

Package ‘timeSeries’

September 28, 2009

Version 2100.84

Revision 4419

Date 2009-09-28

Title Rmetrics - Financial Time Series Objects

Author Diethelm Wuertz and Yohan Chalabi

Depends R (>= 2.6.0), graphics, grDevices, methods, stats, utils, timeDate (>= 2100.86)

Suggests robustbase, RUnit

Maintainer Rmetrics Core Team <Rmetrics-core@r-project.org>

Description Environment for teaching “Financial Engineering and Computational Finance”

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.

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL <http://www.rmetrics.org>

Repository CRAN

Date/Publication 2009-09-28 14:10:46

R topics documented:

timeSeries-package	3
aggregate-methods	7
align-methods	8
apply	9
as	11
attach	12
base-methods	14
bind	14
colCum	15
colStats	16
comment	17
cumulated	18
DataPart,timeSeries-method	19
description	19
dimnames	19
drawdowns	20
durations	21
filter	22
finCenter	23
is.timeSeries	24
isRegular	24
isUnivariate	26
lag	27
math	28
model.frame	29
monthly	30
na	31
na.contiguous	33
orderColnames	34
orderStatistics	35
plot-methods	36
print-methods	38
rank	39
returns	39
rowCum	40
series-methods	41
SpecialDailySeries	42
spreads	44
str-methods	45
t	46
time	47
timeSeries-deprecated	49
timeSeries-method-stats	49
TimeSeriesClass	50
TimeSeriesData	53
TimeSeriesSubsettings	54

timeSeries-package *Utilities and Tools Package*

Description

Package of time series tools and utilities.

Details

Package: timeSeries
Type: Package
URL: <http://www.rmetrics.org>

Overview of Topics:

1. S4 timeSeries Class Definition
2. Creating timeSeries Objects
3. Printing and Plotting timeSeries Objects
4. Modifying 'timeSeries' Objects

1. S4 timeSeries Class

timeSeries Class Definition

'timeSeries' ...

Extracting Information from a timeSeries object:

isUnivariate
isMultivariate

2. Creating timeSeries Objects

'timeSeries' objects can be created in several ways. One can create them from scratch, or one can read them from a file. If we have a file it is assumed that the first column holds a character string with the date/time positions, named "charvec", and the remaining column(s), depending if we consider the univariate or multivariate case the numeric time series records named "data".

Create a 'timeSeries' object from scratch:

timeSeries

Read a time Series from a file:

readSeries

Attach a 'timeSeries' object to the search path:

List of Functions:

<code>timeSeries</code>	Creates a 'timeSeries' from scratch,
<code>readSeries</code>	reads a time series from a file,
<code>attach</code>	attaches a 'timeSeries' object,
<code>detach</code>	detaches a 'timeSeries' object [see base package].

3. Printing and Plotting 'timeSeries' Objects

List of Functions:

<code>print</code>	...
<code>plot</code>	...
<code>points</code>	...
<code>lines</code>

4. Modifying 'timeSeries' Objects

If we have a price/index series or a return series, it is often required that we have to convert from one to the other representation. So Rmetrics makes functions available which compute `returns` from price/index series or the `cumulated` series from returns. Further modifications are concerned with drawdowns, durations, spreads and midquotes.

List of Functions:

<code>returns</code>	Compute returns from prices or indexes,
<code>cumulated</code>	compute cumulated series from a returns,
<code>drawdowns</code>	compute series of drawdowns from financial returns,
<code>durations</code>	compute durations from a financial time series,
<code>spreads</code>	compute spreads from a price/index stream,
<code>midquotes</code>	compute mid quotes from a price/index stream.

Subsetting timeSeries Objects

'timeSeries' objects can be subsetted in several ways. The method `window` (and the deprecated function `cut`) extracts the subset of a 'timeSeries' observed between the times `from` and `to`. The methods `head` and `tail` return the first or last part of a time series for a specified length. The method `outlier` can be used to remove huge "outliers" which may appear for example through the redefinition of an index (it should not be used for outlier detection).

There are other functions provided which can be considered in broader sense as subset functions. For example the function

```
fapply(x, from, to, FUN, ...)
```

can be used to subset a time series with very complex rules, e.g. subset Mondays from a series, subset the last Thursdays in every Month, subset from a daily series open (first), high, low, close (last) prices to a end-of-month time series, etc.

As another example, we mention the function `aggregate` which can data records subset to monthly or quarterly information with information specified by the function `FUN`

```
aggregate(x, by = c("monthly", "quarterly"), FUN = colMeans,
          units = NULL, ...)
```

List of Functions:

```
"["      ...,
"<-"     ...,
window   ...,
head     ...,
tail     ...,
outlier  ...,
fapply   ...,
aggreagte ... .
```

Merging and Binding timeSeries Objects

List of Functions:

```
merge  merges ...,
rbind  binds ...,
lag    lags ... .
```

Reordering the columns of timeSeries Objects

List of Functions:

```
orderColnames  ...,
sortColname    ...,
sampleColnames ...,
statsColnames  ...,
pcaColnames    ...,
hclustColnames ... .
```

Renaming column and rows of timeSeries Objects

List of Functions:

```
dim          ...,
dimnames     ...,
colnames<-  ...,
rownames<-  ...,
is.array    ...
```

Mathematical Operations

List of Functions:

```
Ops.timeSeries S3: 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,
```

<code>sign</code>	Returns the signs of a 'timeSeries' object,
<code>diff</code>	Differences a 'timeSeries' object,
<code>scale</code>	Centers and/or scales a 'timeSeries' object,
<code>quantile</code>	Returns quantiles of an univariate 'timeSeries'.

Handling Missing Data

List of Functions:

<code>na.omit</code>	Handles NAs in a timeSeries object,
<code>removeNA</code>	removes NAs from a matrix object,
<code>substituteNA</code>	substitutes NAs by zero, the column mean or median,
<code>interpNA</code>	interpolates NAs using R's "approx" function.

Special Daily and Monthly Operations

Note, a daily time series is not necessarily a series which has only position dates, rather it should mean that we have not more than one record per day. The availability of daily data, e.g. end of day data, is in real time usually at the same local time every working day, but this may be influenced due to holidays or due to the fact that you get the data from a financial center located in another time zone than your own.

If all all these facts can be neglected and only date positions without time information is relevant and we speak loosely of "daily" data records. It is also worth to note, that Rmetrics uses in this case by default the ISO-8601 format "%Y-%m-%d".

In the case of monthly data, it is important to note that Rmetrics always expects the full date for the position vector and not only the years and months. ...

List of Functions:

<code>dummyDailySeries</code>	Creates a dummy daily 'timeSeries' object,
<code>alignDailySeries</code>	Aligns a daily 'timeSeries' to new positions,
<code>rollDailySeries</code>	Rolls daily a 'timeSeries' on a given period,
<code>ohlcDailyPlot</code>	Plots open high low close bar chart,
<code>countMonthlyRecords</code>	Returns a series with monthly counts of records,
<code>isMonthly</code>	Decides if the series consists of monthly records,
<code>rollMonthlyWindows</code>	Returns start and end dates for rolling time windows,
<code>rollMonthlySeries</code>	Rolls monthly a 'timeSeries' on a given period.

Column and Row Statistics

List of Functions:

<code>colStats</code>	...
<code>rowStats</code>	...

Coercion of timeSeries Objects

List of Functions:

```
as
is.timeSeries

as.timeSeries
as.timeSeries.default
as.timeSeries.numeric
as.timeSeries.data.frame
as.timeSeries.matrix
as.timeSeries.ts
as.timeSeries.character
as.timeSeries.zoo

as.vector.timeSeries
as.matrix.timeSeries
as.data.frame.timeSeries
as.ts.timeSeries
```

aggregate-methods *timeSeries Class, Functions and Methods*

Description

Aggregates a 'timeSeries' Object.

Usage

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

Arguments

by	[aggregate] - sequence of <code>timeDate</code> objects denoting the aggregation period.
FUN	the function to be applied.
x	an object of class <code>timeSeries</code> .
...	arguments passed to other methods.

Value

returns an aggregated S4 object of class `timeSeries`.

Examples

```
## Load Microsoft Data Set -
data(MSFT)
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))
aggregate(x, by, mean)

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

align-methods

timeSeries Class, Functions and Methods

Description

Aligns a 'timeSeries' Object.

Usage

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

Arguments

x	an object of class <code>timeSeries</code> .
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.

Value

returns an aligned S4 object of class `timeSeries`.

Examples

```
## Load Microsoft Dataset -
data(MSFT)

## Compute Sample Size -
dim(MSFT)

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

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

 apply

Apply Functions Over Time Series Periods

Description

Apply a function over time series periods of arbitrary positions and lengths.

Usage

```
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

<code>x</code>	an object of class <code>timeSeries</code> .
<code>from, to</code>	starting date and end date as <code>timeDate</code> objects. Note, <code>to</code> must be time ordered after <code>from</code> . If <code>from</code> and <code>to</code> are missing in function <code>fapply</code> they are set by default to <code>from=start(x)</code> , and <code>to=end(x)</code> .
<code>FUN</code>	the function to be applied. For the function <code>applySeries</code> the default setting is <code>FUN=colMeans</code> .
<code>by</code>	a character value either "monthly" or "quarterly" used in the function <code>applySeries</code> . The default value is "monthly". Only operative when both arguments <code>from</code> and <code>to</code> have their default values <code>NULL</code> . In this case the function <code>FUN</code> will be applied to monthly or quarterly periods.
<code>units</code>	an optional character string, which allows to overwrite the current column names of a <code>timeSeries</code> object. By default <code>NULL</code> which means that the column names are selected automatically.
<code>format</code>	the format specification of the input character vector in POSIX notation.
<code>zone</code>	the time zone or financial center where the data were recorded.

<code>FinCenter</code>	a character value with the the location of the financial center named as "continent/city", or "city".
<code>recordIDs</code>	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.
<code>title</code>	an optional title string, if not specified the inputs data name is deparsed.
<code>documentation</code>	optional documentation string, or a vector of character strings.
<code>...</code>	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 -
data(LPP2005REC)
LPP = 100 * LPP2005REC[, c("SBI", "SPI")]
head(LPP, 20)

## Aggregate Quarterly Returns -
applySeries(LPP, by = "quarterly", FUN = colSums)

## Aggregate Quarterly every last Friday in Quarter -
oneDay = 24*3600
from = unique(timeFirstDayInQuarter(time(LPP))) - oneDay
from = timeLastNdayInMonth(from, nday = 5)
to = unique(timeLastDayInQuarter(time(LPP)))
to = timeLastNdayInMonth(to, nday = 5)
data.frame(from = as.character(from), to = as.character(to))
applySeries(LPP, from, to, FUN = colSums)

## Alternative Use -
fapply(LPP, from, to, FUN = colSums)

## Count Trading Days per Month -
colCounts = function(x) rep(NROW(x), times = NCOL(x))
applySeries(LPP, FUN = colCounts, by = "monthly")
```

Description

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

Functions to create 'timeSeries' objects from other objects:

<code>as.timeSeries</code>	Generic function to convert an object to a 'timeSeries' object,
<code>as.timeSeries.default</code>	Returns the unchanged object,
<code>as.timeSeries.numeric</code>	Converts from a numeric vector,
<code>as.timeSeries.data.frame</code>	Converts from a numeric vector,
<code>as.timeSeries.matrix</code>	Converts from a matrix,
<code>as.timeSeries.ts</code>	Converts from an object of class 'ts',
<code>as.timeSeries.character</code>	Converts from a named demo file,
<code>as.timeSeries.zoo</code>	Converts an object of class zoo.

Functions to transform 'timeSeries' objects into other objects:

<code>as.matrix.timeSeries</code>	Coerces a 'timeSeries' to a matrix,
<code>as.data.frame.timeSeries</code>	Coerces a 'timeSeries' to a data.frame,
<code>as.ts.timeSeries</code>	S3: Coerces a 'timeSeries' to a 'ts' object.
<code>as.ts.timeSeries</code>	S3: Coerces a 'timeSeries' to a 'logical' object.

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.

Value

`as.timeSeries`

returns a S4 object of class `timeSeries`.

`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

```
## 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 -
as.matrix(TS)

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

attach

Attach a timeSeries to the search path

Description

Attach a 'timeSeries' object to the search path.

```
attach  attaches a 'timeSeries' object,
detach  detaches a 'timeSeries' object [see base package].
```

Usage

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

Arguments

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

Note

Preliminary, further work has to be done.

Examples

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

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

 base-methods

Methods for 'timeSeries' object

Description

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

var

summary

var

Methods

`x = "timeSeries"` a `timeSeries` object.

Examples

```
## None -
```

 bind

Bind two timeSeries objects

Description

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

Value

returns a S4 object of class `timedate`.

Examples

```
## Load Microsoft Data Set -
data(MSFT)
x = MSFT[1:12, ]

## 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

```
## 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

<code>method</code>	a character string to indicate if geometric (TRUE) or simple (FALSE) returns should be computed.
<code>na.rm</code>	a logical. Should missing values be removed?
<code>x</code>	a time series, may be an object of class "matrix", or "timeSeries".
<code>...</code>	arguments to be passed.

Value

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

Examples

```
## Simulated Return Data -  
x = matrix(rnorm(24), ncol = 2)  
  
## Cumulative Sums Column by Column -  
colCumsums(x)
```

colStats

*Column Statistics***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].
x	a rectangular object which can be transformed into a matrix by the function <code>as.matrix</code> .
...	arguments to be passed.

Value

the functions return a numeric vector of the statistics.

See Also

`link{rowStats}`.

Examples

```
## 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

```
## 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.

Examples

```
## Load Swiss Pension Fund Benchmark -  
data(LPP2005REC)  
  
## Get Description from timeSeries -  
comment(LPP2005REC)
```

`cumulated`*Cumulated Time Series from Returns*

Description

Compute cumulated financial series, 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

<code>method</code>	a character string naming the method how the returns were computed.
<code>percentage</code>	a logical value. By default FALSE, if TRUE the series will be expressed in percentage changes.
<code>x</code>	an object of class <code>timeSeries</code> .
<code>...</code>	arguments to be passed.

Value

all functions return a time series object of the same class as the input argument `x`.

Examples

```
## Load Microsoft Data -  
data(MSFT)  
setRmetricsOptions(myFinCenter = "GMT")  
MSFT <- MSFT[1:10, "Close"]  
MSFT  
  
## Discrete Return Series -  
MSFT = 100 * MSFT/as.numeric(MSFT[1, 1])  
MSFT.RET = returns(MSFT, method = "discrete")  
MSFT.RET  
  
## Cumulative Series, Indexed to 100 -  
cumulated(MSFT.RET, method = "discrete")
```

```
DataPart,timeSeries-method
      DataPart,timeSeries-method
```

Description

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

Examples

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

## Get Data Part -
  getDataPart(X)
```

```
description      Date and User Information
```

Description

Returns data and user string.

Usage

```
description()
```

Examples

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

```
dimnames      Time Series Columns and Rows
```

Description

Functions and methods handling columns and rows of 'timeSeries' objects.

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,

Arguments

value a valid value for names component of `dimnames(x)`.
 x an object of class `timeSeries`.

Value

NA

Examples

```
## Load Swiss Pension Fund Benchmark Data -
data(LPP2005REC)
X = LPP2005REC[1:10, 1:3]

## Get Dimension -
dim(X)

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

## Get Column Names -
colnames(X)

## Get Row Names -
rownames(X)
```

drawdowns

Calculations of Drawdowns

Description

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

The functions are:

<code>drawdowns</code>	Generates 'timeSeries' object of drawdown levels,
<code>drawdownsStats</code>	Compute drawdown stats for univariate time series.

Usage

`drawdowns(x, ...)`

`drawdownsStats(x, ...)`

Arguments

x [`drawdowns`] -
 an uni- or multivariate 'timeSeries' object of financial returns,

[drawdowns] -
an univariate 'timeSeries' object of financial returns.

... [drawdowns] -
optional arguments passed to the function `na.omit`,
[drawdownsStats] -
arguments passed to the function `drawdowns`.

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.

Author(s)

Peter Carl and Sankalp Upadhyay for code from the contributed R package `PerformanceAnalytics`.

Examples

```
## Load Swiss Pension Fund Data Set -
  setRmetricsOptions(myFinCenter = "GMT")
  data(LPP2005REC)
  head(LPP2005REC)

## Compute Drawdowns Statistics -
  drawdownsStats(LPP2005REC[, "SPI"])

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

durations

Durations from a Time Series

Description

Compute durations from an object of class 'timeSeries'.

Usage

```

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

durationSeries(...)

```

Arguments

trim	a logical value. By default TRUE, the first missing observation in the return series will be removed.
units	[durationSeries] - a character value or vector which allows to set the units in which the durations are measured. By default durations are measured in seconds.
x	an object of class <code>timeSeries</code> .
...	arguments to be passed.

Value

returns an object of class `timeSeries`.

Examples

```

## Load Microsoft Data -
data(MSFT)
setRmetricsOptions(myFinCenter = "GMT")
MSFT = MSFT[1:20, "Open"]
MSFT

## Compute Durations in hours, Use Continuous Returns -
durations(MSFT, units = "hours")

```

filter

Linear Filtering on a Time Series

Description

Applies linear filtering to a univariate time series or to each series separately of a multivariate time series.

Value

A `timeSeries` object without missing values.

Examples

```
## Dummy timeSeries
data <- matrix(rnorm(24), ncol = 2)
s <- timeSeries(data, timeCalendar())

## Filter
filter(s, rep(1, 3))
```

`finCenter`*Financial Center of to a timeSeries*

Description

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

Usage

```
## 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

```
## A timeSeries Object -
data <- matrix(rnorm(24), ncol = 2)
charvec <- as.character(timeCalendar())
ts <- timeSeries(data, charvec, FinCenter = "NewYork")
ts

## Print Financial Center -
finCenter(ts)

## Assign New Financial Center -
finCenter(ts) <- "Zurich"
ts
```

`is.timeSeries` *timeSeries Class, Coercion and Transformation*

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 R object.

Value

returns TRUE or FALSE depending on whether its argument is of `timeSeries` type 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. i.e. if a series is a daily, a monthly, or a weekly time series. If the series is regular the frequency of the serie scan determined calling the function `frequency`.

Usage

```
## S4 method for signature 'timeSeries':  
isDaily(x)  
## S4 method for signature 'timeSeries':  
isMonthly(x)  
## S4 method for signature 'timeSeries':  
isQuarterly(x)  
  
## S4 method for signature 'timeSeries':  
isRegular(x)  
  
## S4 method for signature 'timeSeries':  
frequency(x, ...)
```

Arguments

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

Details

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 quarter.

A monthly series is also a daily series, a quarterly series is also a monthly series.

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

```
## None
```

isUnivariate	<i>Checks if a Time Series is Univariate</i>
--------------	--

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)
```

`lag`*Lag a Time Series*

Description

Compute a lagged version of a 'timeSeries' object.

Usage

```
## S4 method for signature 'timeSeries':  
lag(x, k = 1, trim = FALSE, units = NULL, ...)
```

Arguments

<code>k</code>	[lagSeries] - an integer value. The number of lags (in units of observations). By default 1.
<code>trim</code>	a logical value. By default TRUE, the first missing observation in the return series will be removed.
<code>units</code>	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.
<code>x</code>	an object of class timeSeries.
<code>...</code>	arguments passed to other methods.

Value

returns a lagged S4 object of class timeSeries.

Examples

```
## Load Microsoft Data Set -  
x = MSFT[1:20, "Open"]  
  
## Lag the timeSeries Object:  
lag(x, k = -1:1)
```

math

*Mathematical Time Series Operations***Description**

Functions and methods dealing with mathematical 'timeSeries' operations.

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
diff	Differences a 'timeSeries' object,
scale	Centers and/or scales a 'timeSeries' object,
quantile	Returns quantiles of an univariate 'timeSeries'.

Usage

```
## S4 method for signature 'timeSeries':
diff(x, lag = 1, diff = 1, trim = FALSE, pad = NA, ...)
## S4 method for signature 'timeSeries':
scale(x, center = TRUE, scale = TRUE)
## S4 method for signature 'timeSeries':
quantile(x, ...)
```

Arguments

center, scale
 [scale] -
 either a logical value or a numeric vector of length equal to the number of columns of x.

diff
 an integer indicating the order of the difference. By default 1.

lag
 an integer indicating which lag to use. By default 1.

pad
 [diffSeries] -
 which value should get the padded values? By default NA. Another choice often used would be zero.

trim
 a logical value. By default TRUE, the first missing observation in the return series will be removed.

x
 an object of class `timeSeries`.

...
 arguments to be passed.

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

```
## 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
```

model.frame

Model Frames for Time Series Objects

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

an object of class `timeSeries`.

Note

This function is preliminary and untested.

See Also

[model.frame](#).

Examples

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

```
## Extract High's and Low's:
  model.frame( ~ High + Low, data = X)

## 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)")
  Open
```

 monthly

Special Monthly Series

Description

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

countMonthlyRecords	Returns a series with monthly counts of records,
rollMonthlyWindows	Returns start and end dates for rolling time windows,
rollMonthlySeries	Rolls monthly a 'timeSeries' on a given period.

Usage

```
countMonthlyRecords(x)

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

Arguments

by	a character string specifying the rolling shift composed by the length of the shift and its unit, e.g. "3m" represents quarterly shifts.
FUN	the function to be applied. [applySeries] - a function to use for aggregation, by default colAvg.
period	[rollMonthlySeries] - a character string specifying the rolling period composed by the length of the period and its unit, e.g. "12m" represents one year.
x	an object of class timeSeries.
...	arguments passed to other methods.

Value

NA

Examples

```
## Load Microsoft Daily Data Set:
data(MSFT)

## Count Monthly Records -
MSFT.CTS <- countMonthlyRecords(MSFT)
MSFT.CTS
```

na

Handling Missing Time Series Values

Description

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

The functions are listed by topic.

na.omit	Handles NAs,
removeNA	Removes NAs from a matrix object,
substituteNA	substitute NAs by zero, the column mean or median,
interpNA	interpolates NAs using R's "approx" function.

Usage

```
## 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 [na.omit][substituteNA] -
 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 the column median.

method [na.omit] -
 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: <code>method="linear"</code> , <code>method="before"</code> or <code>method="after"</code> . For the interpolation the function <code>approx</code> is used.
object	an object of class("timeSeries").
x	a numeric matrix, or any other object which can be transformed into a matrix through <code>x = as.matrix(x, ...)</code> . If x is a vector, it will be transformed into a one-dimensional matrix.
...	arguments to be passed to the function <code>as.matrix</code> .

Details

Missing Values in Price and Index Series:

Applied to `timeSeries` objects the function `removeNA` just removes rows with NAs from the series. For an interpolation of time series points one can use the function `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.

Missing Values in Return Series:

For return series the function `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.

Note

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

References

Troyanskaya O., Cantor M., Sherlock G., Brown P., Hastie T., Tibshirani R., Botstein D., Altman R.B., (2001); *Missing Value Estimation Methods for DNA microarrays* *Bioinformatics* 17, 520–525.

Examples

```
## Create a Matrix -
X = matrix(rnorm(100), ncol = 5)

## Replace a Single NA Inside -
X[3, 5] = NA

## Replace Three in a Row Inside -
X[17, 2:4] = c(NA, NA, NA)

## Replace Three in a Column Inside -
X[13:15, 4] = c(NA, NA, NA)
```

```
## Replace Two at the Right Border -
  X[11:12, 5] = c(NA, NA)

## Replace One in the Lower Left Corner -
  X[20, 1] = NA
  print(X)

## Remove Rows with NAs -
  removeNA(X)

## Substitute NA's by Zeros or Column Mean -
  substituteNA(X, type = "zeros")
  substituteNA(X, type = "mean")

## Interpolate NA's Linearly -
  interpNA(X, method = "linear")
  # Note the corner missing value cannot be interpolated!

## Take Previous Values in a Column -
  interpNA(X, method = "before")
  # Also here, the corner value is excluded
```

na.contiguous	<i>Find Longest Contiguous Stretch of non-NAs</i>
---------------	---

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

```
## 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

```
## 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	<i>Reorder Column Names of a Time Series</i>
---------------	--

Description

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

orderColnames	Returns ordered column names of a time Series,
sortColnames	Returns sorted column names of a time Series,
sampleColnames	Returns sampled column names of a time Series,
statsColnames	Returns statistically rearranged column names,
pcaColnames	Returns PCA correlation ordered column names,
hclustColnames	Returns hierarchical clustered column names.

Usage

```
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 <code>dist</code> , and the second determines the choice of the agglomeration method, see <code>hclust</code> .
robust	a logical flag which indicates if robust correlations should be used.
x	an object of class <code>timeSeries</code> or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a numeric matrix.
...	further arguments to be passed, see details.

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 the 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 of the default which calculates Pearson'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

```
## Load Swiss Pension Fund Benchmark Data -
  data(LPP2005REC)

## Abbreviate Column Names -
  colnames(LPP2005REC)

## Sort Alphabetically -
  sortColnames(LPP2005REC)

## Sort by Column Names by Hierarchical Clustering -
  hclustColnames(LPP2005REC)
  head(LPP2005REC[, hclustColnames(LPP2005REC)])
```

orderStatistics *order Statistics*

Description

Computes order statistic of a 'timeSeries'.

Usage

```
orderStatistics(x)
```

Arguments

`x` an object of class `timeSeries`.

Value

```
orderStatistics
returns ...
```

Examples

```
## Load Swiss Pension Fund Benchmark Data -
  setRmetricsOptions(myFinCenter = "GMT")
  data(LPP2005REC)

## Compute Order Statistics -
  head(orderStatistics(LPP2005REC))
```

plot-methods

Plot a Time Series

Description

Plot 'timeSeries' objects.

Usage

```
## S4 method for signature 'timeSeries':
plot(x, y, FinCenter = NULL, plot.type =
  c("multiple", "single"), format = "auto", at = "auto", 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, ...)
```

Arguments

`x` an object of class `timeSeries`.
`y` an object of class `timeSeries`.
`FinCenter` a character with the the location of the financial center named as "continent/city".

<code>plot.type</code>	for multivariate time series, should the series be plotted separately (with a common time axis) or on a single plot?
<code>format</code>	...
<code>at</code>	...
<code>widths</code>	...
<code>heights</code>	...
<code>xy.labels</code>	logical, indicating if <code>text()</code> labels should be used for an x-y plot, <code>_or_</code> character, supplying a vector of labels to be used. The default is to label for up to 150 points, and not for more.
<code>xy.lines</code>	logical, indicating if lines should be drawn for an x-y plot. Defaults to the value of <code>xy.labels</code> if that is logical, otherwise to <code>TRUE</code>
<code>panel</code>	a function(<code>x</code> , <code>col</code> , <code>bg</code> , <code>pch</code> , <code>type</code> , ...) which gives the action to be carried out in each panel of the display for <code>plot.type="multiple"</code> . The default is <code>lines</code>
<code>nc</code>	the number of columns to use when <code>type="multiple"</code> . Defaults to 1 for up to 4 series, otherwise to 2.
<code>yax.flip</code>	: logical indicating if the y-axis (ticks and numbering) should flip from side 2 (left) to 4 (right) from series to series when <code>type="multiple"</code> .
<code>mar.multi</code> , <code>oma.multi</code>	the (default) <code>par</code> settings for <code>plot.type="multiple"</code> .
<code>axes</code>	logical indicating if x- and y- axes should be drawn.
...	additional graphical arguments, see <code>plot</code> , <code>plot.default</code> and <code>par</code>

Value

a plot or plot elements of an object of class `timeSeries`.

Examples

```
## 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")

## 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)
finCenter(NYC)
plot(ZRH, 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")

```

print-methods

Print a Time Series

Description

Print 'timeSeries' pbjects.

Arguments

object an object of class timeSeries.

Value

prints an object of class timeSeries.

Examples

```

## Load Micsrosoft 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)
```

rank	<i>Sample Ranks of a Time Series</i>
------	--------------------------------------

Description

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

Value

returns the ranks of a timeSeries object

Examples

```
## Load Microsoft Data -
  data(MSFT)
  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)
```

returns	<i>Financial Returns</i>
---------	--------------------------

Description

Compute financial returns from prices or indexes.

Usage

```
returns(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(...)
```

Arguments

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.
x	an object of class <code>timeSeries</code> .
...	arguments to be passed.

Value

all functions return an object of class `timeSeries`.

Note

The functions `returnSeries`, `getReturns`, are synonymes for `returns.timeSeries`.

Examples

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

## Continuous Returns -
  returns(X)

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

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

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

rowCum

Cumulated Column Statistics

Description

Compute cumulative row Statistics.

Usage

```
## 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

```
## Simulated Monthly Return Data -
X = matrix(rnorm(24), ncol = 2)

## Compute cumulated Sums -
rowCumsums(X)
```

series-methods *Data of a timeSeries object*

Description

`series` returns `Data` slot a `timeSeries` object in a `matrix` form. New series can also be assign to the `timeSeries`.

Usage

```
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
```

SpecialDailySeries *Special Daily Time Series*

Description

Special daily 'timeSeries' functions.

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

Usage

```
dummyDailySeries(x = rnorm(365), units = NULL, zone = "",
  FinCenter = "")
alignDailySeries(x, method = c("before", "after", "interp", "fillNA"),
  include.weekends = FALSE, units = NULL, zone = "",
  FinCenter = "")
rollDailySeries(x, period = "7d", FUN, ...)
ohlcDailyPlot(x, volume = TRUE, colOrder = c(1:5), units = 1e6,
  xlab = c("Date", "Date"), ylab = c("Price", "Volume"),
  main = c("O-H-L-C", "Volume"), grid.nx = 7, grid.lty = "solid", ...)
```

Arguments

`colOrder` `[ohlcDailyPlot]` -
an integer vector which gives the order of the prices and the volume in the input

	object. By default the following order of columns from 1 to 5 is assumed: Open, high, low, close, and volume.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>FUN</code>	the function to be applied. [applySeries] - a function to use for aggregation, by default <code>colAvg</code> s.
<code>grid.lty</code> , <code>grid.nx</code>	[ohlcDailyPlot] - The type of grid line and the number of grid lines used in the plot.
<code>include.weekends</code>	[alignDailySeries] - a logical value. Should weekend dates be included or removed from the series.
<code>main</code>	[ohlcDailyPlot] - a character string to title the price and volume plot.
<code>method</code>	[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".
<code>period</code>	[rollDailySeries] - a character string specifying the rolling period composed by the length of the period and its unit, e.g. "7d" represents one week.
<code>units</code>	[alignDailySeries] - an optional character string, which allows to overwrite the current column names of a <code>timeSeries</code> object. By default <code>NULL</code> which means that the column names are selected automatically. [ohlcDailyPlot] - a numeric value, specifying in which multiples the volume should be referenced on the plot labels. By default <code>1e6</code> , i.e. in units of 1 Million.
<code>volume</code>	[ohlcDailyPlot] - a logical value. Should a volume plot added to the OHLC Plot. By default <code>TRUE</code> .
<code>x</code>	an object of class <code>timeSeries</code> .
<code>xlab</code> , <code>ylab</code>	[ohlcDailyPlot] - two string vectors to name the x and y axis of the price and volume plot.
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	arguments passed to other methods.

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.

`ohlcdailyPlot` displays a Open-High-Low-Close Plot of daily data records.

Examples

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

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

## Align 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)

## Interpolate -
alignDailySeries(tsRet, method = "interp", include.weekend = FALSE)
alignDailySeries(tsRet, method = "interp", include.weekend = TRUE)
```

spreads

Spreads and Mid Quotes

Description

Compute spreads and midquotes from price streams.

Usage

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

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

Arguments

<code>tickSize</code>	the default is <code>NULL</code> to simply compute price changes in original price levels. If <code>tickSize</code> is supplied, the price changes will be divided by the value of <code>inTicksOfSize</code> to compute price changes in ticks.
<code>which</code>	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 <code>c("Bid", "Ask")</code> .
<code>x</code>	an object of class <code>timeSeries</code> .
<code>...</code>	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

```
## 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
```

str-methods

Object Str

Description

Compactly Display the Structure of a 'timeSeries' Object.

Usage

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

Arguments

object an object of class `timeSeries`.
 ... arguments passed to other methods.

Value

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

Examples

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

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

t	<i>timeSeries Transpose</i>
---	-----------------------------

Description

Returns the transpose of a `timeSeries` object.

Usage

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

Arguments

x a `timeSeries` object.

Value

A matrix object.

Examples

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

## transpose
t(s)
```

time *timeSeries, Positions*

Description

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

The functions and methods for the Generation of 'timeSeries' Objects are:

<code>time.timeSeries</code>	Extracts time positions from a 'timeSeries',
<code>time<-.timeSeries</code>	Assign time positions to a 'timeSeries',
<code>sample.timeSeries</code>	Resamples a 'timeSeries' object in time,
<code>sort.timeSeries</code>	Sorts reverts a 'timeSeries' object in time,
<code>rev.timeSeries</code>	Reverts a 'timeSeries' object in time,
<code>start.timeSeries</code>	Extracts start date of a 'timeSeries' object,
<code>end.timeSeries</code>	Extracts end date of a 'timeSeries' object.

Usage

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

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

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

Arguments

<code>decreasing</code>	logical. Should the sort be increasing or decreasing? Not available for partial sorting.
<code>value</code>	a valid value for the component of <code>time(x)</code> .
<code>x</code>	[as] - a <code>matrix</code> type object to be converted. [as.vector][as.matrix][as.data.frame] - [applySeries] - [cut][end][mergeSeries][plot][print][rev][start] - an object of class <code>timeSeries</code> .

... arguments passed to other methods.

Value

timeSeries
read.timeSeries
as.timeSeries
return a S4 object of class `timeSeries`.

seriesData
seriesPositions
extract the `@.Data` and `@position` slots from a `timeSeries` object. Thus, `seriesData` returns an object of class `matrix`, and `seriesPositions` returns an object of class `timeDate`.

is.timeSeries
returns TRUE or FALSE depending on whether its argument is of `timeSeries` type or not.

aggregateSeries
applySeries
cutSeries
mergeSeries
returnSeries
revSeries
return a S4 object of class `timeSeries`.

end, start
return a S4 object of class `timedate`. These are the start and end dates of a `timeSeries` object.

as.vector
as.matrix
as.data.frame
these are methods which convert a S4 object of class `timeSeries` either to a vector, a matrix or to a data frame.

plot
lines
points
print
plot and print methods for an object of class `timeSeries`. Note that the plot function requires the packages `its` and `Hmisc`.

Examples

```
## Create Dummy timeSeries:
X = timeSeries(matrix(rnorm(24), 12), timeCalendar())
```

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

```
timeSeries-deprecated
```

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

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

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

TimeSeriesClass *timeSeries Class*

Description

A collection and description of functions and methods dealing with regular and irregular 'timeSeries' objects. Dates and times are implemented as 'timeDate' objects.

Functions to generate and modify 'timeSeries' objects:

```
timeSeries  Creates a 'timeSeries' object from scratch,
readSeries  Reads a 'timeSeries' from a spreadsheet file.
```

Data Slot and classification of 'timeSeries' objects:

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

Usage

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

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

```
seriesData(object)
```

Arguments

<code>charvec</code>	a character vector of dates and times or any objects which can be coerced to a <code>timeDate</code> object.
<code>data</code>	a matrix object or any objects which can be coerced to a matrix.
<code>documentation</code>	optional documentation string, or a vector of character strings.
<code>file</code>	the filename of a spreadsheet data set from which to import the data records.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>header</code>	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.
<code>format</code>	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.
<code>object</code>	[is][seriesData][seriesPositions][show][summary] - an object of class <code>timeSeries</code> .
<code>recordIDs</code>	a data frame which can be used for record identification information. [print] - a logical value. Should the <code>recordIDs</code> printed together with the data matrix and time series positions?
<code>sep</code>	[readSeries] - the field separator used in the spreadsheet file to separate columns.
<code>title</code>	an optional title string, if not specified the inputs data name is deparsed.
<code>units</code>	an optional character string, which allows to overwrite the current column names of a <code>timeSeries</code> object. By default <code>NULL</code> which means that the column names are selected automatically.
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	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
readSeries
returnSeries
```

return a S4 object of class `timeSeries`.

```
orderStatistics
returns ...
```

```
series
```

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

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

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

## Create a timeSeries Object, The Direct Way ...
Close = MSFT[, 5]
head(Close)

## Create a timeSeries Object from Scratch -
data = as.matrix(MSFT[, 4])
charvec = rownames(MSFT)
Close = timeSeries(data, charvec, units = "Close")
head(Close)
c(start(Close), end(Close))

## Cut out April Data from 2001 -
tsApril01 = window(Close, "2001-04-01", "2001-04-30")
tsApril01

## Compute Continuous Returns -
returns(tsApril01)

## Compute Discrete Returns -
returns(tsApril01, type = "discrete")

## Compute Discrete Returns, Don't trim -
```

```

returns(tsApril01, trim = FALSE)

## Compute Discrete Returns, Use Percentage Values -
tsRet = returns(tsApril01, percentage = TRUE, trim = FALSE)
tsRet

## Aggregate Weekly -
GoodFriday(2001)
to = timeSequence(from = "2001-04-11", length.out = 3, by = "week")
from = to - 6*24*3600
from
to
applySeries(tsRet, from, to, FUN = sum)

## Create large timeSeries objects with different 'charvec' object classes -
# charvec is a 'timeDate' object
head(timeSeries(1:1e6L, timeSequence(length.out = 1e6L, by = "sec")))
head(timeSeries(1:1e6L, seq(Sys.timeDate(), length.out = 1e6L, by = "sec")))
# 'charvec' is a 'POSIXt' object
head(timeSeries(1:1e6L, seq(Sys.time(), length.out = 1e6L, by = "sec")))
# 'charvec' is a 'numeric' object
head(timeSeries(1:1e6L, 1:1e6L))

```

TimeSeriesData

Time Series Data Sets

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 xchange rates.

Examples

```

## 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, type = "l")

```

TimeSeriesSubsettings
Subsetting Time Series

Description

Subset a 'timeSeries' objects 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

```
## 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 = 10, 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 than 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))))  
tS = timeSeries(data, charvec, units = "tS")  
tS  
  
## Subset Series by Counts "[" -  
tS[1:3, ]  
  
## Subset the Head of the Series -  
head(tS, 6)
```

Index

*Topic **chron**

- aggregate-methods, 6
- align-methods, 7
- apply, 8
- as, 10
- attach, 11
- base-methods, 13
- bind, 13
- comment, 16
- cumulated, 17
- DataPart, timeSeries-method, 18
- dimnames, 18
- drawdowns, 19
- durations, 20
- finCenter, 22
- is.timeSeries, 23
- isRegular, 23
- isUnivariate, 25
- lag, 26
- math, 27
- model.frame, 28
- monthly, 29
- orderColnames, 33
- orderStatistics, 34
- plot-methods, 35
- print-methods, 37
- rank, 38
- returns, 38
- SpecialDailySeries, 41
- spreads, 43
- str-methods, 44
- time, 46
- timeSeries-method-stats, 48
- TimeSeriesClass, 49
- TimeSeriesSubsettings, 53

*Topic **datasets**

- TimeSeriesData, 52

*Topic **math**

- na, 30

*Topic **methods**

- aggregate-methods, 6
- align-methods, 7
- math, 27
- timeSeries-method-stats, 48

*Topic **package**

- timeSeries-package, 2

*Topic **programming**

- description, 18

*Topic **univar**

- colCum, 14
- colStats, 15
- rowCum, 39

- +, timeSeries, missing-method
(*math*), 27

- , timeSeries, missing-method
(*math*), 27

- [, timeSeries, ANY, index_timeSeries-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, character, character-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, character, index_timeSeries-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, character, missing-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, index_timeSeries, character-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, index_timeSeries, index_timeSeries-m
(*TimeSeriesSubsettings*), 53

- [, timeSeries, index_timeSeries, missing-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, matrix, index_timeSeries-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, matrix, missing-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, missing, character-method
(*TimeSeriesSubsettings*), 53

- [, timeSeries, missing, index_timeSeries-method

- `cbind`(*bind*), 13
- `cbind2`(*bind*), 13
- `cbind2`, ANY, `timeSeries`-method
(*bind*), 13
- `cbind2`, `timeSeries`, ANY-method
(*bind*), 13
- `cbind2`, `timeSeries`, `missing`-method
(*bind*), 13
- `cbind2`, `timeSeries`, `timeSeries`-method
(*bind*), 13
- `coerce`, ANY, `timeSeries`-method
(*as*), 10
- `coerce`, character, `timeSeries`-method
(*as*), 10
- `coerce`, `data.frame`, `timeSeries`-method
(*as*), 10
- `coerce`, `timeSeries`, `data.frame`-method
(*as*), 10
- `coerce`, `timeSeries`, `list`-method
(*as*), 10
- `coerce`, `timeSeries`, `matrix`-method
(*as*), 10
- `coerce`, `timeSeries`, `ts`-method(*as*),
10
- `coerce`, `timeSeries`, `tse`-method
(*as*), 10
- `coerce`, `ts`, `timeSeries`-method(*as*),
10
- `colAvg`s(*colStats*), 15
- `colCum`, 14
- `colCummax`s(*colCum*), 14
- `colCummax`s, `matrix`-method
(*colCum*), 14
- `colCummax`s, `timeSeries`-method
(*colCum*), 14
- `colCummin`s(*colCum*), 14
- `colCummin`s, `matrix`-method
(*colCum*), 14
- `colCummin`s, `timeSeries`-method
(*colCum*), 14
- `colCumprod`s(*colCum*), 14
- `colCumprod`s, `matrix`-method
(*colCum*), 14
- `colCumprod`s, `timeSeries`-method
(*colCum*), 14
- `colCumreturn`s(*colCum*), 14
- `colCumreturn`s, `matrix`-method
(*colCum*), 14
- `colCumreturn`s, `timeSeries`-method
(*colCum*), 14
- `colCumsum`s(*colCum*), 14
- `colCumsum`s, `matrix`-method
(*colCum*), 14
- `colCumsum`s, `timeSeries`-method
(*colCum*), 14
- `colKurtosis`(*colStats*), 15
- `colMax`s(*colStats*), 15
- `colMean`s, `timeSeries`-method
(*colStats*), 15
- `colMin`s(*colStats*), 15
- `colname`s, `timeSeries`-method
(*dimname*s), 18
- `colname`s<-, `timeSeries`-method
(*dimname*s), 18
- `colProd`s(*colStats*), 15
- `colQuantile`s(*colStats*), 15
- `colSd`s(*colStats*), 15
- `colSkewness`(*colStats*), 15
- `colStats`, 15
- `colStdev`s(*colStats*), 15
- `colSum`s, `timeSeries`-method
(*colStats*), 15
- `colVar`s(*colStats*), 15
- `comment`, 16
- `comment`, `timeSeries`-method
(*comment*), 16
- `comment`<-, `timeSeries`-method
(*comment*), 16
- `cor`, `timeSeries`-method
(*timeSeries-method-stats*),
48
- `cor`-methods
(*timeSeries-method-stats*),
48
- `countMonthlyRecords`(*monthly*), 29
- `cov`, `timeSeries`-method
(*timeSeries-method-stats*),
48
- `cov`-methods
(*timeSeries-method-stats*),
48
- `cummax`, `timeSeries`-method(*math*),
27
- `cummin`, `timeSeries`-method(*math*),
27
- `cumprod`, `timeSeries`-method(*math*),

- 27
- cumsum, timeSeries-method (*math*), 27
- cumulated, 17
- cut, timeSeries-method (*TimeSeriesSubsettings*), 53
- cut.timeSeries (*TimeSeriesSubsettings*), 53
- DataPart, timeSeries-method, 18
- dcauchy, timeSeries-method (*timeSeries-method-stats*), 48
- dcauchy-methods (*timeSeries-method-stats*), 48
- description, 18
- diff, timeSeries-method (*math*), 27
- diff.timeSeries (*math*), 27
- dim, timeSeries-method (*dimnames*), 18
- dim<-, timeSeries-method (*dimnames*), 18
- dimnames, 18
- dimnames, timeSeries-method (*dimnames*), 18
- dimnames<-, timeSeries, list-method (*dimnames*), 18
- dnorm, timeSeries-method (*timeSeries-method-stats*), 48
- dnorm-methods (*timeSeries-method-stats*), 48
- drawdowns, 19
- drawdownsStats (*drawdowns*), 19
- dt, timeSeries-method (*timeSeries-method-stats*), 48
- dt-methods (*timeSeries-method-stats*), 48
- dummyDailySeries (*SpecialDailySeries*), 41
- dummySeries (*SpecialDailySeries*), 41
- durations, 20
- durationSeries (*durations*), 20
- end, timeSeries-method (*time*), 46
- end.timeSeries (*time*), 46
- fapply (*apply*), 8
- filter, 21
- filter, timeSeries-method (*filter*), 21
- finCenter, 22
- finCenter, timeSeries-method (*finCenter*), 22
- finCenter<-, timeSeries-method (*finCenter*), 22
- frequency, timeSeries-method (*isRegular*), 23
- getDataPart, timeSeries-method (*DataPart, timeSeries-method*), 18
- getReturns (*returns*), 38
- hclustColnames (*orderColnames*), 33
- head, timeSeries-method (*TimeSeriesSubsettings*), 53
- head.timeSeries (*TimeSeriesSubsettings*), 53
- index_timeSeries (*TimeSeriesClass*), 49
- index_timeSeries-class (*TimeSeriesClass*), 49
- initialize, timeSeries-method (*TimeSeriesClass*), 49
- interpNA (*na*), 30
- is.na, timeSeries-method (*is.timeSeries*), 23
- is.signalSeries (*is.timeSeries*), 23
- is.timeSeries, 23
- is.unsorted, timeSeries-method (*is.timeSeries*), 23
- isDaily, timeSeries-method (*isRegular*), 23
- isMonthly, timeSeries-method (*isRegular*), 23
- isMultivariate (*isUnivariate*), 25
- isQuarterly, timeSeries-method (*isRegular*), 23
- isRegular, 23
- isRegular, timeSeries-method (*isRegular*), 23

- isUnivariate, [25](#)
- lag, [26](#)
- lag, timeSeries-method (*lag*), [26](#)
- lag.timeSeries (*lag*), [26](#)
- lines, timeSeries-method (*plot-methods*), [35](#)
- log, timeSeries-method (*math*), [27](#)
- LPP2005REC (*TimeSeriesData*), [52](#)
- math, [27](#)
- Math, timeSeries-method (*math*), [27](#)
- Math2, timeSeries-method (*math*), [27](#)
- mean, timeSeries-method (*base-methods*), [13](#)
- merge (*bind*), [13](#)
- merge, ANY, timeSeries-method (*bind*), [13](#)
- merge, matrix, timeSeries-method (*bind*), [13](#)
- merge, numeric, timeSeries-method (*bind*), [13](#)
- merge, timeSeries, ANY-method (*bind*), [13](#)
- merge, timeSeries, matrix-method (*bind*), [13](#)
- merge, timeSeries, missing-method (*bind*), [13](#)
- merge, timeSeries, numeric-method (*bind*), [13](#)
- merge, timeSeries, timeSeries-method (*bind*), [13](#)
- merge.timeSeries (*bind*), [13](#)
- midquotes (*spreads*), [43](#)
- midquoteSeries (*spreads*), [43](#)
- model.frame, [28](#), [28](#)
- monthly, [29](#)
- MSFT (*TimeSeriesData*), [52](#)
- na, [30](#)
- na.contiguous, [32](#)
- na.contiguous, timeSeries-method (*na.contiguous*), [32](#)
- na.omit.timeSeries (*na*), [30](#)
- names, timeSeries-method (*dimnames*), [18](#)
- names<-, timeSeries-method (*dimnames*), [18](#)
- newPositions<- (*timeSeries-deprecated*), [48](#)
- ohlcDailyPlot (*SpecialDailySeries*), [41](#)
- Ops, array, timeSeries-method (*math*), [27](#)
- Ops, timeSeries, array-method (*math*), [27](#)
- Ops, timeSeries, timeSeries-method (*math*), [27](#)
- Ops, timeSeries, ts-method (*math*), [27](#)
- Ops, timeSeries, vector-method (*math*), [27](#)
- Ops, ts, timeSeries-method (*math*), [27](#)
- Ops, vector, timeSeries-method (*math*), [27](#)
- orderColnames, [33](#)
- orderStatistics, [34](#)
- outlier (*TimeSeriesSubsettings*), [53](#)
- outlier, ANY-method (*TimeSeriesSubsettings*), [53](#)
- outlier, timeSeries-method (*TimeSeriesSubsettings*), [53](#)
- pcaColnames (*orderColnames*), [33](#)
- plot (*plot-methods*), [35](#)
- plot, timeSeries-method (*plot-methods*), [35](#)
- plot-methods, [35](#)
- points, timeSeries-method (*plot-methods*), [35](#)
- print, timeSeries-method (*print-methods*), [37](#)
- print-methods, [37](#)
- quantile, timeSeries-method (*math*), [27](#)
- quantile.timeSeries (*math*), [27](#)
- rank, [38](#)
- rank, timeSeries-method (*rank*), [38](#)
- rbind (*bind*), [13](#)
- rbind2 (*bind*), [13](#)
- rbind2, ANY, timeSeries-method (*bind*), [13](#)

- rbind2, timeSeries, ANY-method
(*bind*), 13
- rbind2, timeSeries, missing-method
(*bind*), 13
- rbind2, timeSeries, timeSeries-method
(*bind*), 13
- readSeries (*TimeSeriesClass*), 49
- removeNA (*na*), 30
- returns, 38
- returns, ANY-method (*returns*), 38
- returns, timeSeries-method
(*returns*), 38
- returnSeries (*returns*), 38
- rev, timeSeries-method (*time*), 46
- rev.timeSeries (*time*), 46
- rollDailySeries
(*SpecialDailySeries*), 41
- rollMonthlySeries (*monthly*), 29
- rollMonthlyWindows (*monthly*), 29
- rowCum, 39
- rowCumsums (*rowCum*), 39
- rowCumsums, ANY-method (*rowCum*), 39
- rowCumsums, timeSeries-method
(*rowCum*), 39
- rownames, timeSeries-method
(*dimnames*), 18
- rownames<-, timeSeries, ANY-method
(*dimnames*), 18
- rownames<-, timeSeries, timeDate-method
(*dimnames*), 18

- sample, timeSeries-method (*time*),
46
- sampleColnames (*orderColnames*), 33
- scale, timeSeries-method (*math*), 27
- scale.timeSeries (*math*), 27
- sd, timeSeries-method
(*timeSeries-method-stats*),
48
- sd-methods
(*timeSeries-method-stats*),
48
- series (*series-methods*), 40
- series, timeSeries-method
(*series-methods*), 40
- series-methods, 40
- series<- (*series-methods*), 40
- series<-, timeSeries, ANY-method
(*series-methods*), 40

- series<-, timeSeries, data.frame-method
(*series-methods*), 40
- series<-, timeSeries, matrix-method
(*series-methods*), 40
- series<-, timeSeries, vector-method
(*series-methods*), 40
- seriesData (*TimeSeriesClass*), 49
- seriesPositions
(*timeSeries-deprecated*), 48
- setDataPart, timeSeries-method
(*DataPart, timeSeries-method*),
18
- show, timeSeries-method
(*print-methods*), 37
- sort, timeSeries-method (*time*), 46
- sort.timeSeries (*time*), 46
- sortColnames (*orderColnames*), 33
- SpecialDailySeries, 41
- spreads, 43
- spreadSeries (*spreads*), 43
- start, timeSeries-method (*time*), 46
- start.timeSeries (*time*), 46
- statsColnames (*orderColnames*), 33
- str (*str-methods*), 44
- str, timeSeries-method
(*str-methods*), 44
- str-methods, 44
- substituteNA (*na*), 30
- Summary, timeSeries-method (*math*),
27
- summary, timeSeries-method
(*base-methods*), 13

- t, 45
- t, timeSeries-method (*t*), 45
- tail, timeSeries-method
(*TimeSeriesSubsettings*), 53
- tail.timeSeries
(*TimeSeriesSubsettings*), 53
- time, 46
- time, timeSeries-method (*time*), 46
- time.timeSeries (*time*), 46
- time<- (*time*), 46
- time_timeSeries
(*TimeSeriesClass*), 49
- time_timeSeries-class
(*TimeSeriesClass*), 49
- timeSeries (*TimeSeriesClass*), 49

timeSeries, ANY, ANY-method
 (*TimeSeriesClass*), 49

timeSeries, ANY, missing-method
 (*TimeSeriesClass*), 49

timeSeries, ANY, timeDate-method
 (*TimeSeriesClass*), 49

timeSeries, matrix, ANY-method
 (*TimeSeriesClass*), 49

timeSeries, matrix, missing-method
 (*TimeSeriesClass*), 49

timeSeries, matrix, numeric-method
 (*TimeSeriesClass*), 49

timeSeries, matrix, timeDate-method
 (*TimeSeriesClass*), 49

timeSeries, missing, ANY-method
 (*TimeSeriesClass*), 49

timeSeries, missing, missing-method
 (*TimeSeriesClass*), 49

timeSeries, missing, timeDate-method
 (*TimeSeriesClass*), 49

timeSeries-class
 (*TimeSeriesClass*), 49

timeSeries-deprecated, 48

timeSeries-method-stats, 48

timeSeries-package, 2

TimeSeriesClass, 49

TimeSeriesData, 52

TimeSeriesSubsettings, 53

trunc, timeSeries-method (*math*), 27

USDCHF (*TimeSeriesData*), 52

var, timeSeries-method
 (*timeSeries-method-stats*),
 48

var-methods
 (*timeSeries-method-stats*),
 48

window, timeSeries-method
 (*TimeSeriesSubsettings*), 53

window.timeSeries
 (*TimeSeriesSubsettings*), 53