- clone, 12
- clone (clone.DTSg), 6
- clone.DTSg, 6
- colapply, 12, 13, 15, 16, 30
- colapply (colapply.DTSg), 7
- colapply.DTSg, 7
- cols, 4, 8, 12, 15, 17, 19, 22, 25, 27, 28
- cols (cols.DTSg), 9
- cols.DTSg, 9
- cumsum, 8

- data.frame, 11, 13, 31
- data.table, 11, 13–17, 25–27, 31
- difftime, 8, 13
- DTSg, 2–8, 10, 10, 12, 14–22, 24–31
- dygraph, 19

- expression, 24

- fastPOSIXct, 29, 30
- flow, 14
- function, 7, 8, 15, 16, 21, 22, 24

- getCol, 12
- getCol (getCol.DTSg), 14
- getCol.DTSg, 14
- GForce, 3, 4
- grep, 10

- interpolateLinear, 8, 15

- list, 3, 4, 8, 13, 15, 21, 22, 25, 29, 30

- max, 3

mean, 3
merge, 12, 16, 17
merge (merge.DTSg), 16
merge.DTSg, 16
min, 3

nas, 12
nas (nas.DTSg), 17
nas.DTSg, 17
new, 16, 17
new (DTSg), 10

OlsonNames, 13
options, 13

plot, 12
plot (plot.DTSg), 18
plot.DTSg, 18
POSIXct, 3–5, 8, 11, 13, 14, 18, 19, 23, 29, 30
print, 12
print (print.DTSg), 19
print.DTSg, 19

R6Class, 6, 11, 13, 24
refresh, 12, 31
refresh (refresh.DTSg), 20
refresh.DTSg, 20
rollapply, 12, 13
rollapply (rollapply.DTSg), 21
rollapply.DTSg, 21
rollback, 5, 23

S3Methods, 24
S3WrapperGenerator, 24
sd, 3
seq.POSIXt, 5, 23
set, 25
setCols, 12
setCols (setCols.DTSg), 24
setCols.DTSg, 24
setkey, 11, 13
subset, 12, 30
subset (subset.DTSg), 26
subset.DTSg, 26
sum, 13
summary, 12
summary (summary.DTSg), 27
summary.data.frame, 27, 28
summary.DTSg, 27

table, 27, 28
TALFs, 3, 4, 8, 26, 27, 28

values, 5, 12, 20
values (values.DTSg), 30
values.DTSg, 30
var, 3

weighted.mean, 21