Package ‘timeSeries’

December 14, 2015

Title Rmetrics - Financial Time Series Objects
Date 2015-12-12
Version 3022.101.2
Author Rmetrics Core Team,
       Diethelm Wuertz [aut],
       Tobias Setz [cre],
       Yohan Chalabi [ctb]
Maintainer Tobias Setz <tobias.setz@rmetrics.org>
Description Environment for teaching
      "Financial Engineering and Computational Finance".
      Managing financial time series objects.
Depends R (>= 2.10), graphics, grDevices, stats, methods, utils,
       timeDate (>= 2150.95)
Suggests RUnit, robustbase, xts, PerformanceAnalytics, fTrading
Note SEVERAL PARTS ARE STILL PRELIMINARY AND MAY BE CHANGED IN THE
      FUTURE. THIS TYPICALLY INCLUDES FUNCTION AND ARGUMENT NAMES, AS
      WELL AS DEFAULTS FOR ARGUMENTS AND RETURN VALUES.
LazyData yes
License GPL (>= 2)
URL http://www.rmetrics.org
NeedsCompilation no
Repository CRAN
Date/Publication 2015-12-14 08:24:31

R topics documented:

timeSeries-package ......................................................... 3
aggregate-methods ....................................................... 7
align-methods ............................................................ 8
apply ................................................................. 9
as ............................................................... 11
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>attach</td>
<td>13</td>
</tr>
<tr>
<td>attributes</td>
<td>14</td>
</tr>
<tr>
<td>base-methods</td>
<td>15</td>
</tr>
<tr>
<td>bind</td>
<td>15</td>
</tr>
<tr>
<td>colCum</td>
<td>16</td>
</tr>
<tr>
<td>colStats</td>
<td>17</td>
</tr>
<tr>
<td>comment</td>
<td>18</td>
</tr>
<tr>
<td>cumulated</td>
<td>19</td>
</tr>
<tr>
<td>DataPart, timeSeries-method</td>
<td>20</td>
</tr>
<tr>
<td>description</td>
<td>20</td>
</tr>
<tr>
<td>diff</td>
<td>21</td>
</tr>
<tr>
<td>dimnames</td>
<td>22</td>
</tr>
<tr>
<td>drawdowns</td>
<td>23</td>
</tr>
<tr>
<td>durations</td>
<td>24</td>
</tr>
<tr>
<td>filter</td>
<td>25</td>
</tr>
<tr>
<td>finCenter</td>
<td>26</td>
</tr>
<tr>
<td>is.timeSeries</td>
<td>27</td>
</tr>
<tr>
<td>isRegular</td>
<td>27</td>
</tr>
<tr>
<td>isUnivariate</td>
<td>29</td>
</tr>
<tr>
<td>lag</td>
<td>30</td>
</tr>
<tr>
<td>math</td>
<td>30</td>
</tr>
<tr>
<td>merge</td>
<td>32</td>
</tr>
<tr>
<td>model.frame</td>
<td>32</td>
</tr>
<tr>
<td>monthly</td>
<td>33</td>
</tr>
<tr>
<td>na</td>
<td>35</td>
</tr>
<tr>
<td>na.contiguous</td>
<td>37</td>
</tr>
<tr>
<td>orderColnames</td>
<td>38</td>
</tr>
<tr>
<td>orderStatistics</td>
<td>39</td>
</tr>
<tr>
<td>periodical</td>
<td>40</td>
</tr>
<tr>
<td>plot-methods</td>
<td>41</td>
</tr>
<tr>
<td>print-methods</td>
<td>44</td>
</tr>
<tr>
<td>rank</td>
<td>45</td>
</tr>
<tr>
<td>readSeries</td>
<td>46</td>
</tr>
<tr>
<td>returns</td>
<td>47</td>
</tr>
<tr>
<td>rev</td>
<td>49</td>
</tr>
<tr>
<td>rollMean</td>
<td>49</td>
</tr>
<tr>
<td>rowCum</td>
<td>51</td>
</tr>
<tr>
<td>runlengths</td>
<td>51</td>
</tr>
<tr>
<td>sample</td>
<td>52</td>
</tr>
<tr>
<td>scale</td>
<td>52</td>
</tr>
<tr>
<td>series-methods</td>
<td>53</td>
</tr>
<tr>
<td>smooth</td>
<td>54</td>
</tr>
<tr>
<td>sort</td>
<td>56</td>
</tr>
<tr>
<td>SpecialDailySeries</td>
<td>57</td>
</tr>
<tr>
<td>splits</td>
<td>59</td>
</tr>
<tr>
<td>spreads</td>
<td>59</td>
</tr>
<tr>
<td>start</td>
<td>61</td>
</tr>
<tr>
<td>str-methods</td>
<td>61</td>
</tr>
</tbody>
</table>
Description

Package of time series tools and utilities.

Details

Package: timeSeries
Type: Package
Version: see description file
Date: 2011
License: GPL Version 2 or later
Copyright: (c) 1999-2014 Rmetrics Association
URL: http://www.rmetrics.org

timeSeries - S4 timeSeries Class

timeSeries
getdatapart, series
getUnits
getTime, time
use: slot
getFinCenter, finCenter
use: slot
getTitle
use: slot

Creates a 'timeSeries' from scratch
Extracts the time series units
Extracts the positions of timestamps
Extracts the format of the timestamp
Extracts the financial center
Extracts the record IDs
Extracts the title
Extracts the documentation
Base Time Series Functions

- **apply**: Applies a function to blocks of a 'timeSeries'
- **attach**: Attaches a 'timeSeries' to the search path
- **cbind**: Combines columns of two 'timeSeries' objects
- **rbind**: Combines rows of two 'timeSeries' objects
- **diff**: Returns differences of a 'timeSeries' object
- **dim**: Returns dimensions of a 'timeSeries' object
- **merge**: Merges two 'timeSeries' objects
- **rank**: Returns sample ranks of a 'timeSeries' object
- **rev**: Reverts a 'timeSeries' object
- **sample**: Resamples a 'timeSeries' object
- **scale**: Scales a 'timeSeries' object
- **sort**: Sorts a 'timeSeries' object
- **start**: Returns start date/time of a 'timeSeries'
- **end**: Returns end date/time of a 'timeSeries'
- **t**: Returns the transpose of a 'timeSeries' object

Subsetting 'timeSeries' Objects

- `.subset_`: Subsets 'timeSeries' objects
- `.findIndex`: Index search in a 'timeSeries' object
- `[` Subsets a 'timeSeries' object
- `[<]`: Assigns values to a subset
- `$`: Subsets a 'timeSeries' by column names
- `$<-`: Replaces Subset by column names
- `t`: Returns the transpose of a 'timeSeries'
- `head`: Returns the head of a 'timeSeries'
- `tail`: Returns the tail of a 'timeSeries'
- `na.omit`: Handles NAs in a timeSeries object
- `removeNA`: Removes NAs from a matrix object
- `substituteNA`: Substitutes NAs by zero, column mean or median
- `interpNA`: Interpolates NAs using R's "approx" function

Mathematical Operation

- **Ops.timeSeries**: S4: Arith method for a 'timeSeries’ object
- **abs**: Returns absolute values of a 'timeSeries’ object
- **sqrt**: Returns square root of a 'timeSeries’ object
- **exp**: Returns the exponential values of a 'timeSeries’ object
- **log**: Returns the logarithm of a 'timeSeries’ object
- **sign**: Returns the signs of a 'timeSeries’ object
- **diff**: Differences a 'timeSeries’ object
timeSeries-package

scale Centers and/or scales a ‘timeSeries’ object
quantile Returns quantiles of an univariate ‘timeSeries’

Methods

as.timeSeries Defines method for a ‘timeSeries’
as.*.default Returns the input
as.*.ts Transforms a ‘ts’ object into a ‘timeSeries’
as.*.data.frame Transforms a ‘data.frame’ into a ‘timeSeries’
as.*.character Loads and transforms from a demo file
as.*.zoo Transforms a ‘zoo’ object into a ‘timeSeries’
as.vector.* Converts univariate timeSeries to vector
as.matrix.* Converts timeSeries to matrix
as.numeric.* Converts timeSeries to numeric
as.data.frame.* Converts timeSeries to data.frame
as.ts.* Converts timeSeries to ts
as.logical.* Converts timeSeries to logical
is.timeSeries Tests for a ‘timeSeries’ object
plot Displays a X-Y ‘timeSeries’ Plot
lines Adds connected line segments to a plot
points Adds Points to a plot
show Prints a ‘timeSeries’ object

Financial time series functions

align Aligns a ‘timeSeries’ to time stamps
cumulated Computes cumulated series from a returns
alignDailySeries Aligns a ‘timeSeries’ to calendarical dates
rollDailySeries Rolls a ‘timeSeries’ daily
drawdowns Computes series of drawdowns from financial returns
drawdownsStats Computes drawdowns statistics
durations Computes durations from a financial time series
countMonthlyRecords Counts monthly records in a ‘timeSeries’
rollMonthlyWindows Rolls Monthly windows
rollMonthlySeries Rolls a ‘timeSeries’ monthly
endOfPeriodSeries Returns end of periodical series
endOfPeriodStats Returns end of period statistics
endOfPeriodBenchmarks Returns period benchmarks
returns Computes returns from prices or indexes
returns@ Computes untrimmed returns from prices or indexes
runlengths Computes run lengths of a ‘timeSeries’
smooth Smooths a ‘timeSeries’
breaks Detects ‘timeSeries’ breaks by outlier detection
spreads  Computes spreads from a price/index stream
turns   Computes turning points in a 'timeSeries' object
turnstats Computes turning points statistics

Statistics Time Series functions

colCumsums Computes cumulated column sums of a 'timeSeries'
colCummaks Computes cumulated maximum of a 'timeSeries'
colCummins Computes cumulated minimum of a 'timeSeries'
colCumprods Computes cumulated product values by column
colCumreturns Computes cumulated returns by column
cols Computes sums of all values in each column
colMeans Computes means of all values in each column
colSds Computes standard deviations of all values in each column
colVars Computes variances of all values in each column
colSkewness Computes skewness of all values in each column
colKurtosis Computes kurtosis of all values in each column
colMaxs Computes maxima of all values in each column
colMins Computes minima of all values in each column
colProds Computes products of all values in each column
colStats Computes statistics of all values in each column
orderColnames Returns ordered column names of a 'timeSeries'
sortColnames Returns alphabetically sorted column names
sampleColnames Returns sampled column names of a 'timeSeries'
pcaColnames Returns PCA correlation ordered column names
hClustColnames Returns hierarchically clustered column names
statsColnames Returns statisticall rearrange column names
orderStatistics Computes order statistics of a 'timeSeries' object
rollMean Computes rolling means of a 'timeSeries' object
rollMin Computes rolling minima of a 'timeSeries' object
rollMax Computes rolling maxima of a 'timeSeries' object
rollMedian Computes rolling medians of a 'timeSeries' object
rollStats Computes rolling statistics of a 'timeSeries' object
rowCumsums Computes cumulated column sums of a 'timeSeries'
smoothLowess Smoothes a series with lowess function
smoothSupsmu Smoothes a series with supsmu function
smoothSpline Smoothes a series with smooth.spline function

Misc Functions

dummyDailySeries Creates a dummy daily 'timeSeries' object
isMonthly Decides if the series consists of monthly records
getArgs Extracts arguments from a S4 method
Description

Aggregates a 'timeSeries' Object.

Usage

```r
## S4 method for signature 'timeSeries'
aggregate(x, by, FUN, ...)

daily2monthly(x, init=FALSE)
daily2weekly(x, startOn="Tue", init=FALSE)
```

Arguments

- `x` an object of class 'timeSeries'.
- `by` a sequence of `timeDate` objects denoting the aggregation period.
- `FUN` the function to be applied.
- `startOn` a string value, specifying the day of week as a three letter abbreviation. Weekly aggregated data records are then fixed to the weekdays given by the argument `startOn`.
- `init` a logical value, if set to TRUE then the time series will be indexed to 1 for its first value. By default `init` is set to FALSE.
- `...` arguments passed to other methods.

Details

The function `aggregate` is a function which can aggregate time series on general aggregation periods.

In addition there are two tailored function for simple usage: Function `daily2monthly` and `daily2weekly` which allow to aggregate 'timeSeries' objects from daily to monthly or weekly levels, respectively.

In the case of the function `daily2weekly` one can explicitely the starting day of the week, the default value is Tuesday, `startOn="Tue"`.

Value

- `aggregate` returns an aggregated S4 object of class `timeSeries`.
- `daily2monthly` returns an aggregated monthly object of class `timeSeries`.
- `daily2weekly` returns an aggregated weekly object of class `timeSeries` starting on the specified day of week.
Examples

## Load Microsoft Data Set -
```r
x <- MSFT
```

## Aggregate by Weeks -
```
by <- timeSequence(from = start(x), to = end(x), by = "week")
aggregate(x, by, mean)
```

## Aggregate to Last Friday of Month -
```
by <- unique(timeLastNDayInMonth(time(x), 5))
X <- aggregate(x, by, mean)
X
dayOfWeek(time(X))
isMonthly(time(X))
```

## Aggregate to Last Day of Quarter -
```
by <- unique(timeLastNDayInQuarter(time(x)))
X <- aggregate(x, by, mean)
X
isQuarterly(x)
```

## Aggregate daily records to end of month records -
```
X <- daily2monthly(x)
X
isMonthly(x)
```

## Aggregate daily records to end of week records -
```
X <- daily2weekly(x, startOn="Fri")
X
dayOfWeek(time(X))
```

---

### Description

Aligns a 'timeSeries' Object.

### Usage

```
## S4 method for signature 'timeSeries'
align(x, by = "1d", offset = "0s", 
method = c("before", "after", "interp", "fillNA", 
"fmm", "periodic", "natural", "monoh.FC"), 
include.weekends = FALSE, ...)
```
Arguments

x an object of class `timeSeries`.
by a character string denoting the period
offset a character string denoting the offset
method a character string denoting the alignment approach.
include.weekends a logical flag, should weekend be included.
... Further arguments to be passed to the interpolating function.

Value

Returns an aligned S4 `timeSeries` object.

Examples

```r
## Use MSFT and Compute Sample Size -
dim(MSFT)

## Align the Series -
MSFT.AL <- align(MSFT)

## Show the Size of the Aligned Series -
dim(MSFT.AL)
```

Description

Applies a function to a `timeSeries` object over time periods of arbitrary positions and lengths.

Usage

```r
fapply(x, from, to, FUN, ...)
applySeries(x, from = NULL, to = NULL, by = c("monthly", "quarterly"),
FUN = colMeans, units = NULL, format = x@format, zone = x@FinCenter,
FinCenter = x@FinCenter, recordIDs = data.frame(), title = x@title,
documentation = x@documentation, ...)
```
Arguments

x an object of class timeSeries.

from, to starting date and end date as timeDate objects. Note, to must be time ordered after from. If from and to are missing in function fapply they are set by default to from=start(x) and to=end(x).

FUN the function to be applied. For the function applySeries the default setting is FUN=colMeans.

by a character value either "monthly" or "quarterly" used in the function applySeries. The default value is "monthly". Only operative when both arguments from and to have their default values NULL. In this case the function FUN will be applied to monthly or quarterly periods.

units an optional character string, which allows to overwrite the current column names of a timeSeries object. By default NULL which means that the column names are selected automatically.

format the format specification of the input character vector in POSIX notation.

zone the time zone or financial center where the data were recorded.

FinCenter a character value with the the location of the financial center named as "continent/city", or "city".

recordIDs a data frame which can be used for record identification information. Note, this is not yet handled by the apply functions, an empty data.frame will be returned.

title an optional title string, if not specified the inputs data name is deparsed.

documentation optional documentation string, or a vector of character strings.

... arguments passed to other methods.

Details

Like apply applies a function to the margins of an array, the function fapply applies a function to the time stamps or signal counts of a financial (therefore the "f" in front of the function name) time series of class 'timeSeries'.

The function fapply inputs a timeSeries object, and if from and to are missing, they take the start and end time stamps of the series as default values. The function then behaves like apply on the column margin.

Note, the function fapply can be used repetitive in the following sense: If from and to are two timeDate vectors of equal length then for each period spanned by the elements of the two vectors the function FUN will be applied to each period. The resulting time stamps, are the time stamps of the to vector. Note, the periods can be regular or irregular, and they can even overlap.

The function fapply calls the more general function applySeries which also offers, to create automatical monthly and quarterly periods.

Examples

## Percentual Returns of Swiss Bond Index and Performance Index -
LPP <- 100 * LPP200SREC[, c("SBI", "SPI")]
head(LPP, 20)
as

timeSeries Class, Coercion and Transformation

Description

Functions and methods dealing with the coercion of 'timeSeries' objects.

Usage

## Default S3 method:
as.timeSeries(x, ...)

## S3 method for class 'ts'
as.timeSeries(x, ...)

## S3 method for class 'data.frame'
as.timeSeries(x, ...)

## S3 method for class 'character'
as.timeSeries(x, ...)

## S3 method for class 'zoo'
as.timeSeries(x, ...)

## S4 method for signature 'timeSeries'
as.matrix(x, ...)

## S4 method for signature 'timeSeries'
as.ts(x, ...)

## S4 method for signature 'timeSeries'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'timeSeries'
as.ts(x, ...)
Arguments

optional A logical value. If TRUE, setting row names and converting column names (to
syntactic names) is optional.

row.names NULL or a character vector giving the row names for the data frame. Missing
values are not allowed.

x an object which is coerced according to the generic function.

... arguments passed to other methods.

Details

Functions to create 'timeSeries' objects from other objects:

- `as.timeSeries` Generic to convert an object to a 'timeSeries'.
- `as.timeSeries.default` Returns the unchanged object.
- `as.timeSeries.numeric` Converts from a numeric vector.
- `as.timeSeries.data.frame` Converts from a numeric vector.
- `as.timeSeries.matrix` Converts from a matrix.
- `as.timeSeries.ts` Converts from an object of class 'ts'.
- `as.timeSeries.character` Converts from a named demo file.
- `as.timeSeries.zoo` Converts an object of class zoo.

Functions to transform 'timeSeries' objects into other objects:

- `as.matrix.timeSeries` Coerces a 'timeSeries' to a matrix.
- `as.data.frame.timeSeries` Coerces a 'timeSeries' to a data.frame.
- `as.ts.timeSeries` S3: Coerces a 'timeSeries' to a 'ts' object.
- `as.ts.timeSeries` S3: Coerces a 'timeSeries' to a 'logical' object.

Value

Function `as.timeSeries` returns a S4 object of class 'timeSeries'.

Functions `as.numeric`, `as.data.frame`, `as.matrix`, `as.ts` return depending on the generic function a numeric vector, a data frame, a matrix, or an object of class ts.

Examples

```r
## Create an Artificial timeSeries Object -
setRmetricsOptions(myFinCenter = "GMT")
charvec <- timeCalendar()
data <- matrix(rnorm(12))
TS <- timeSeries(data, charvec, units = "RAND")
TS

## Coerce to Vector -
as.vector(TS)

## Coerce to Matrix -
```
attach

as.matrix(TS)

## Coerce to Data Frame -
  as.data.frame(TS)

### Description

Attaches a 'timeSeries' object to the search path.

### Usage

```r
## S4 method for signature 'timeSeries'
attach(what, pos = 2, name = deparse(substitute(what)),
   warn.conflicts = TRUE)
```

### Arguments

- **name**: alternative way to specify the database to be attached. See for details help(attach, package=base).
- **pos**: an integer specifying position in search() where to attach the database. See for details help(attach, package=base).
- **warn.conflicts**: a logical value. If TRUE, warnings are printed about conflicts from attaching the database, unless that database contains an object .conflicts.OK. A conflict is a function masking a function, or a non-function masking a non-function. See for details help(attach, package=base).
- **what**: database to be attached. This may currently be a timeSeries object, a data.frame or a list or a R data file created with save or NULL or an environment. See for details help(attach, package=base).

### Value

The environment is returned invisibly with a name attribute.

### Note

Note, the function detach from the base package can be used to detach the attached objects.

### Examples

```r
## Load Microsoft Data Set -
  x <- MSFT[1:10, ]
  colnames(x)

## Attach the Series and Compute the Range -
  attach(x)
```
range <- High - Low
range

## Convert Vector to a timeSeries Object -
timeSeries(data=range, charvec=time(x), units="Range")

## Detach the series from the search path -
detach("x")
anS <- try(High, silent=TRUE)
cat(ans[1])

---

### attributes

**Get and Set Optional Attributes of a 'timeSeries'**

**Description**

Extracts or assigns optional attributes from or to a `timeSeries` object.

**Usage**

```r
getAttributes(obj)
setAttributes(obj) <- value
```

**Arguments**

- `obj` a `timeSeries` object whose optional attributes are to be accessed.
- `value` an object, the new value of the attribute, or `NULL` to remove the attribute.

**Details**

Each `timeSeries` object is documented. By default a time series object holds in the documentation slot a string with creation time and the user who has defined it. But this is not all. Optionally the whole creation process and history can be recorded. For this the `@documentation` slot may have an optional "Attributes" element. This attribute is tracked over the whole life time of the object whenever the time series is changed. Whenever you like to be informed about the optional attributes, or you like to recover them you can dot it, and evenmore, whenever you like to add information as an additional attribute you can also do it.

The two functions `getAttributes` and `setAttributes` provide access to and allow to modify the optional attributes of a `timeSeries` object.

**Examples**

```r
## Create an artificial timeSeries Object -
tS <- dummySeries()
tS

## Get Optional Attributes -
getAttributes(tS)
```
base-methods

Methods for 'timeSeries' object

Description

Methods for function in Package `base` for `timeSeries` object.

Methods

x = "timeSeries" a `timeSeries` object.

Examples

## None -

bind

Bind two `timeSeries` objects

Description

Binds two `timeSeries` objects either by column or by row.

Value

returns a S4 object of class `timeDate`.

Examples

## Load Microsoft Data Set -

x <- MSFT[1:12, ]

x

## Bind Columnwise -

X <- cbind(x[, "Open"], returns(x[, "Open"]))

colnames(X) <- c("Open", "Return")

X

## Bind Rowwise -

Y <- rbind(x[1:3, "Open"], x[10:12, "Open"])

Y
**colCum**  
*Cumulated Column Statistics*

**Description**

Functions to compute cumulative column statistics.

**Usage**

```r
## S4 method for signature 'timeSeries'
colCumsums(x, na.rm = FALSE, ...)

## S4 method for signature 'timeSeries'
colCummaxs(x, na.rm = FALSE, ...)

## S4 method for signature 'timeSeries'
colCummins(x, na.rm = FALSE, ...)

## S4 method for signature 'timeSeries'
colCumprods(x, na.rm = FALSE, ...)

## S4 method for signature 'timeSeries'
colCumreturns(x, method = c("geometric", "simple"), na.rm = FALSE, ...)
```

**Arguments**

- `method`  
  a character string to indicate if geometric (TRUE) or simple (FALSE) returns should be computed.

- `na.rm`  
  a logical. Should missing values be removed?

- `x`  
  a time series, may be an object of class "matrix", or "timeSeries".

- `...`  
  arguments to be passed.

**Value**

all functions return an S4 object of class `timeSeries`.

**Examples**

```r
## Simulated Return Data -
x = matrix(rnorm(24), ncol = 2)

## Cumulative Sums Column by Column -
colCumsums(x)
```
Description

A collection and description of functions to compute column statistical properties of financial and economic time series data.

The functions are:

- `colStats` calculates column statistics,
- `colSums` calculates column sums,
- `colMeans` calculates column means,
- `colSds` calculates column standard deviations,
- `colVars` calculates column variances,
- `colSkewness` calculates column skewness,
- `colKurtosis` calculates column kurtosis,
- `colMaxs` calculates maximum values in each column,
- `colMins` calculates minimum values in each column,
- `colProds` computes product of all values in each column,
- `colQuantiles` computes quantiles of each column.

Usage

```
colStats(x, FUN, ...)  
colSds(x, ...)  
colVars(x, ...)  
colSkewness(x, ...)  
colKurtosis(x, ...)  
colMaxs(x, ...)  
colMins(x, ...)  
colProds(x, ...)  
colQuantiles(x, prob = 0.05, ...)  
colStdevs(x, ...)  
colAvgs(x, ...)  
```

Arguments

- `FUN` a function name. The statistical function to be applied.
- `prob` a numeric value, the probability with value in [0,1].
comment

x a rectangular object which can be transformed into a matrix by the function as.matrix.

... arguments to be passed.

Value

the functions return a numeric vector of the statistics.

See Also

link{rowStats}.

Examples

```r
## Simulated Return Data in Matrix Form -
x = matrix(rnorm(252), ncol = 2)

## Mean Columnwise Statistics -
colStats(x, FUN = mean)

## Quantiles Column by Column -
colQuantiles(x, prob = 0.10, type = 1)
```

comment

comment for timeSeries objects

Description

Print or assign new comment to a timeSeries object.

Usage

```r
## S4 method for signature 'timeSeries'
comment(x)
## S4 replacement method for signature 'timeSeries'
comment(x) <- value
```

Arguments

- **x**: a timeSeries object.
- **value**: a character string - the comment.
cumulated

Examples

```r
## Get Description from timeSeries -
comment(LPP2005REC)

## Add User to comment -
comment(LPP2005REC) <- paste(comment(LPP2005REC), "by User Rmetrics")
comment(LPP2005REC)
```

---

cumulated | *Cumulated Time Series from Returns*

Description

Computes a cumulated financial 'timeSeries', e.g. prices or indexes, from financial returns.

Usage

cumulated(x, ...)  

## Default S3 method:
cumulated(x, method = c("continuous", "discrete",  
"compound", "simple"), percentage = FALSE, ...)

Arguments

- **method**: a character string naming the method how the returns were computed.
- **percentage**: a logical value. By default FALSE, if TRUE the series will be expressed in percentage changes.
- **x**: an object of class `timeseries`.
- **...**: arguments to be passed.

Details

Note, the function `cumulated` assumes as input discrete returns from a price or index series. Only then the cumulatrd series agrees with the original price or index series. The first values of the cumulated series cannot be computed, it is assumed that the series is indexed to 1.

Value

Returns a 'timeSeries' object of the same class as the input argument `x`. 
Examples

```r
## Use the Microsofts' Close Prices Indexed to 1 -
MSFT.CL <- MSFT[, "Close"]
MSFT.CL <- MSFT.CL/MSFT[[1, "Close"]]
head(MSFT.CL)

## Compute Discrete Return -
MSFT.RET <- returns(MSFT.CL, method = "discrete")

## Cumulated Series and Compare -
MSFT.CUM <- cumulated(MSFT.RET, method = "discrete")
head(cbind(MSFT.CL, MSFT.CUM))
```

Description

Utilities called to implement object@.Data of timeSeries objects.

Examples

```r
## Load Microsoft Data -
X <- MSFT[1:10, 1:4]

## Get Data Part -
DATA <- getDataPart(X)
class(DATA)
```

Description

Creates Date and User Information

Usage

description()

Examples

```r
## Show Default Description String -
description()
```
Description

Differences a 'timeSeries' object.

Usage

diff(x, ...)

Arguments

x an object of class 'timeSeries'.
... further arguments to be passed. These may include

Details

Arguments to be passed may include:

  lag - an integer indicating which lag to use. By default 1.
  diff - an integer indicating the order of the difference. By default 1.
  trim - a logical flag. Should NAs at the beginning of the series be removed? By default FALSE.
  pad - a numeric value with which NAs should be replaced at the beginning of the series. By default NA.

Value

Returns a differenced S4 'timeSeries' object.

Examples

```r
## Load Microsoft Data Set -
x <- MSFT[1:12, ]
x

## Compute Differences -
diff(x)

## Trimmed Differences -
diff(x, trim=TRUE)

## Padded Differences -
diff(x, trim=FALSE, pad=0)
```
**dimnames**

*Time Series Columns and Rows*

**Description**

Handling columns and rows of `timeSeries` objects.

**Details**

- `dim`  Returns the dimension of a `timeSeries` object
- `dimnames`  Returns the dimension names of a `timeSeries` object
- `colnames<-`  Assigns column names to a `timeSeries` object
- `rownames<-`  Assigns row names to a `timeSeries` object

**Value**

Returns the dimensions and related numbers of a `timeSeries` object.

**Examples**

```r
## Load Swiss Pension Fund Benchmark Data -
X <- LPP2005REC[1:10, 1:3]

## Get Dimension -
dim(X)

## Get Column and Row Names -
dimnames(X)

## Get Column / Row Names -
colnames(X)
rownames(X)

## Try your own DIM -
DIM <- function(x) {c(NROW(x), NCOL(x))}
DIM(X)
DIM(X[, 1])

## Try length / LENGTH -
length(X)
length(X[, 1])
LENGTH <- function(X) NROW(X)
LENGTH(X)

## Columns / Rows -
ncol(X); NCOL(X)
```
drawdowns

**Calculations of Drawdowns**

**Description**

Compute series of drawdowns from financial returns and calculate drawdown statistics.

**Usage**

```r
drawdowns(x, ...)
drawdownsStats(x, ...)
```

**Arguments**

- `x` a `timeSeries` object of financial returns. Note, drawdowns can be calculated from an uni- or multivariate time series object, statistics can only be computed from an univariate time series object.
- `...` optional arguments passed to the function `na.omit`.

**Details**

The code in the core of the function `drawdownsStats` was borrowed from the package `PerformanceAnalytics` authored by Peter Carl and Sankalp Upadhyay.

**Value**

- `drawdowns` returns an object of class 'timeSeries'.
- `drawdownsStats` returns an object of class 'data.frame' with the following entries:
  - "drawdown" - the depth of the drawdown,
  - "from" - the start date,
  - "trough" - the trough period,
  - "to" - the end date,
  - "length" - the length in number of records,
  - "peaktrough" - the peak trough, and,
  - "recovery" - the recovery length in number of records.
durations

Author(s)

Peter Carl and Sankalp Upadhyay for code from the contributed R package PerformanceAnalytics used in the function drawdownsStats.

Examples

```r
## Use Swiss Pension Fund Data Set of Returns -
head(LPP2005REC)
SPI <- LPP2005REC[, "SPI"]
head(SPI)

## Plot Drawdowns -
dd = drawdowns(LPP2005REC[, "SPI"], main = "Drawdowns")
plot(dd)

## Compute Drawdowns Statistics -
ddStats <- drawdownsStats(SPI)
class(ddStats)

## Note, Only Univariate Series are allowed -
ddStats <- try(drawdownsStats(LPP2005REC))
class(ddStats)
```

durations

Durations from a Time Series

Description

Computes durations from an object of class 'timeSeries'.

Usage

```r
durations(x, trim = FALSE, units = c("secs", "mins", "hours", "days"))
```

Arguments

- **x**: an object of class `timeSeries`.
- **trim**: a logical value. By default TRUE, the first missing observation in the return series will be removed.
- **units**: a character value or vector which allows to set the units in which the durations are measured. By default durations are measured in seconds.
Details

Durations measure how long it takes until we get the next record in a timesSeries object. We return a time series in which for each time stamp we get the length of the period from when we got the last record. This period is measured in length specified by the argument units, for daily data use units="days".

Value

returns an object of class timeSeries.

Examples

```r
## Compute Durations in days for the MSFT Series -
head(durations(MSFT, units = "days"))
head(durations(MSFT, trim = TRUE, units = "days"))

## The same in hours -
head(durations(MSFT, trim = TRUE, units = "hours"))
```

filter

Linear Filtering on a Time Series

Description

Applies linear filtering to a univariate 'timeSeries'.

Value

A 'timeSeries' object without missing values.

Examples

```r
## Create a Dummy Signal 'timeSeries' -
data <- matrix(rnorm(100), ncol = 2)
s <- timeSeries(data, units=c("A", "B"))
head(s)

## Filter the series -
f <- filter(s, rep(1, 3))
head(f)

## Plot and Compare the first series -
plot(cbind(s[, 1], f[, 1]), plot.type="s")
```
### Description

Print or assign new financial center to a 'timeSeries' object.

### Usage

```r
getFinCenter(x)
ssetFinCenter(x) <- value
```

```r
## S4 method for signature 'timeSeries'
finCenter(x)
## S4 replacement method for signature 'timeSeries'
finCenter(x) <- value
```

### Arguments

- **x**: a 'timeSeries' object.
- **value**: a character with the the location of the financial center named as "continent/city".

### See Also

- `listFinCenter`

### Examples

```r
## An artificial timeSeries Object -
tS <- dummySeries()
tS

## Print Financial Center -
finCenter(tS)
getFinCenter(tS)

## Assign New Financial Center -
finCenter(tS) <- "Zurich"
tS
setFinCenter(tS) <- "New_York"
tS
```
is.timeSeries

Description

is.timeSeries tests if its argument is a timeSeries. is.timeSeries tests if series has no timestamps.

Usage

is.timeSeries(x)
is.signalSeries(x)

Arguments

x an object of class 'timeSeries'.

Value

Returns TRUE or FALSE depending on whether its argument is an object of class 'timeSeries' or not.

Examples

## Create an Artificial timeSeries Object -
setRmetricsOptions(myFinCenter = "GMT")
charvec <- timeCalendar()
data <- matrix(rnorm(12))
TS <- timeSeries(data, charvec, units = "RAND")
TS

## Test for timeSeries -
is.timeSeries(TS)

isRegular

Checks if a time series is regular

Description

Checks if a time series is regular.
Usage

```r
## S4 method for signature 'timeSeries'
isDaily(x)
isMonthly(x)
isQuarterly(x)
isRegular(x)

## S4 method for signature 'timeSeries'
frequency(x, ...)
```

Arguments

- `x`: an R object of class `timeSeries`.
- `...`: arguments to be passed.

Details

What is a regular time series? If a series is daily, monthly, or weekly then we speak of a regular series. This can be tested calling the functions `isDaily`, `isMonthly`, `isQuarterly`, or in general `isRegular`. If the series is regular then the frequency of the series can be determined calling the function `frequency`.

A time series is defined as daily if the series has not more than one date/time stamp per day.
A time series is defined as monthly if the series has not more than one date/time stamp per month.
A time series is defined as quarterly if the series has not more than one date/time stamp per month.
Note, a monthly series is also a daily series, a quarterly series is also a monthly series.
With these definitions a regular series is either a monthly or a quarterly series.

NOT yet implemented is the case of weekly series.

Value

The `is*` functions return `TRUE` or `FALSE` depending on whether the series fulfills the condition or not.

The function `frequency` returns in general 1, for quarterly series 4, and for monthly series 12.

Examples

```r
## None
```
isUnivariate  Checks if a Time Series is Univariate

Description
Checks if a time series object or any other rectangular object is univariate or multivariate.

Usage
isUnivariate(x)
isMultivariate(x)

Arguments
x an object of class timeSeries or any other rectangular object.

Details
A rectangular object x is considered to be univariate if the function NCOL(x) returns one, and is considered to be multivariate if NCOL(x) returns a value bigger than one.

Value
isUnivariate
isMultivariate

return a logical depending if the test is true or not.

Examples
## Load Microsoft Data -
setRmetricsOptions(myFinCenter = "GMT")
data(MSFT)
Open = MSFT[, "Open"]

## Is the timeSeries Univariate -
isUnivariate(MSFT)
isUnivariate(Open)

## Is the timeSeries Multivariate -
isMultivariate(MSFT)
isMultivariate(Open)
Description

Compute a lagged version of a 'timeSeries' object.

Usage

## S4 method for signature 'timeSeries'

lag(x, k = 1, trim = FALSE, units = NULL, ...)

Arguments

- **k**: an integer value. The number of lags (in units of observations). By default 1.
- **trim**: a logical value. By default TRUE, the first missing observation in the return series will be removed.
- **units**: an optional character string, which allows to overwrite the current column names of a timeseries object. By default NULL which means that the column names are selected automatically.
- **x**: an object of class timeseries.
- **...**: arguments passed to other methods.

Value

returns a lagged S4 object of class 'timeSeries'.

Examples

```r
## Load Microsoft Data Set
x = MSFT[1:20, "Open"]

## Lag the timeSeries Object:
lag(x, k = -1:1)
```

Description

Functions and methods dealing with mathematical 'timeSeries' operations.
Usage

```r
## S4 method for signature 'timeSeries'
quantile(x, ...)
```

Arguments

- `x`: an object of class `timeSeries`.
- `...`: arguments to be passed.

Details

The math functions include:

- `Ops-method`: Group 'Ops' methods for a 'timeSeries' object
- `Math-method`: Group 'Math' methods for a 'timeSeries' object
- `Math2-method`: Group 'Math2' methods for a 'timeSeries' object
- `Summary-method`: Group 'Summary' methods for a 'timeSeries' object
- `quantile`: Returns quantiles of an univariate 'timeSeries'.

Value

Returns the value from a mathematical or logical operation operating on objects of class 'timeSeries[]', or the value computed by a mathematical function.

Examples

```r
## Create an Artificial timeSeries Object
setRmetricsOptions(myFinCenter = "GMT")
charvec = timeCalendar()
set.seed(4711)
data = matrix(exp(cumsum(rnorm(12, sd = 0.1))))
TS = timeSeries(data, charvec, units = "TS")
TS

## Mathematical Operations: +/- * ^ ...
TS*2
TS[2:4]
OR = returns(TS)
OR
OR > 0
```
merge

Merges two 'timeSeries' objects

Description
Merges several object types with 'timeSeries' objects. The number of rows must match.

Details
The following combinations are supported:

<table>
<thead>
<tr>
<th>Object Type 1</th>
<th>Object Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeseries</td>
<td>ANY</td>
</tr>
<tr>
<td>timeseries</td>
<td>missing</td>
</tr>
<tr>
<td>timeseries</td>
<td>numeric</td>
</tr>
<tr>
<td>timeseries</td>
<td>matrix</td>
</tr>
<tr>
<td>timeseries</td>
<td>timeSeries</td>
</tr>
</tbody>
</table>

Value
Returns a 'timeSeries' object of two merged time series.

Examples
```r
## Load Series -
  x <- MSFT[1:12, ]

## Merge 'timeSeries' with missing Object -
  merge(x)

## Merge 'timeSeries' with numeric Object -
  y <- rnorm(12)
  class(y)
  merge(x, y)

## Merge 'timeSeries' with matrix Object -
  y <- matrix(rnorm(24), ncol=2)
  class(y)
  merge(x, y)

## Merge 'timeSeries' with matrix Object -
  y <- timeSeries(data=rnorm(12), charvec=time(x))
  class(y)
  merge(x, y)
```
monthly

Description

Allow to work with model frames for 'timeSeries' objects.

Details

The function model.frame is a generic function which returns in the R-ststs framework by default a data.frame with the variables needed to use formula and any ... arguments. In contrast to this the method returns an object of class timeSeries when the argument data was not a data.frame but also an object of class 'timeSeries'.

Value

Returns an object of class 'timeSeries'.

Note

This function is preliminary and untested.

See Also

model.frame.

Examples

## Load Microsoft Data -
setRmetricsOptions(myFinCenter = "GMT")
X <- MSFT[1:12, ]

## Extract High's and Low's:
DATA <- model.frame(~ High + Low, data = X)
class(DATA)
as.timeSeries(DATA)

## Extract Open Prices and their log10's:
base <- 10
Open <- model.frame(Open ~ log(Open, base = 'base'), data = X)
colnames(Open) <- c("X", "log10(X)")
class(Open)
as.timeSeries(Open)

---

monthly

Special Monthly Series

Description

Functions and methods dealing with special monthly 'timeSeries' objects.
Usage

countMonthlyRecords(x)
rollMonthlyWindows(x, period = "12m", by = "1m")
rollMonthlySeries(x, period = "12m", by = "1m", FUN, ...)

Arguments

x a 'timeSeries' object.
period a character string specifying the rolling period composed by the length of the period and its unit. As examples: "3m" represents quarterly shifts, and "6m", jcode"12m", and "24m" semi-annual, annual and bi-annual shifts. To determine the proper start of the series is in the responsibility of the user.
by a character string specifying the rolling shift composed by the length of the shift and its unit. As examples: "1m" represents monthly shifts, "3m" represents quarterly shifts, and "6m" semi-annual shifts. To determine the proper start of the series is in the responsibility of the user.
FUN the function for the statistic to be applied. For example in the case of aggregation use colAvg.
... arguments passed to the function FUN.

Details

The function countMonthlyRecords computes a 'timeSeries' that holds the number of monthly counts of the records.
The function rollMonthlyWindows computes start and end dates for rolling time windows.
The function rollMonthlySeries computes a static over rolling periods defined by the function rollMonthlyWindows.

Value

The function countMonthlyRecords returns a 'timeSeries' object.
The function rollMonthlyWindows returns a list with two named 'timeDate' entries: $from and to. An attribute "control" is added which keeps the start and end dates of the series.
The function rollMonthlySeries computes the statistics defined by the function FUN over a rolling window internally computed by the function rollMonthlyWindows. Note, the periods may be overlapping, may be dense, or even may have gaps.

Examples

## Load Microsoft Daily Data Set:
```r
x <- MSFT
```

## Count Monthly Records -
```r
counts <- countMonthlyRecords(x)
counts
```
### Handling Missing Time Series Values

**Description**

Functions for handling missing values in 'timeSeries' objects

**Usage**

```r
## S4 method for signature 'timeSeries'
na.omit(object, method = c("r", "s", "z", "ir", "iz", "ie"),
       interp = c("before", "linear", "after"), ...)

removeNA(x, ...)
substituteNA(x, type = c("zeros", "mean", "median"), ...)
interpNA(x, method = c("linear", "before", "after"), ...)
```

**Arguments**

- `interp, type`  
  Three alternative methods are provided to remove NAs from the data: type="zeros" replaces the missing values by zeros, type="mean" replaces the missing values by the column mean, type="median" replaces the missing values by the column median.

- `method`  
  Specifies the method how to handle NAs. One of the applied vector strings:
  method="s" na.rm = FALSE, skip, i.e. do nothing, method="r" remove NAs, method="z" substitute NAs by zeros, method="ir" interpolate NAs and remove NAs at the beginning and end of the series, method="iz" interpolate NAs and substitute NAs at the beginning and end of the series, method="ie" interpolate NAs and extrapolate NAs at the beginning and end of the series. [interpNA] - Specifies the method how to interpolate the matrix column by column. One of the applied vector strings: method="linear", method="before" or method="after". For the interpolation the function approx is used.

- `object`  
  an object of class("timeSeries").
a numeric matrix, or any other object which can be transformed into a matrix through \( x = \text{as.matrix}(x, \ldots) \). If \( x \) is a vector, it will be transformed into a one-dimensional matrix.

Arguments to be passed to the function \text{as.matrix}. 

Details

Functions for handling missing values in 'timeSeries' objects and in objects which can be transformed into a vector or a two-dimensional matrix.

The functions are listed by topic.

\begin{itemize}
  \item \text{na.omit} \begin{itemize}
      \item Handles NAs.
  \end{itemize}
  \item \text{removeNA} \begin{itemize}
      \item Removes NAs from a matrix object.
  \end{itemize}
  \item \text{substituteNA} \begin{itemize}
      \item Substitute NAs by zero, the column mean or median.
  \end{itemize}
  \item \text{interpNA} \begin{itemize}
      \item interpolates NAs using R's "approx" function.
  \end{itemize}
\end{itemize}

\textbf{Missing Values in Price and Index Series:}

Applied to \text{timeSeries} objects the function \text{removeNA} just removes rows with NAs from the series. For an interpolation of time series points one can use the function \text{interpNA}. Three different methods of interpolation are offered: "linear" does a linear interpolation, "before" uses the previous value, and "after" uses the following value. Note, that the interpolation is done on the index scale and not on the time scale.

\textbf{Missing Values in Return Series:}

For return series the function \text{substituteNA} may be useful. The function allows to fill missing values either by method="zeros", the method="mean" or the method="median" value of the appropriate columns.

\textbf{Note}

The functions \text{removeNA}, \text{substituteNA} and \text{interpNA} are older implementations. Please use in all cases if possible the new function \text{na.omit}.

When dealing with daily data sets, there exists another function \text{alignDaily} Series which can handle missing data in un-aligned calendarical 'timeSeries' objects.

\textbf{References}


\textbf{Examples}

\begin{verbatim}
## Create a Matrix -
X <- matrix(rnorm(100), ncol = 5)

## Replace a Single NA Inside -
X[3, 5] <- NA
\end{verbatim}
## na.contiguous

Find Longest Contiguous Stretch of non-NAs

### Description
Find the longest consecutive stretch of non-missing values in a timeSeries object. (In the event of a tie, the first such stretch.)

### Usage
```r
## S4 method for signature 'timeSeries'
na.contiguous(object, ...)  
```

### Arguments
- `object`: a timeSeries object.
- `...`: further arguments passed to or from other methods.

### Value
A timeSeries object without missing values.
Examples

```r
## Dummy timeSeries with NAs entries
data <- matrix(sample(c(1:20, rep(NA,4))), ncol = 2)
s <- timeSeries(data, timeCalendar())

## Find the longest consecutive non-missing values
na.contiguous(s)
```

---

**orderColnames**

*Reorder Column Names of a Time Series*

**Description**

Functions and methods dealing with the rearrangement of column names of `timeSeries` objects.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderColnames</td>
<td>Returns ordered column names of a time Series</td>
</tr>
<tr>
<td>sortColnames</td>
<td>Returns sorted column names of a time Series</td>
</tr>
<tr>
<td>sampleColnames</td>
<td>Returns sampled column names of a time Series</td>
</tr>
<tr>
<td>statsColnames</td>
<td>Returns statistically rearranged column names</td>
</tr>
<tr>
<td>pcaColnames</td>
<td>Returns PCA correlation ordered column names</td>
</tr>
<tr>
<td>hclustColnames</td>
<td>Returns hierarchical clustered column names</td>
</tr>
</tbody>
</table>

**Usage**

```r
orderColnames(x, ...)  
sortColnames(x, ...)   
sampleColnames(x, ...)  
statsColnames(x, FUN = colMeans, ...)  
pcaColnames(x, robust = FALSE, ...)  
hclustColnames(x, method = c("euclidean", "complete"), ...)
```

**Arguments**

- **FUN**
  - a character string indicating which statistical function should be applied. By default statistical ordering operates on the column means of the time series.

- **method**
  - a character string with two elements. The first determines the choice of the distance measure, see `dist`, and the second determines the choice of the agglomeration method, see `hclust`.

- **robust**
  - a logical flag which indicates if robust correlations should be used.

- **x**
  - an object of class `timesSeries` or any other rectangular object which can be transformed by the function `as.matrix` into a numeric matrix.

- **...**
  - further arguments to be passed, see details.
orderStatistics

Details

Statistically Motivated Rearrangement

The function `statscolnames` rearranges the column names according to a statistical measure. These measure must operate on the columns of the time series and return a vector of values which can be sorted. Typical functions are those listed in `help` page `colStats` but one can also create his own functions which compute for example risk or any other statistical measure. The ... argument allows to pass additional arguments to the underlying function `FUN`.

PCA Ordering of the Correlation Matrix

The function `pcacolnames` rearranges the column names according to the PCA ordered correlation matrix. The argument `robust` allows to select between the use of the standard `cor` and computation of robust correlations using the function `covMcd` from contributed R package `robustbase`. The ... argument allows to pass additional arguments to the two underlying functions `cor` or `covMcd`. E.g. adding `method="kendall"` to the argument list calculates Kendall’s rank correlations instead the default which calculates Person’s correlations.

Ordering by Hierarchical Clustering

The function `pcacolnames` uses the hierarchical clustering approach `hclust` to rearrange the column names of the time series.

Value

returns a vector of character string, the rearranged column names.

Examples

```r
## Load Swiss Pension Fund Benchmark Data -
data <- LPP2005REC[,1:6]

## Abbreviate Column Names -
colnames(data)

## Sort Alphabetically -
sortColnames(data)

## Sort by Column Names by Hierarchical Clustering -
hclustColnames(data)
head(data[, hclustColnames(data)])
```

Description

Computes order statistic of a timeSeries.
Usage

orderStatistics(x)

Arguments

x an univariate 'timeSeries' object.

Value

Function orderStatistics returns the order statistic of an univariate 'timeSeries' object. The output is an object of class 'list'.

Examples

## Load Swiss Pension Fund Benchmark Data -
setRmetricsOptions(myFinCenter = "GMT")
X <- LPP2005REC[, "SPI"]
colnames(X)

## Compute 1% Order Statistics -
N <- round(0.01*nrow(X))
O5 <- orderStatistics(X)[[1]]
O5[1:N]

Description

Computes periodical statistics back to a given period.

Usage

endOfPeriodSeries(x,
nYearsBack = c("1y", "2y", "3y", "5y", "10y", "YTD"))

eendOfPeriodStats(x,
nYearsBack = c("1y", "2y", "3y", "5y", "10y", "YTD"))

eendOfPeriodBenchmarks(x, benchmark = ncol(x),
nYearsBack = c("1y", "2y", "3y", "5y", "10y", "YTD"))
plot-methods

Arguments

- **x**: an end-of-month recorded multivariate 'timeSeries' object. One of the columns holds the benchmark series specified by the argument `benchmark`. By default, this is the last column of `x`.
- **nYearsBack**: a period string. How long back should the series be treated? Options include values from 1 year to 10 years, and year-to-date: "1y", "2y", "3y", "5y", "10y", "YTD".
- **benchmark**: an integer giving the position of the benchmark series in `x`.

Details

The function `endOfPeriodSeries` returns series back to a given period.
The function `endOfPeriodStats` returns statistics back to a given period.
The function `endOfPeriodBenchmarks` returns benchmarks back to a given period.

`x` must be end of month data. Note you can create such series using for example the functions: `align`, `alignDailySeries`, `daily2monthly`.

Examples

```r
# Load Series: Column 1:3 Swiss Market, Column 8 (4) Benchmark
x <- 100 * LPP2005REC[, c(1:3, 8)]
colnames(x)
x <- daily2monthly(x)
x

# Get the Monthly Series -
endOfPeriodSeries(x, nYearsBack="1y")

# Compute the Monthly Statistics -
endOfPeriodStats(x, nYearsBack="1y")

# Compute the Benchmark -
endOfPeriodBenchmarks(x, benchmark=4)
```

plot-methods

Plot a Time Series

Description

Plots 'timeSeries' objects and add lines and points.

Usage

```r
## S4 method for signature 'timeSeries'
plot(x, y, FinCenter = NULL,
     plot.type = c("multiple", "single"), format = "auto",
     at = pretty(x), widths = 1, heights = 1, xy.labels,
```
```
xy.lines, panel = lines, nc, yax.flip = FALSE,
mar.multi = c(0, 5.1, 0, if (yax.flip) 5.1 else 2.1),
oma.multi = c(6, 0, 5, 0), axes = TRUE, ...)

## S4 method for signature 'timeSeries'
lines(x, FinCenter = NULL, ...)
## S4 method for signature 'timeSeries'
points(x, FinCenter = NULL, ...)

## S3 method for class 'timeSeries'
pretty(x, n=5, min.n=n%%3, shrink.sml=0.75,
       high.u.bias=1.5, u5.bias=0.5+1.5*high.u.bias, eps.correct=0, ...)

Arguments

x, y          objects of class timeSeries.
FinCenter     a character with the the location of the financial center named as "continent/city".
plot.type     for multivariate time series, should the series by plotted separately (with a common time axis) or on a single plot?
format        POSIX label format, e.g. "%Y-%m-%d" or "%F" for ISO-8601 standard date format.
at            a timeDate object setting the plot label positions. If at=pretty(x), the positions are generated automatized calling the function pretty. Default option at="auto" selects 6 equal spaced time label positions. For the new plot themes set at="pretty" or at="chic". In this case additional arguments can be passed through the ... arguments, see details.
widths, heights widths and heights for individual graphs, see layout.
xy.labels     logical, indicating if text() labels should be used for an x-y plot, \_or\_ character, supplying a vector of labels to be used. The default is to label for up to 150 points, and not for more.
xy.lines      logical, indicating if lines should be drawn for an x-y plot. Defaults to the value of xy.labels if that is logical, otherwise to TRUE
panel         a function(x, col, bg, pch, type, ...) which gives the action to be carried out in each panel of the display for plot.type="multiple". The default is lines.
nc             the number of columns to use when type="multiple". Defaults to 1 for up to 4 series, otherwise to 2.
yax.flip      logical indicating if the y-axis (ticks and numbering) should flip from side 2 (left) to 4 (right) from series to series when type="multiple".
mar.multi, oma.multi
              the (default) par settings for plot.type="multiple".
axes          logical indicating if x- and y- axes should be drawn.
n             an integer giving the desired number of intervals.
min.n         a nonnegative integer giving the minimal number of intervals.
```
shrink.sml  a positive numeric by which a default scale is shrunk in the case when range(x) is very small.

high.u.bias  a non-negative numeric, typically > 1. Larger high.u.bias values favor larger units.

u5.bias  a non-negative numeric multiplier favoring factor 5 over 2.

eps.correct  an integer code, one of 0,1,2. If non-0, a correction is made at the boundaries.

...  additional graphical arguments, see plot, plot.default and par.

Details

The original plotting function plot was build along R’s plotting function plot.ts with an additional argument to tailor the position marks at user defined position specified by the argument at. We call this style or theme "ts".

With version R 3.1 we have introduced two new additional plotting themes called "pretty" and "chic". They are becoming active when we set at="pretty" or at="chic".

Plot style or theme "pretty" is an extension of our original plotting function.

Plot style or theme "chic" an implementation along the contributed packages xts and PerformanceAnalytics from the Chicago finance group members. "Chicago" gave the name to call the theme "chic".

For both themes, "pretty" and "chic" additional arguments are passed through the ... arguments. These are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;l&quot;</td>
<td>types pf plot</td>
</tr>
<tr>
<td>col</td>
<td>1</td>
<td>colors for lines and points</td>
</tr>
<tr>
<td>pch</td>
<td>20</td>
<td>plot symbol</td>
</tr>
<tr>
<td>cex</td>
<td>1</td>
<td>character and symbol scales</td>
</tr>
<tr>
<td>cex</td>
<td>1</td>
<td>scale of axes</td>
</tr>
<tr>
<td>lty</td>
<td>1</td>
<td>line types</td>
</tr>
<tr>
<td>lwd</td>
<td>2</td>
<td>line widths</td>
</tr>
<tr>
<td>cex.axes</td>
<td>1</td>
<td>scale of axes</td>
</tr>
<tr>
<td>cex.lab</td>
<td>1</td>
<td>scale of labels</td>
</tr>
<tr>
<td>cex.pch</td>
<td>1</td>
<td>scale of plot symbols</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid</td>
<td>TRUE</td>
<td>should grid lines plotted?</td>
</tr>
<tr>
<td>frame.plot</td>
<td>TRUE</td>
<td>should b box around the plot?</td>
</tr>
<tr>
<td>axes</td>
<td>TRUE</td>
<td>should be axes drawn on the plot?</td>
</tr>
<tr>
<td>ann</td>
<td>TRUE</td>
<td>should default annotations appear?</td>
</tr>
</tbody>
</table>

Concerning the plot elements, the length of these vectors has to be the same as the number of columns in the time series to be plotted. If their length is only one, then they are repeated.

There is an almost 70 pages vignette added to the package, with dozens of examples of tailored plots. Have a look in it.

Value

Displays a plot or plot elements of an object of class 'timeSeries'.

...
Examples

```r
## Load Swiss Pension Fund Benchmark Data -
LPP <- LPP2005REC[1:12, 1:4]
colnames(LPP) <- abbreviate(colnames(LPP), 2)
finCenter(LPP) <- "GMT"

## Example Plot 1 -
plot(LPP[, 1], type = "o", col = "steelblue",
    main = "LPP", xlab = "2005", ylab = "Return")
plot(LPP[, 1], at="auto", type = "o", col = "steelblue",
    main = "LPP", xlab = "2005", ylab = "Return")

## Example Plot 2 -
plot(LPP[, 1:2], type = "o", col = "steelblue",
    main = "LPP", xlab = "2005", ylab = "Return")

## Example Plot 3 -
plot(LPP[, 1], LPP[, 2], type = "p", col = "steelblue",
    main = "LPP", xlab = "Return 1", ylab = "Return 2")

## Example Plot 4a, The Wrong Way to do it! -
LPP <- as.timeSeries(data(LPP2005REC))
ZRH <- as.timeSeries(LPP[,"SPI"], zone = "Zurich", FinCenter = "Zurich")
NYC <- as.timeSeries(LPP[,"LMI"], zone = "NewYork", FinCenter = "NewYork")
finCenter(ZRH)
fncenter(NYC)
plot(ZRH, at="auto", type = "p", pch = 19, col = "blue")
points(NYC, pch = 19, col = "red")

## Example Plot 4b, Convert NYC to Zurich Time -
finCenter(ZRH) <- "Zurich"
finCenter(NYC) <- "Zurich"
at <- unique(round(time(ZRH)))
plot(ZRH, type = "p", pch = 19, col = "blue", format = "%b %d", at = at,
    xlab = paste(ZRH@FinCenter, "local Time"), main = ZRH@FinCenter)
points(NYC, pch = 19, col = "red")

## Example 4c, Force Everything to GMT Using "FinCenter" Argument -
finCenter(ZRH) <- "Zurich"
finCenter(NYC) <- "NewYork"
at <- unique(round(time(ZRH)))
plot(ZRH, type = "p", pch = 19, col = "blue", format = "%b %d", at = at,
    FinCenter = "GMT", xlab = "GMT", main = "ZRH - GMT")
points(NYC, FinCenter = "GMT", pch = 19, col = "red")
```

Description

Print 'timeSeries' objects.
Arguments

object an object of class timeSeries.

Value

Prints an object of class timeSeries.

Examples

```r
## Load Microsoft Data -
setRmetricsOptions(myFinCenter = "GMT")
LPP <- MSFT[1:12, 1:4]

## Abbreviate Column Names -
colnames(LPP) <- abbreviate(colnames(LPP), 6)

## Print Data Set -
print(LPP)

## Alternative Use, Show Data Set -
show(LPP)
```

Description

Returns the sample ranks of the values of a 'timeSeries' object.

Usage

```r
## S4 method for signature 'timeSeries'
rank(x, na.last = TRUE, ties.method = )
```

Arguments

- **x**: an univariate object of class timeSeries.
- **na.last**: for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed; if "keep" they are kept with rank NA.
- **ties.method**: a character string specifying how ties are treated; can be abbreviated.
Details

If all components are different (and no NAs), the ranks are well defined, with values in seq_len(x). With some values equal (called ???ties???), the argument ties.method determines the result at the corresponding indices. The "first" method results in a permutation with increasing values at each index set of ties. The "random" method puts these in random order whereas the default, "average", replaces them by their mean, and "max" and "min" replaces them by their maximum and minimum respectively, the latter being the typical sports ranking.

NA values are never considered to be equal: for na.last = TRUE and na.last = FALSE they are given distinct ranks in the order in which they occur in x.

Value

returns the ranks of a timeSeries object.

Examples

```r
## Load Microsoft Data -
X <- 100 * returns(MSFT)

## Compute the Ranks -
head(rank(X[, "Open"], 10)

## Only Interested in the Vector, then use -
head(rank(series(X[, "Open"])), 10)
```

---

readSeries

Reads a 'timeSeries' from a File

Description

Reads a file in table format and creates a timeSeries object from it.

Usage

```r
readSeries(file, header = TRUE, sep = ";", zone = "", FinCenter = "", format, ...)
```

Arguments

- **file**: the filename of a spreadsheet data set from which to import the data records.
- **FinCenter**: a character with the the location of the financial center named as "continent/city".
- **header**: a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: 'header' is set to 'TRUE' if and only if the first row contains one fewer field than the number of columns.
format a character string with the format in POSIX notation specifying the timestamp format. Note, the format has not to be specified if the first column in the file has the timestamp format specifier, e.g. \texttt{"\%Y-\%m-\%d"} for the short ISO 8601 format.

sep the field separator used in the spreadsheet file to separate columns. By default \\texttt{";"}. Note, if \texttt{sep=";"} is specified, and reading the series fails, then the reading is automatically repeated with \texttt{sep="","}.

zone the time zone or financial center where the data were recorded. By default \texttt{zone=""} which is short for GMT.

... Additional arguments passed to \texttt{read.table()} function which is used to read the file.

**Details**

The first column of the table must hold the timestamps. Format of the timestamps can be either specified in the header of the first column or by the \texttt{format} argument.

**Value**

Returns a S4 object of class \texttt{timeSeries}.

---

**Description**

Compute financial returns from prices or indexes.

**Usage**

\begin{verbatim}
returns(x, ...) returns0(x, ...)

## S4 method for signature 'ANY'
returns(x, method = c("continuous", "discrete",
    "compound", "simple"), percentage = FALSE, ...)

## S4 method for signature 'timeSeries'
returns(x, method = c("continuous", "discrete",
    "compound", "simple"), percentage = FALSE, na.rm = TRUE,
    trim = TRUE, ...)

getReturns(...)
returnSeries(...)
\end{verbatim}
Arguments

- **x**: an object of class `timeSeries`.
- **percentage**: a logical value. By default FALSE, if TRUE the series will be expressed in percentage changes.
- **method**: a character string. Which method should be used to compute the returns, "continuous", "discrete", or "compound", "simple". The second pair of methods is a synonyme for the first two methods.
- **na.rm**: a logical value. Should NAs be removed? By Default TRUE.
- **trim**: a logical value. Should the time series be trimmed? By Default TRUE.
- ...: arguments to be passed.

Value

all functions return an object of class `timeSeries`.

`returns` returns an untrimmed series with the first row of returns set to zero(s).

Note

The functions `returnSeries`, `getReturns`, are synonymes for the function `returns`. We do not recommend to use these functions.

Examples

```r
## Load Microsoft Data -
setRmetricsOptions(myFinCenter = "GMT")
data(MSFT)
X = MSFT[1:10, 1:4]
X

## Continuous Returns -
returns(X)
returns0(X)

## Discrete Returns:
returns(X, method = "discrete")

## Don't trim:
returns(X, trim = FALSE)

## Use Percentage Values:
returns(X, percentage = TRUE, trim = FALSE)
```
### rev

**Reversion of a 'timeSeries'**

**Description**

Reverses an uni- or multivariate 'timeSeries' object by reversing the order of the time stamps.

**Usage**

```r
## S4 method for signature 'timeSeries'
rev(x)
```

**Arguments**

- `x`: an uni- or multivariate 'timeSeries' object.

**Value**

Returns a reversed 'timeSeries' object.

**Examples**

```r
## Create Dummy timeSeries -
ts <- dummySeries()

## Reverse Series -
rev(ts)
```

---

### rollmean

**Rolling Statistics**

**Description**

Computes rolling mean, min, max and median for a 'timeSeries' object.

**Usage**

```r
rollStats(x, k, FUN=mean, na.pad=FALSE,
         align=c("center", "left", "right"), ...)
rollMean(x, k, na.pad = FALSE,
         align = c("center", "left", "right"), ...)
rollMin(x, k, na.pad = FALSE,
        align = c("center", "left", "right"), ...)
rollMax(x, k, na.pad = FALSE,
```
rollMean

```r
grow = c("center", "left", "right"), ...)
rollMedian(x, k, na.pad = FALSE,
grow = c("center", "left", "right"), ...)

Arguments

x an uni-or multivariate 'timeSeries' object.
k an integer width of the rolling window. Must be odd for rollMedian.
FUN the function to be rolled.
na.pad a logical flag. Should NA padding be added at beginning? By default FALSE.
align a character string specifying whether the index of the result should be left- or
right-aligned or centered compared to the rolling window of observations. The
default choice is set to align="center".
...
optional arguments to be passed.

Details

The code in the core of the functions rollMean, rollMin, rollMax, and rollMedian was borrowed
from the package zoo authored by Achim Zeileis, Gabor Grothendieck and Felix Andrews.

Value

returns an object of class 'timeSeries'.

Author(s)

Achim Zeileis, Gabor Grothendieck and Felix Andrews for code from the contributed R package
zoo used in the functions rollMean, rollMin, rollMax, and rollMedian.

Examples

```r
## Use Swiss Pension Fund Data Set of Returns -
head(LPP2005REC)
SPI <- LPP2005REC[, "SPI"]
head(SPI)

## Plot Drawdowns -
mean <- rollMean(SPI, k = 10)
plot(rmean)
**rowCum**  

*Cumulated Column Statistics*

**Description**
Compute cumulative row Statistics.

**Usage**

```r
## S4 method for signature 'ANY'
rowCumsums(x, na.rm = FALSE, ...)
## S4 method for signature 'timeSeries'
rowCumsums(x, na.rm = FALSE, ...)
```

**Arguments**
- `na.rm` a logical. Should missing values be removed?
- `x` a time series, may be an object of class "matrix" or "timeSeries".
- `...` arguments to be passed.

**Value**

all functions return an S4 object of class `timeSeries`.

**Examples**

```r
## Simulated Monthly Return Data -
X = matrix(rnorm(24), ncol = 2)
## Compute cumulated Sums -
rowCumsums(X)
```

---

**runlengths**  

*Runlengths of a Time Series*

**Description**
Computes runlengths of an univariate 'timeSeries' object.

**Usage**

```r
runlengths(x, ...)
```

**Arguments**
- `x` an univariate time series of class 'timeSeries'.
- `...` arguments to be passed.
Value

returns an object of class `timeSeries`.

Examples

```r
## random time series -
set.seed(4711)
x <- rnorm(12)
tS <- timeSeries(data=x, charvec=timeCalendar(), units="x")
tS

## return runlengths -
runlengths(tS)
```

---

**sample**

### Description

Takes a sample of the specified size from the elements of a `timeSeries`.

### Value

Returns a resampled `timeSeries` object.

### Examples

```r
## Monthly Calendar Series -
x <- daily2monthly(LPP2005REC[,1:2])[3:14,]

## Resample the Series with respect to the time stamps -
resampled <- sample(x)
resampled
is.unsorted(resampled)
```

---

**scale**

### Description

Scales a `timeSeries` object.
Details

scale is a function to center and/or scale the columns of a 'timeSeries' object.

The value of center determines how column centering is performed. If center is a numeric vector with length equal to the number of columns of x, then each column of x has the corresponding value from center subtracted from it. If center is TRUE then centering is done by subtracting the column means (omitting NAs) of x from their corresponding columns, and if center is FALSE, no centering is done.

The value of scale determines how column scaling is performed (after centering). If scale is a numeric vector with length equal to the number of columns of x, then each column of x is divided by the corresponding value from scale. If scale is TRUE then scaling is done by dividing the (centered) columns of x by their standard deviations if center is TRUE, and the root mean square otherwise. If scale is FALSE, no scaling is done.

Value

Returns a centered and/or scaled 'timeSeries' object.

Examples

```r
## Load Series:
  x <- 100* LPP2005REC[, c("SBI", "SPI")]

## Scale and Center -
  X <- scale(x)
  hist(X[, 1], prob=TRUE)
  s <- seq(-3, 3, length=201)
  lines(s, dnorm(s, col="red")
```

Description

series returns the @.Data slot of a timeSeries object in matrix form. New series can also be assign to an already existing timeSeries.

coredata is a synonyme function nameing for series.

Usage

```r
series(x)
series(x) <- value
```
Arguments

x  a timeSeries object.
value a vector, a data.frame or a matrix object of numeric data.

See Also
timeSeries()

Examples

## A Dummy timeSeries Object
  ts <- timeSeries()
ts

## Get the Matrix Part -
  mat <- series(ts)
class(mat)
  mat

## Assign a New Univariate Series -
  series(ts) <- rnorm(12)
ts

## Assign a New Bivariate Series -
  series(ts) <- rnorm(12)
ts

smooth Smoothes Time Series Objects

Description

Smothes a 'timeSeries' object.

Usage

smoothLowess(x, f = 0.5, ...)
smoothSpline(x, spar = NULL, ...)
smoothSupsmu(x, bass = 5, ...)

Arguments

x  an univariate 'timeSeries' object.

f  the lowess smoother span. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness.

spar smoothing parameter, typically (but not necessarily) in (0,1]. By default NULL, i.e. the value will be automatically selected.
bass controls the smoothness of the fitted curve. Values of up to 10 indicate increasing smoothness.

... optional arguments to be passed to the underlying smoothers.

Details

The functions smoothLowess, smoothSpline, smoothSupsmu allow to smooth timeSerie object. The are interfaces to the function lowess, supmsu. and smooth.spline in R’s stats package.

The ... arguments allow to pass optional arguments to the underlying stats functions and tailor the smoothing process. We refer to the manual pages of these functions for a proper setting of these options.

Value

returns a bivariate 'timeSeries' object, the first column holds the original time series data, the second the smoothed series.

Author(s)

The R core team for the underlying smoother functions.

Examples

```r
## Use Close from MSFT's Price Series -
head(MSFT)
MSFT.CLOSE <- MSFT[, "Close"]
head(MSFT.CLOSE)

## Plot Original and Smoothed Series by Lowess -
MSFT.LOWESS <- smoothLowess(MSFT.CLOSE, f = 0.1)
head(MSFT.LOWESS)
plot(MSFT.LOWESS)
title(main = "Close - Lowess Smoothed")

## Plot Original and Smoothed Series by Splines -
MSFT.SPLINE <- smoothSpline(MSFT.CLOSE, spar = 0.4)
head(MSFT.SPLINE)
plot(MSFT.SPLINE)
title(main = "Close - Spline Smoothed")

## Plot Original and Smoothed Series by Supsmu -
MSFT.SUPSMU <- smoothSupsmu(MSFT.CLOSE)
head(MSFT.SUPSMU)
plot(MSFT.SUPSMU)
title(main = "Close - Spline Smoothed")
```
Description

Sorts a 'timeSeries' object with respect to its time stamps.

Usage

```r
## S4 method for signature 'timeSeries'
sort(x, decreasing = FALSE, ...)
```

Arguments

- `x`: an uni- or multivariate `timeSeries` object.
- `decreasing`: a logical flag. Should we sort in increasing or decreasing order? By default `FALSE`.
- `...`: optional arguments passed to other methods.

Details

Sorts a time series either in increasing or decreasing time stamp order. Internally the function `order` from R's base package is used. `order` generates a permutation which rearranges the time stamps in ascending or descending order.

To find out if the series is unsorted, the function `isNunsorted` from R's base package can be called.

Value

Returns a sorted 'timeSeries' object, which can be increasing or decreasing in time.

Examples

```r
## Monthly Calendar Series -
x <- daily2monthly(LPP2005REC[, 1:2])[3:14, ]

## Resample the Series with respect to the time stamps -
resampled <- sample(x)
resampled
is.unsorted(resampled)

## Now sort the serie in decreasing time order -
sorted <- sort(resampled, , decreasing = TRUE)
sorted
is.unsorted(sorted)

## Is the reverted series ordered? -
reverted <- rev(sorted)
```
reverted
is.unsorted(reverted)

Special Daily Time Series

Description
Special daily 'timeSeries' functions.

Usage
dummyDailySeries(x = rnorm(365), units = NULL, zone = "", FinCenter = "")
alignDailySeries(x, method = c("before", "after", "interp", "fillNA", "fmm", "periodic", "natural", "monH.FC"), include.weekends = FALSE, units = NULL, zone = "", FinCenter = "", ...)
rollDailySeries(x, period = "7d", FUN, ...)

Arguments
FinCenter a character with the the location of the financial center named as "continent/city".
FUN the function to be applied.
[applySeries] - a function to use for aggregation, by default colAvgs.
include.weekends
[alignDailySeries] - a logical value. Should weekend dates be included or removed from the series.
method
[alignDailySeries] - the method to be used for the alignment. A character string, one of "before", use the data from the row whose position is just before the unmatched position, or "after", use the data from the row whose position is just after the unmatched position, or "linear", interpolate linearly between "before" and "after".
period
[rollDailySeries] - a character string specifying the rolling period composed by the length of the period and its unit, e.g. "7d" represents one week.
units
[alignDailySeries] - an optional character string, which allows to overwrite the current column names of a timeSeries object. By default NULL which means that the column names are selected automatically.
x an object of class timeSeries.
zone the time zone or financial center where the data were recorded.
... arguments passed to interpolating methods.

Details

dummyDailySeries  Creates a dummy daily 'timeSeries' object,
alignDailySeries   Aligns a daily 'timeSeries' to new positions,
rollDailySeries   Rolls daily a 'timeSeries' on a given period,
ohlcdailyPlot     Plots open high low close bar chart,
dummySeries       Creates a dummy monthly 'timeSeries' object

Value

dummyDailySeries
creates from a numeric matrix with daily records of unknown dates a timeSeries object with
dummy daily dates.

alignDailySeries
returns from a daily time series with missing holidays a weekly aligned daily timeSeries object

rollDailySeries
returns an object of class timeSeries with rolling values, computed from the function FUN.

Examples

## Use Microsoft's OHLCV Price Series -
head(MSFT)
end(MSFT)

## Cut out April Data from 2001 -
Close <- MSFT[, "Close"]
tsApril01 <- window(Close, start="2001-04-01", end="2001-04-30")
tsApril01

## Align Daily Series with NA -
tsRet <- returns(tsApril01, trim = TRUE)
GoodFriday(2001)
EasterMonday(2001)
alignDailySeries(tsRet, method = "fillNA", include.weekends = FALSE)
alignDailySeries(tsRet, method = "fillNA", include.weekends = TRUE)

## Align Daily Series by Interpolated Values -
alignDailySeries(tsRet, method = "interp", include.weekend = FALSE)
alignDailySeries(tsRet, method = "interp", include.weekend = TRUE)
splits

Description

Searches for outlier splits in a 'timeSeries' object.

Usage

splits(x, sd = 3, complement = TRUE, ...)

Arguments

x a 'timeSeries' object.

sd a numeric value of standard deviations, e.g. 5 means that values larger or smaller than five times the standard deviation of the series will be detected.

complement a logical flag, should the outlier series or its complements be returned?

... arguments to be passed.

Details

This function is thought to find splits in financial price or index series If a price or index is splitted we observe in the returns a big jump of several standard deviations which is identified usual as an outlier.

Examples

## Create a Return Series with a Split -
  data <- runif(12, -1, 1)
  data[6] <- 20
  x <- timeSeries(data, timeCalendar(), units="RUNIF")
  x

## Search for the Split:
  splits(x, sd=3, complement=TRUE)
  splits(x, sd=3, complement=FALSE)

spreads

Description

Compute spreads and midquotes from price streams.
Usage

```r
spreads(x, which = c("Bid", "Ask"), tickSize = NULL)
midquotes(x, which = c("Bid", "Ask"))

midquoteSeries(...)
spreadSeries(...)
```

Arguments

- `tickSize` the default is NULL to simply compute price changes in original price levels. If ticksize is supplied, the price changes will be divided by the value of `inTicksOfSize` to compute price changes in ticks.
- `which` a vector with two character strings naming the column names of the time series from which to compute the mid quotes and spreads. By default these are bid and ask prices with column names `c("Bid", "Ask")`.
- `x` an object of class `timeSeries`.
- `...` arguments to be passed.

Value

all functions return an object of class `timeSeries`.

Note

The functions `returnSeries`, `getReturns`, `midquoteSeries`, `spreadSeries` are synonyms for `returns`, `midquotes`, and `spreads`.

Examples

```r
## Load the Microsoft Data -
setRmetricsOptions(myFinCenter = "GMT")
data(MSFT)
X = MSFT[1:10, ]
head(X)

## Compute Open/Close Midquotes -
X.MID <- midquotes(X, which = c("Close", "Open"))
colnames(X.MID) <- "X.MID"
X.MID

## Compute Open/Close Spreads -
X.SPREAD <- spreads(X, which = c("Close", "Open"))
colnames(X.SPREAD) <- "X.SPREAD"
X.SPREAD
```
Start and End of a 'timeSeries'

Description

Returns start and/or end time stamps of a 'timeSeries' object.

Usage

## S4 method for signature 'timeSeries'
start(x, ...)

## S4 method for signature 'timeSeries'
end(x, ...)

Arguments

x    an uni- or multivariate `timeSeries` object.

...  optional arguments passed to other methods.

Value

returns a `timeSeries` object.

Examples

## Create Dummy `timeSeries` -
ts <- dummySeries(1, 1)
ts

c <- start(tS), end(tS))
range(time(tS))

str-methods

Description

Compactly display the structure of a 'timeSeries' Object.

Usage

## S4 method for signature 'timeSeries'
str(object, ...)

```r
## Create Dummy `timeSeries` -
ts <- dummySeries(1, 1)
ts

c <- start(tS), end(tS))
range(time(tS))
```
Arguments

object an object of class `timeseries`.

... arguments passed to other methods.

Value

returns a str report for an object of class `timeseries`.

Examples

```r
## Load Microsoft Data Set -
data(MSFT)
X <- MSFT[1:12, 1:4]
colnames(X) <- abbreviate(colnames(X), 4)

## Display Structure -
str(X)
```

---

### t

`timeSeries Transpose`

Description

Returns the transpose of a `timeSeries` object.

Usage

```r
## S4 method for signature 'timeSeries'
t(x)
```

Arguments

x a `timeSeries` object.

Value

Returns a matrix object.

Examples

```r
## Dummy timeSeries with NAs entries
data <- matrix(1:24, ncol = 2)
s <- timeSeries(data, timeCalendar())
s

## Transpose 'timeSeries' -
t(s)
```
Get and Set Time stamps of a 'timeSeries'  

Description

Functions and methods extracting and modifying positions of 'timeSeries' objects.

Usage

```r
getTime(x)
setTime(x) <- value

# S4 method for signature 'timeSeries'
time(x, ...)
# S3 replacement method for class 'timeSeries'
time(x) <- value
```

Arguments

- `value` a valid value for the component of `time(x)`.  
- `x` an object of class `timeSeries`.  
- `...` optional arguments passed to other methods.

Value

Returns a 'timeDate' object.

Examples

```r
## Create Dummy timeSeries -
X <- timeSeries(matrix(rnorm(24), 12), timeCalendar())

## Return Series Positions -
getTime(X)
time(X)

## Add / Subtract one Day from X
setTime(X) <- time(X) - 24*3600 # sec
X
time(X) <- time(X) + 24*3600 # sec
X
```

Deprecated functions in `timeSeries` package

Description
seriesPositions <- Extracts positions slot from a 'timeSeries',
newPositions<- Modifies positions of a 'timeSeries' object,

timeSeries-method-stats

Time Series Correlations

Description

S4 methods of stats package for timeSeries objects.

cov Computes Covariance from a 'timeSeries' object,
cor Computes Correlations from a 'timeSeries' object.
dcauchy ...
dnorm ...
dt ...

Usage

## S4 method for signature 'timeSeries'
cov(x, y = NULL, use = "all.obs", method = c("pearson", "kendall", "spearman"))

## S4 method for signature 'timeSeries'
cor(x, y = NULL, use = "all.obs", method = c("pearson", "kendall", "spearman"))

Arguments

method a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
use an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "all.obs", "complete.obs" or "pairwise.complete.obs".
x an univariate object of class timeSeries.
y NULL (default) or a timeSeries object with compatible dimensions to x. The default is equivalent to y = x (but more efficient).

Value

returns the covariance or correlation matrix.
Examples

```r
## Load Microsoft Data Set -
data(MSFT)
X <- MSFT[, 1:4]
X <- 100 * returns(X)

## Compute Covariance Matrix -
cov(X[, "Open"], X[, "Close"])
cov(X)
```

Description

Functions to generate and modify 'timeSeries' objects:

```r
timeSeries  Creates a 'timeSeries' object from scratch.
```

Data Slot and classification of 'timeSeries' objects:

```r
seriesData  Extracts data slot from a 'timeSeries'.
```

Usage

```r
timeSeries(data, charvec, units = NULL, format = NULL, zone = "", 
FinCenter = "", recordIDs = data.frame(), title = NULL, 
documentation = NULL, ...)

seriesData(object)
```

Arguments

```r
charvec  a character vector of dates and times or any objects which can be coerced to a 
timeDate object.
data  a matrix object or any objects which can be coerced to a matrix.
documentation  optional documentation string, or a vector of character strings.
FinCenter  a character with the the location of the financial center named as "continent-city".
format  the format specification of the input character vector,
[as.timeSeries] -
a character string with the format in POSIX notation to be passed to the time 
series object.
object  [is][seriesData][seriesPositions][show][summary] - an object of class timeSeries.
recordIDs  a data frame which can be used for record identification information.
[print] -
```
TimeSeriesClass

a logical value. Should the recordIds printed together with the data matrix and
time series positions?

**title**
an optional title string, if not specified the inputs data name is deparsed.

**units**
an optional character string, which allows to overwrite the current column names
of a timeSeries object. By default NULL which means that the column names
are selected automatically.

**zone**
the time zone or financial center where the data were recorded.

...arguments passed to other methods.

Details

**Generation of Time Series Objects:**

We have defined a timeSeries class which is in many aspects similar to the S-Plus class with the
same name, but has also some important differences. The class has seven Slots, the 'Data' slot
which holds the time series data in matrix form, the 'position' slot which holds the time/date as a
character vector, the 'format' and 'FinCenter' slots which are the same as for the 'timeDate' object,
the 'units' slot which holds the column names of the data matrix, and a 'title' and a 'documentation'
slot which hold descriptive character strings. Date and time is managed in the same way as for
timeDate objects.

**Value**

`timeSeries`
returns a S4 object of class timeSeries.

`seriesData`

extracts the .Data slot from a timeSeries object and is equivalent to `as.matrix`.

**Note**

These functions were written for Rmetrics users using R and Rmetrics under Microsoft’s Windows
operating system where time zones, daylight saving times and holiday calendars are insufficiently
supported.

**Examples**

```r
## Load Microsoft Data -
# Microsoft Data:
setRmetricsOptions(myFinCenter = "GMT")
data(MSFT)
head(MSFT)

## Create a timeSeries Object, The Direct Way ...
Close <- MSFT[, 5]
head(Close)
```
## Description

Three data sets used in example files.

The data sets are:

- **LPP2005REC**: Swiss pension fund assets returns benchmark,
- **MSFT**: Daily Microsoft OHLC prices and volume,
- **USDCHF**: USD CHF intraday foreign exchange exchange rates.
Examples

```r
## Plot LPP2005 Example Data Set -
data(LPP2005REC)
plot(LPP2005REC, type = "l")

## Plot MSFT Example Data Set -
data(MSFT)
plot(MSFT[, 1:4], type = "l")
plot(MSFT[, 5], type = "h")

## Plot USDCHF Example Data Set -
# plot(USDCHF)
```

---

Description

Subset a 'timeSeries' object due to different aspects.

- `[]`: "[" method for a 'timeSeries' object,
- `[-]`: "[-" method to assign value for a subset of a 'timeSeries' object,
- `window`: Windows a piece from a 'timeSeries' object,
- `cut`: A no longer used synonyme for window,
- `head`: Returns the head of a 'timeSeries' object,
- `tail`: Returns the tail of a 'timeSeries' object,
- `outliers`: Removes outliers from a 'timeSeries' object.

Usage

```r
## S4 method for signature 'timeSeries'
window(x, start, end, ...)

## S4 method for signature 'timeSeries'
head(x, n = 6, recordIDs = FALSE, ...)

## S4 method for signature 'timeSeries'
tail(x, n = 6, recordIDs = FALSE, ...)

## S4 method for signature 'timeSeries'
outlier(x, sd = 3, complement = TRUE, ...)

## S4 method for signature 'timeSeries'
```
cut(x, from, to, ...)

Arguments

complement [outlierSeries] -
a logical flag, should the outlier series or its complement be returns, by default
TRUE which returns the series free of outliers.

from, to starting date and end date, to must be after from.

start, end starting date and end date, end must be after start.

n [head][tail] -
an integer specifying the number of lines to be returned. By default n=6.

recordIDs [head][tail] -
a logical value. Should the recordIDs returned together with the data matrix
and time series positions?

sd [outlierSeries] -
a numeric value of standard deviations, e.g. 10 means that values larger or
smaller tahn ten times the standard deviation will be removed from the series.

x an object of class timeSeries.

... arguments passed to other methods.

Value

All functions return an object of class 'timeSeries'.

Examples

## Create an Artificial timeSeries Object -
setRmetricsOptions(myFinCenter = "GMT")
charvec <- timeCalendar()
set.seed(4711)
data <- matrix(exp(cumsum(rnorm(12, sd = 0.1))))
tsS <- timeSeries(data, charvec, units = "tS")
tS

## Subset Series by Counts 
"[" -
tS[1:3, ]

## Subset the Head of the Series -
head(tS, 6)

---

**Description**

Extracts and analyzes turn points of an univariate timeSeries object.
Usage

`turns(x, ...)`

`turnsStats(x, doplot = TRUE)`

Arguments

- `x` an univariate 'timeSeries' object of financial indices or prices.
- `...` optional arguments passed to the function `na.omit`.
- `doplot` a logical flag, should the results be plotted? By default TRUE.

Details

The function `turns` determines the number and the position of extrema (turning points, either peaks or pits) in a regular time series.

The function `turnsStats` calculates the quantity of information associated to the observations in this series, according to Kendall’s information theory.

The functions are borrowed from the contributed R package `pastecs` and made ready for working together with univariate `timeSeries` objects. You need not to load the R package `pastecs`, the code parts we need here are built-in in the `timeSeries` package.

We have renamed the function `turnpoints` to `turns` to distinguish between the original function in the contributed R package `pastecs` and our Rmetrics function wrapper.

For further details please consult the help page from the contributed R package `pastecs`.

Value

`turns` returns an object of class `timeSeries`.

`turnsStats` returns an object of class `turnpoints` with the following entries:
- `data` - The dataset to which the calculation is done.
- `n` - The number of observations.
- `points` - The value of the points in the series, after elimination of ex-aequos.
- `pos` - The position of the points on the time scale in the series (including ex-aequos).
- `exaequos` - Location of exaequos (1), or not (0).
- `nturns` - Total number of turning points in the whole time series.
- `firstispeak` - Is the first turning point a peak (TRUE), or not (FALSE).
- `peaks` - Logical vector. Location of the peaks in the time series without ex-aequos.
- `pits` - Logical vector. Location of the pits in the time series without ex-aequos.
- `tppos` - Position of the turning points in the initial series (with ex-aequos).
- `proba` - Probability to find a turning point at this location.
- `info` - Quantity of information associated with this point.
units

Author(s)
Frederic Ibanez and Philippe Grosjean for code from the contributed R package pastecs and Rmetrics for the function wrapper.

References

Examples
```r
## Load Swiss Equities Series -
SPI.RET <- LPP2005REC[, "SPI"]
head(SPI.RET)

## Cumulate and Smooth the Series -
SPI <- smoothLoess(cumulated(SPI.RET), f=0.05)
plot(SPI)

## Plot Turn Points Series -
SPI.SMOOTH <- SPI[, 2]
tP <- turns(SPI.SMOOTH)
plot(tP)

## Compute Statistics -
turnsStats(SPI.SMOOTH)
```

---

### units

Get and Set Unit Names of a 'timeSeries'

Description
Gets and sets the column names of a 'timeSeries' object. The column names are also called units or unit names.

Usage
```r
getUnits(x)
setUnits(x) <- value
```

Arguments
- **x**: a 'timeSeries' object.
- **value**: a vector of unit names.
See Also
timeSeries()

Examples

## A Dummy timeSeries Object
tS <- dummySeries()
tS

## Get the Units -
getUnits(tS)

## Assign New Units to the Series -
setUnits(tS) <- c("A", "B")
head(tS)

wealth
Conversion of an index to wealth

Description
Converts an index series to a wealth series normalizing the starting value to one.

Usage
index2wealth(x)

Arguments

x
 an object of class 'timeSeries'.

Value
returns a time series object of the same class as the input argument x normalizing the starting value to one.

Examples

## Load MSFT Open Prices -
INDEX <- MSFT[1:20, 1]
INDEX

## Compute Wealth Normalized to 100 -
100 * index2wealth(INDEX)
Description

Extracts a part from a 'timeSeries Object

Examples

```r
## Load LPP Benchmark Returns -
x <- LPP2005REC[, 7:9]
range(time(x))

## Extract Data for January 2006 -
window(x, "2006-01-01", "2006-01-31")
```
Index

*Topic chron
  aggregate-methods, 7
  align-methods, 8
  apply, 9
  as, 11
  attach, 13
  base-methods, 15
  bind, 15
  comment, 18
  cumulated, 19
  DataPart, TimeSeries-method, 20
  diff, 21
  dimnames, 22
  drawdowns, 23
  durations, 24
  is.timeSeries, 27
  isRegular, 27
  isUnivariate, 29
  lag, 30
  math, 30
  merge, 32
  model.frame, 32
  monthly, 33
  orderColNames, 38
  orderStatistics, 39
  periodical, 40
  plot-methods, 41
  print-methods, 44
  rank, 45
  returns, 47
  rev, 49
  rollMean, 49
  runlengths, 51
  sample, 52
  scale, 52
  smooth, 54
  sort, 56
  SpecialDailySeries, 57
  spreads, 59
  start, 61
  str-methods, 61
  t, 62
  time, 63
  timeSeries-method-stats, 64
  TimeSeriesClass, 65
  TimeSeriesSubsettings, 68
  turns, 69
  wealth, 72
  window, 73
*Topic datasets
  TimeSeriesData, 67
*Topic math
  na, 35
*Topic methods
  aggregate-methods, 7
  align-methods, 8
  math, 30
  timeSeries-method-stats, 64
*Topic package
  TimeSeries-package, 3
*Topic programming
  attributes, 14
  description, 20
  finCenter, 26
  series-methods, 53
  units, 71
*Topic univar
  colCum, 16
  colStats, 17
  rowCum, 51
  +, timeSeries, missing-method (math), 30
  -, timeSeries, missing-method (math), 30
  [, timeSeries, ANY, index_timeSeries-method (TimeSeriesSubsettings), 68
  [, timeSeries, character, character-method (TimeSeriesSubsettings), 68
  [, timeSeries, character, index_timeSeries-method (TimeSeriesSubsettings), 68
INDEX
attach, timeSeries-method (attach), 13
attributes, 14
base-methods, 15
bind, 15
cbind (bind), 15
cbind2 (bind), 15
cbind2, ANY, timeSeries-method (bind), 15
cbind2, timeSeries, ANY-method (bind), 15
cbind2, timeSeries, missing-method (bind), 15
cbind2, timeSeries, timeSeries-method (bind), 15
coerce, ANY, timeSeries-method (as), 11
coerce, character, timeSeries-method (as), 11
coerce, data.frame, timeSeries-method (as), 11
coerce, timeSeries, data.frame-method (as), 11
coerce, timeSeries, list-method (as), 11
colavgs (colStats), 17
colCum, 16
colCummaxs (colCum), 16
colCummaxs, matrix-method (colCum), 16
colCummaxs, timeSeries-method (colCum), 16
colCummins (colCum), 16
colCummins, matrix-method (colCum), 16
colCummins, timeSeries-method (colCum), 16
colCumprods (colCum), 16
colCumprods, matrix-method (colCum), 16
colCumprods, timeSeries-method (colCum), 16
colCumreturns (colCum), 16
colCumreturns, matrix-method (colCum), 16
colCumreturns, timeSeries-method (colCum), 16
colCumsums (colCum), 16
colCumsums, matrix-method (colCum), 16
colCumsums, timeSeries-method (colCum), 16
colKurtosis (colStats), 17
colMaxs (colStats), 17
colMeans, timeSeries-method (colStats), 17
colMins (colStats), 17
colnames, timeSeries-method (dimnames), 22
colnames <-, timeSeries-method (dimnames), 22
colProds (colStats), 17
colQuantiles (colStats), 17
colSds (colStats), 17
colSkewness (colStats), 17
colStats, 17
colStdevs (colStats), 17
colSums, timeSeries-method (colStats), 17
colVars (colStats), 17
comment, 18
comment, timeSeries-method (comment), 18
cov, timeSeries-method (timeSeries-method-stats), 64
cov-methods (timeSeries-method-stats), 64
coredata (series-methods), 53
coredata, timeSeries-method (series-methods), 53
coredata <- (series-methods), 53
coredata <-, timeSeries, ANY-method (series-methods), 53
coredata <-, timeSeries, data.frame-method (series-methods), 53
coredata <-, timeSeries, matrix-method (series-methods), 53
coredata <-, timeSeries, vector-method (series-methods), 53
countMonthlyRecords (monthly), 33
cov, timeSeries-method (timeSeries-method-stats), 64
cov-methods (timeSeries-method-stats), 64
cummax, timeSeries-method (math), 30
cummin, timeSeries-method (math), 30
cumprod, timeSeries-method (math), 30
cumsum, timeSeries-method (math), 30
cumulated, 19
cut, timeSeries-method (TimeSeriesSubsettings), 68
INDEX


cut.timeSeries (TimeSeriesSubsettings), 68
daily (SpecialDailySeries), 57
daily2monthly (aggregate-methods), 7
daily2weekly (aggregate-methods), 7
DataPart, timeSeries-method, 20
dcauchy, timeSeries-method (timeSeries-method-stats), 64
dcauchy-methods (timeSeries-method-stats), 64
description, 20
diff, 21
diff, timeSeries-method (diff), 21
diff.timeSeries (math), 30
dim, timeSeries-method (dimnames), 22
dim<-, timeSeries-method (dimnames), 22
dimnames, timeSeries-method (dimnames), 22
dimnames<-, timeSeries, list-method (dimnames), 22
dnorm, timeSeries-method (timeSeries-method-stats), 64
dnorm-methods (timeSeries-method-stats), 64
documentation (attributes), 14
drawdowns, 23
drawdownsStats (drawdowns), 23
dt, timeSeries-method (timeSeries-method-stats), 64
dt-methods (timeSeries-method-stats), 64
dummyDailySeries (SpecialDailySeries), 57
dummySeries (SpecialDailySeries), 57
durations, 24
durationSeries (durations), 24
end, timeSeries-method (start), 61
dend.timeSeries (start), 61
dendOfPeriod (periodical), 40
dendOfPeriodBenchmarks (periodical), 40
dendOfPeriodSeries (periodical), 40
dendOfPeriodStats (periodical), 40
fapply (apply), 9
filter, 25
filter, timeSeries-method (filter), 25
finCenter, timeSeries-method (finCenter), 26
finCenter<-, timeSeries-method (finCenter), 26
frequency, timeSeries-method (isRegular), 27
getAttributes (attributes), 14
getDataPart, timeSeries-method (DataPart, timeSeries-method), 20
getFinCenter (finCenter), 26
getReturns (returns), 47
getime (time), 63
getUnits (units), 71
hclustColnames (orderColnames), 38
head, timeSeries-method (TimeSeriesSubsettings), 68
head.timeSeries (TimeSeriesSubsettings), 68
index2wealth (wealth), 72
index.timeSeries (TimeSeriesClass), 65
index.timeSeries-class (TimeSeriesClass), 65
initialize, timeSeries-method (TimeSeriesClass), 65
interpNA (na), 35
is.na, timeSeries-method (is.na.timeSeries), 27
is.signalSeries (is.timeSeries), 27
is.timeSeries, 27
is.unsorted, timeSeries-method (is.timeSeries), 27
isDaily, timeSeries-method (isRegular), 27
isMonthly, timeSeries-method (isRegular), 27
isMultivariate (isUnivariate), 29
isQuarterly, timeSeries-method (isRegular), 27
isRegular, 27
isRegular, timeSeries-method (isRegular), 27
isUnivariate, 29
lag, 30
lag, timeSeries-method (lag), 30
lag.timeSeries (lag), 30
lines.timeSeries-method (plot-methods), 41
log.timeSeries-method (math), 30
LPP2005REC (TimeSeriesData), 67
math, 30
Math, timeSeries-method (math), 30
Math2, timeSeries-method (math), 30
mean, timeSeries-method (base-methods), 15
merge, 32
merge, ANY, timeSeries-method (merge), 32
merge, matrix, timeSeries-method (merge), 32
merge, numeric, timeSeries-method (merge), 32
merge, timeSeries, ANY-method (merge), 32
merge, timeSeries, matrix-method (merge), 32
merge, timeSeries, missing-method (merge), 32
merge, timeSeries, numeric-method (merge), 32
merge, timeSeries, timeSeries-method (merge), 32
midquotes (spreads), 59
midquoteSeries (spreads), 59
model.frame, 32, 33
monthly, 33
MSFT (TimeSeriesData), 67
na, 35
na.contiguous, 37
na.contiguous, timeSeries-method (na.contiguous), 37
names, timeSeries-method (dimnames), 22
names<- , timeSeries-method (dimnames), 22
newPositions<- (timeSeries-deprecated), 63
Ops, array, timeSeries-method (math), 30
Ops, timeSeries, array-method (math), 30
Ops, timeSeries, timeSeries-method (math), 30
Ops, timeSeries, ts-method (math), 30
Ops, timeSeries, vector-method (math), 30
Ops, ts, timeSeries-method (math), 30
Ops, vector, timeSeries-method (math), 30
orderColnames, 38
orderStatistics, 39
outlier (TimeSeriesSubsettings), 68
outlier, ANY-method (TimeSeriesSubsettings), 68
outlier, timeSeries-method (TimeSeriesSubsettings), 68
pcaColnames (orderColnames), 38
periodical, 40
plot (plot-methods), 41
plot, timeSeries-method (plot-methods), 41
plot-methods, 41
points, timeSeries-method (plot-methods), 41
pretty.timeSeries (plot-methods), 41
print, timeSeries-method (print-methods), 44
print-methods, 44
quantile, timeSeries-method (math), 30
quantile.timeSeries (math), 30
rank, 45
rank, timeSeries-method (rank), 45
rbind (bind), 15
rbind2 (bind), 15
rbind2, ANY, timeSeries-method (bind), 15
rbind2, timeSeries, ANY-method (bind), 15
rbind2, timeSeries, missing-method (bind), 15
rbind2, timeSeries, timeSeries-method (bind), 15
readSeries, 46
removeNA (na), 35
returns, 47
returns, ANY-method (returns), 47
returns, timeSeries-method (returns), 47
returns@ (returns), 47
returnSeries (returns), 47
rev, 49
rev, timeSeries-method (rev), 49
rev, timeSeries (rev), 49
rollDailySeries (SpecialDailySeries), 57
rollMax (rollMean), 49
rollMean, 49
rollMedian (rollMean), 49
rollMin (rollMean), 49
rollMonthlySeries (monthly), 33
rollMonthlyWindows (monthly), 33
rollStats (rollMean), 49
rowCum, 51
rowCumsums (rowCum), 51
rowCumsums, ANY-method (rowCum), 51
rowCumsums, timeSeries-method (rowCum), 51
rownames, timeSeries-method (dimnames), 22
rownames<-, timeSeries, ANY-method (dimnames), 22
rownames<-, timeSeries, timeDate-method (dimnames), 22
runlengths, 51
sample, 52
sample, timeSeries-method (time), 63
sampleColnames (orderColnames), 38
scale, 52
d, timeSeries-method
(timeSeries-method-stats), 64
d-methods (timeSeries-method-stats), 64
series (series-methods), 53
series, timeSeries-method
(series-methods), 53
series-methods, 53
series<-(series-methods), 53
series<-, timeSeries, ANY-method (series-methods), 53
series<-, timeSeries, data.frame-method (series-methods), 53
series<-, timeSeries, matrix-method (series-methods), 53
series<-, timeSeries, vector-method (series-methods), 53
seriesData (TimeSeriesClass), 65
seriesPositions
(timeSeries-deprecated), 63
setAttributes<-(attributes), 14
setDataPart, timeSeries-method
(DataPart, timeSeries-method), 20
setFinCenter<-(finCenter), 26
setTime<-(time), 63
setUnits<-(units), 71
show, timeSeries-method (print-methods), 44
smooth, 54
smoothLowess (smooth), 54
smoothSpline (smooth), 54
smoothSups (smooth), 54
sort, 56
sort, timeSeries-method (sort), 56
sort.timeSeries (sort), 56
sortColnames (orderColnames), 38
SpecialDailySeries, 57
splits, 59
spreads, 59
spreadSeries (spreads), 59
start, 61
start, timeSeries-method (start), 61
start.timeSeries (start), 61
statsColnames (orderColnames), 38
str (str-methods), 61
str, timeSeries-method (str-methods), 61
str-methods, 61
substituteNA (na), 35
Summary, timeSeries-method (math), 30
summary, timeSeries-method
(base-methods), 15
t, 62
t, timeSeries-method (t), 62
tail, timeSeries-method
(TimeSeriesSubsettings), 68
tail.timeSeries
(TimeSeriesSubsettings), 68
time, 63
time, timeSeries-method (time), 63
time.timeSeries (time), 63
time<-(time), 63
time.timeSeries (TimeSeriesClass), 65
timeSeries-class
(TimeSeriesClass), 65
timeSeries (TimeSeriesClass), 65
timeSeries, ANY, ANY-method
(TimeSeriesClass), 65
timeSeries, ANY, missing-method
(TimeSeriesClass), 65
timeSeries, ANY, timeDate-method
(TimeSeriesClass), 65
timeSeries, matrix, ANY-method
(TimeSeriesClass), 65
timeSeries, matrix, missing-method
(TimeSeriesClass), 65
timeSeries, matrix, numeric-method
(TimeSeriesClass), 65
timeseriesLmatrixLtimeDateMmethod (TimeSeriesClass), 65
timeseriesLmissingLANYLmethod (TimeSeriesClass), 65
timeseriesLmissingLmissingLmethod (TimeSeriesClass), 65
timeseriesLmissingLtimeDateLmethod (TimeSeriesClass), 65
timeseries-class (TimeSeriesClass), 65
timeseries-deprecated, 63
timeseries-methodLstats, 64
timeseries-package, 3
TimeSeriesClass, 65
TimeSeriesData, 67
TimeSeriesSubsettings, 68
trunc, timeSeries-method (math), 30
turns, 69
turnsStats (turns), 69
units, 71
USDCHF (TimeSeriesData), 67

varLtimeseriesLmethod
   (timeSeries-method-stats), 64
var-methods (timeSeries-method-stats), 64

wealth, 72
window, 73
window, timeSeries-method (TimeSeriesSubsettings), 68
window.timeSeries (TimeSeriesSubsettings), 68