

# Package ‘timeSeries’

March 22, 2012

**Version** 2160.94

**Revision** 5282

**Date** 2012-03-22

**Title** Rmetrics - Financial Time Series Objects

**Author** Diethelm Wuertz and Yohan Chalabi

**Depends** R (>= 2.10), graphics, grDevices, methods, stats, utils,timeDate (>= 2150.95)

**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.

**LazyData** yes

**License** GPL (>= 2)

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

**Repository** CRAN

**Date/Publication** 2012-03-22 06:58:53

## R topics documented:

|                              |    |
|------------------------------|----|
| timeSeries-package . . . . . | 3  |
| aggregate-methods . . . . .  | 6  |
| align-methods . . . . .      | 7  |
| apply . . . . .              | 8  |
| as . . . . .                 | 9  |
| attach . . . . .             | 11 |

|                                      |    |
|--------------------------------------|----|
| base-methods . . . . .               | 12 |
| bind . . . . .                       | 13 |
| colCum . . . . .                     | 13 |
| colStats . . . . .                   | 14 |
| comment . . . . .                    | 16 |
| cumulated . . . . .                  | 16 |
| DataPart,timeSeries-method . . . . . | 17 |
| description . . . . .                | 18 |
| diff . . . . .                       | 18 |
| dimnames . . . . .                   | 18 |
| drawdowns . . . . .                  | 19 |
| durations . . . . .                  | 20 |
| extract . . . . .                    | 21 |
| filter . . . . .                     | 22 |
| finCenter . . . . .                  | 22 |
| is.timeSeries . . . . .              | 23 |
| isRegular . . . . .                  | 24 |
| isUnivariate . . . . .               | 25 |
| lag . . . . .                        | 26 |
| math . . . . .                       | 27 |
| merge . . . . .                      | 28 |
| model.frame . . . . .                | 28 |
| monthly . . . . .                    | 29 |
| na . . . . .                         | 30 |
| na.contiguous . . . . .              | 32 |
| orderColnames . . . . .              | 33 |
| orderStatistics . . . . .            | 35 |
| periodical . . . . .                 | 35 |
| plot-methods . . . . .               | 36 |
| print-methods . . . . .              | 38 |
| rank . . . . .                       | 39 |
| readSeries . . . . .                 | 39 |
| returns . . . . .                    | 40 |
| rev . . . . .                        | 41 |
| rollMean . . . . .                   | 42 |
| rowCum . . . . .                     | 43 |
| runlengths . . . . .                 | 44 |
| sample . . . . .                     | 45 |
| scale . . . . .                      | 45 |
| series-methods . . . . .             | 45 |
| smooth . . . . .                     | 46 |
| sort . . . . .                       | 47 |
| SpecialDailySeries . . . . .         | 48 |
| splits . . . . .                     | 50 |
| spreads . . . . .                    | 51 |
| start . . . . .                      | 52 |
| str-methods . . . . .                | 53 |
| t . . . . .                          | 53 |

*timeSeries-package* 3

|                                   |    |
|-----------------------------------|----|
| time . . . . .                    | 54 |
| timeSeries-deprecated . . . . .   | 55 |
| timeSeries-method-stats . . . . . | 55 |
| TimeSeriesClass . . . . .         | 56 |
| TimeSeriesData . . . . .          | 59 |
| TimeSeriesSubsettings . . . . .   | 59 |
| turns . . . . .                   | 61 |
| units . . . . .                   | 63 |
| window . . . . .                  | 63 |

**Index** 64

---

timeSeries-package      *Utilities and Tools Package*

---

### 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) 1999-2011 Rmetrics Association  
URL: <http://www.rmetrics.org>

### timeSeries - S4 timeSeries Class

List of Functions:

|              |                                      |
|--------------|--------------------------------------|
| timeSeries   | Creates a 'timeSeries' from scratch, |
| description  | ...,                                 |
| finCenter    | ...,                                 |
| getDataPart  | ...,                                 |
| isReguar     | ...,                                 |
| isUnivariate | ...,                                 |
| series       | ... .                                |

### methods - Time Series Methods

List of Functions:

as      ...,

```

base      ...,
comment  ...,
is       ...,
mathOps  ...,
plot     ...,
show     ...,
stats    ... .

```

#### mathOps - 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,          |
| sign           | Returns the signs of a 'timeSeries' object,              |
| diff           | Differences a 'timeSeries' object,                       |
| scale          | Centers and/or scales a 'timeSeries' object,             |
| quantile       | Returns quantiles of an univariate 'timeSeries'.         |

#### na.omit - List of Functions:

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

### base - Basic Time Series Functions

#### List of Functions:

```

apply      ...,
attach     ...,
cbind      ...,
diff       ...,
dim        ...,
extract    ...,
merge      ...,
rank       ...,
rev        ...,
sample     ...,
scale      ...,
sort       ...,
start      ...,
subset     ...,
t          ...,
colCumsums ...,
colSums    ...,
rowCumsums ... .

```

**fin - Financial Time Series Functions**

List of Functions:

|                      |   |
|----------------------|---|
| align ...,           |   |
| cumulated            | compute cumulated series from a returns,            |
| daily ...,           |   |
| drawdowns            | compute series of drawdowns from financial returns, |
| durations            | compute durations from a financial time series,     |
| miquotes             | compute mid quotes from a price/index stream,       |
| periodicals ...,     |   |
| returns              | Compute returns from prices or indexes,             |
| runlengths ...,      |   |
| smooth ...,          |   |
| splits ...,          |   |
| spreads              | compute spreads from a price/index stream,          |
| turns ... ,          |   |
| orderColnames ...,   |   |
| orderStatistics ..., |   |
| rollMean ...,        |   |

**more ...**

Daily and Monthly - List of Functions:

|                     |   |
|---------------------|---|
| 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,                  |
| countMonthlyRecords | Returns a series with monthly counts of records,      |
| isMonthly           | Decides if the series consists of monthly records,    |
| rollMonthlyWindows  | Returns start and end dates for rolling time windows, |
| rollMonthlySeries   | Rolls monthly a 'timeSeries' on a given period.       |

Column/Row Statistics - List of Functions:

|          |       |
|----------|-------|
| colStats | ...   |
| rowStats | ... . |

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] -<br>sequence of timeDate objects denoting the aggregation period. |
| FUN | the function to be applied.  |
| x   | an object of class timeSeries.   |
| ... | 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",  
                 "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 object of class timeSeries.

## Examples

```
## 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)
```

---

 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

|               |  |
|---------------|--|
| x             | an object of class <code>timeSeries</code> .   |
| from, to      | 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> .  |
| FUN           | the function to be applied. For the function <code>applySeries</code> the default setting is <code>FUN=colMeans</code> .   |
| by            | 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. |
| units         | 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.   |
| 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 <code>data.frame</code> 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 -
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

|                        |   |
|------------------------|---|
| <code>optional</code>  | A logical value. If TRUE, setting row names and converting column names (to syntactic names) is optional. |
| <code>row.names</code> | NULL or a character vector giving the row names for the data frame. Missing values are not allowed.       |
| <code>x</code>         | an object which is coerced according to the generic function.   |
| <code>...</code>       | 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] -<br>alternative way to specify the database to be attached. See for details <code>help(attach, package=base)</code> .   |
| pos            | [attach] -<br>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] -<br>a logical value. If TRUE, 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] -<br>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 | <i>Bind two timeSeries objects</i> |
|------|------------------------------------|

---

**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 | <i>Cumulated Column Statistics</i> |
|--------|------------------------------------|

---

**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

|        |  |
|--------|--|
| 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

```
## 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 | <i>comment for timeSeries objects</i> |
|---------|---------------------------------------|

---

### 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 | <i>Cumulated Time Series from Returns</i> |
|-----------|---|

---

### 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**

|            |  |
|------------|--|
| 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 cumulated 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 time series object of the same class as the input argument x.

**Examples**

```
## 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")

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

---

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 | <i>Creates Date and User Information</i> |
|-------------|--|

---

**Description**

Creates and returns a data and user string.

**Usage**

```
description()
```

**Examples**

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

---

|      |             |
|------|-------------|
| diff | <i>diff</i> |
|------|-------------|

---

**Description**

Description ...

---

|          |                                     |
|----------|-------------------------------------|
| dimnames | <i>Time Series Columns and Rows</i> |
|----------|-------------------------------------|

---

**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.

**Usage**

```
drawdowns(x, ...)
```

```
drawdownsStats(x, ...)
```

**Arguments**

|                  |   |
|------------------|---|
| <code>x</code>   | 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. |
| <code>...</code> | optional arguments passed to the function <code>na.omit</code> .  |

**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.

**Author(s)**

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

**Examples**

```
## 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)
dd = drawdowns(LPP2005REC[, 1:6], main = "Drawdowns")
plot(dd)

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

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

**Description**

Compute durations from an object of class 'timeSeries'.

**Usage**

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

**Arguments**

|       |  |
|-------|--|
| x     | an object of class <code>timeSeries</code> .   |
| 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 `timeSeries` 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**

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

---

extract

*extract*

---

**Description**

Description ...

---

|        |  |
|--------|--|
| filter | <i>Linear Filtering on a Time Series</i> |
|--------|--|

---

**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 | <i>Get and Set Financial Center of a 'timeSeries'</i> |
|-----------|---|

---

**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

getFinCenter(x)
setFinCenter(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

```
## 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

*timeSeries Class, Coercion and Transformation*

---

### Description

is.timeSeries tests if its argument is a timeSeries. is.timeSeries tests if series has no times-tamps.

### 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 | <i>Checks if a time series is regular</i> |
|-----------|---|

---

### 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 | <i>Lag a Time Series</i> |
|-----|--------------------------|

---

**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     | [lagSeries] -<br>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**

```
## 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] -<br>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] -<br>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
```

---

merge

*merge*

---

**Description**

Description ...

---

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 | [nna.omit][substituteNA] -<br>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] -<br>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").   |
| 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

*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**

```

## 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> .                             |
| <code>robust</code> | a logical flag which indicates if robust correlations should be used.   |
| <code>x</code>      | 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. |
| <code>...</code>    | further arguments to be passed, see details.  |

## Details

### Statistically Motivated Rearrangement

The function `statsColnames` rearranges the column names according to a statical measure. These measure must operate on the columns of the time series and return a vector of values which can be sorted. Typical functions ar those listed in in help page `colStats` but one can also crete 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` allsows 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="kendall1"` 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

```
## 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 | <i>order Statistics</i> |
|-----------------|-------------------------|

---

**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))
```

---

|            |                   |
|------------|-------------------|
| periodical | <i>periodical</i> |
|------------|-------------------|

---

**Description**

Description ...

**Description**

Plot 'timeSeries' objects and add lines and points to an already existing chart or graph.

**Usage**

```
## 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**

|                              |  |
|------------------------------|--|
| <code>x,y</code>             | objects of class <code>timeSeries</code> .   |
| <code>FinCenter</code>       | 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 by plotted separately (with a common time axis) or on a single plot?   |
| <code>format</code>          | POSIX label format, e.g. "%Y-%m-%d" or "%F" for ISO-8601 standard date format.   |
| <code>at</code>              | a <code>timeDate</code> object setting the plot label positions. If <code>at=pretty(x)</code> , the positions are generated automatized calling the function <code>pretty</code> . Former default option <code>at="auto"</code> selects 6 equal spaced time label positions. |
| <code>widths, heights</code> | widths and heights for individual graphs, see <code>layout</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>   |

|                      |  |
|----------------------|--|
| 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.  |

**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")
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)
finCenter(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")

```

---

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)
```

---

|            |   |
|------------|---|
| readSeries | <i>Reads a 'timeSeries' from a File</i> |
|------------|---|

---

### Description

Reads a file in table format and creates a timeSeries object from it. The first column of the table must hold the timestamps. Format of the stimestamps can be either specified in the header of the first column or by the format argument.

### Usage

```
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 function assumes that the first column in the csv file has the timestamps.  |
| sep       | the field separator used in the spreadsheet file to separate columns.  |
| zone      | the time zone or financial center where the data were recorded.  |
| ...       | Additional arguments passed to read.table() function which is used to read the file.   |

**Value**

Return a S4 object of class timeSeries.

---

returns

*Financial Returns*

---

**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)
```

---

 rev

---

*Reversion of a 'timeSeries'*


---

**Description**

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

**Usage**

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

**Arguments**

x                    an uni- or multivariate timeSeries object.

**Value**

returns a timeSeries object.

**Examples**

```
## 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**

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

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

## Plot Drawdowns -
rmean <- rollMean(SPI, k = 10)
plot(rmean)
```

---

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)
```

---

|            |                                    |
|------------|------------------------------------|
| runlengths | <i>Runlengths of a Time Series</i> |
|------------|------------------------------------|

---

**Description**

Computes runlengths of an univariate timeSeries object.

**Usage**

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

**Arguments**

x                an univariate time series of class timeSeries.  
 ...              arguments to be passed.

**Value**

returns an object of class timeSeries.

**Examples**

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

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

---

|        |               |
|--------|---------------|
| sample | <i>sample</i> |
|--------|---------------|

---

**Description**

Description ...

---

|       |              |
|-------|--------------|
| scale | <i>scale</i> |
|-------|--------------|

---

**Description**

Description ...

---

|                |   |
|----------------|---|
| series-methods | <i>Get and Set Data of a 'timeSeries'</i> |
|----------------|---|

---

**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
```

---

smooth

*Smooths Time Series Objects*

---

## Description

Smooths 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 `timeSeries` object. They are interfaces to the functions `lowess`, `supsmu` 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**

```
## 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, 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, main = "Close - Spline Smoothed")

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

---

 sort

*Sorting a 'timeSeries' by Time Stamps*


---

**Description**

Sorts a `'timeSeries'` according to increasing or decreasing order of the time stamps.

**Usage**

```
## 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.

**Value**

returns a timeSeries object.

**Examples**

```
## Create Dummy timeSeries -
tS <- dummySeries()

## Sort the Series in Decreasing Order -
sort(tS, decreasing = TRUE)

## Sort the Series -
sort(tS)
```

---

SpecialDailySeries      *Special Daily Time Series*

---

**Description**

Special daily 'timeSeries' functions.

|                  |   |
|------------------|---|
| 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   |

**Usage**

```
dummyDailySeries(x = rnorm(365), units = NULL, zone = "",
  FinCenter = "")
alignDailySeries(x, method = c("before", "after", "interp", "fillNA",
  "fmm", "periodic", "natural", "monoH.FC"),
  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] -<br>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.   |
| FinCenter         | a character with the the location of the financial center named as "continent/city".   |
| FUN               | the function to be applied.<br>[applySeries] -<br>a function to use for aggregation, by default colAvgs.   |
| grid.lty, grid.nx | [ohlcDailyPlot] -<br>The type of grid line and the number of grid lines used in the plot.  |
| include.weekends  | [alignDailySeries] -<br>a logical value. Should weekend dates be included or removed from the series.  |
| main              | [ohlcDailyPlot] -<br>a character string to title the price and volume plot.  |
| method            | [alignDailySeries] -<br>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] -<br>a character string specifying the rolling period composed by the length of the period and its unit, e.g. "7d" represents one week.  |
| units             | [allignDailySeries] -<br>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.<br>[ohlcDailyPlot] -<br>a numeric value, specifying in which multiples the volume should be referenced on the plot labels. By default 1e6, i.e. in units of 1 Million. |
| volume            | [ohlcDailyPlot] -<br>a logical value. Should a volume plot added to the OHLC Plot. By default TRUE.  |
| x                 | an object of class timeSeries.   |
| xlab, ylab        | [ohlcDailyPlot] -<br>two string vectors to name the x and y axis of the price and volume plot.   |
| zone              | the time zone or financial center where the data were recorded.  |
| ...               | arguments passed to interpolating 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**

```
## Use Microsofts' 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

*splits*


---

**Description**

Description ...

---

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 NULL to simply compute price changes in original price levels. If ticksize 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
```

---

|       |  |
|-------|--|
| start | <i>Start and End of a 'timeSeries'</i> |
|-------|--|

---

## Description

Returns start and end time stamps of a 'timeSeries'.

## 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()

## Return Start Time Stamp -
start(tS)

## Return End Time Stamp -
end(tS)
```

---

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

*timeSeries Transpose*

---

### 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

*Get and Set Time stamps of a 'timeSeries'*

---

**Description**

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

**Usage**

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

getTime(x)
setTime(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 timeSeries object.

**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.

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, ...)
```

```
seriesData(object)
```

**Arguments**

|               |  |
|---------------|--|
| 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,<br>[as.timeSeries] -<br>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.<br>[print] -<br>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  
returnSeries

return a S4 object of class timeSeries.

orderStatistics  
returns ...

series

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

### 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] -<br>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] -<br>an integer specifying the number of lines to be returned. By default n=6.   |
| recordIDs  | [head][tail] -<br>a logical value. Should the recordIDs returned together with the data matrix and time series positions?   |
| sd         | [outlierSeries] -<br>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))))
  tS = timeSeries(data, charvec, units = "tS")
  tS

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

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

turns

*Turning Points of a Time Series***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 builtin 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.

**Author(s)**

Frederic Ibanez and Philippe Grosjean for code from the contributed R package `pastecs` and `Rmetrics` for the function wrapper.

**References**

Ibanez, F., 1982, Sur une nouvelle application de la theorie de l'information a la description des series chronologiques planctoniques. *J. Exp. Mar. Biol. Ecol.*, 4, 619–632

Kendall, M.G., 1976, *Time Series*, 2nd ed. Charles Griffin and Co, London.

**Examples**

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

## Cumulate and Smooth the Series -
SPI <- smoothLowess(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 | <i>Get and Set Unit Names of a 'timeSeries'</i> |
|-------|---|

---

**Description**

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

**Usage**

```
getUnits(x)
setUnits(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 <- dummySeries()
tS

## Get the Units -
getUnits(tS)

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

---

|        |               |
|--------|---------------|
| window | <i>window</i> |
|--------|---------------|

---

**Description**

Description ...

# Index

## \*Topic **chron**

- aggregate-methods, 6
- align-methods, 7
- apply, 8
- as, 9
- attach, 11
- base-methods, 12
- bind, 13
- comment, 16
- cumulated, 16
- DataPart, timeSeries-method, 17
- dimnames, 18
- drawdowns, 19
- durations, 20
- is.timeSeries, 23
- isRegular, 24
- isUnivariate, 25
- lag, 26
- math, 27
- model.frame, 28
- monthly, 29
- orderColnames, 33
- orderStatistics, 35
- plot-methods, 36
- print-methods, 38
- rank, 39
- returns, 40
- rev, 41
- rollMean, 42
- runlengths, 44
- smooth, 46
- sort, 47
- SpecialDailySeries, 48
- spreads, 51
- start, 52
- str-methods, 53
- time, 54
- timeSeries-method-stats, 55
- TimeSeriesClass, 56

- TimeSeriesSubsettings, 59

- turns, 61

## \*Topic **datasets**

- TimeSeriesData, 59

## \*Topic **math**

- na, 30

## \*Topic **methods**

- aggregate-methods, 6
- align-methods, 7
- math, 27
- timeSeries-method-stats, 55

## \*Topic **package**

- timeSeries-package, 3

## \*Topic **programming**

- description, 18
- finCenter, 22
- series-methods, 45
- units, 63

## \*Topic **univar**

- colCum, 13
- colStats, 14
- rowCum, 43

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

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

- [, timeSeries, ANY, index\_timeSeries-method (TimeSeriesSubsettings), 59

- [, timeSeries, character, character-method (TimeSeriesSubsettings), 59

- [, timeSeries, character, index\_timeSeries-method (TimeSeriesSubsettings), 59

- [, timeSeries, character, missing-method (TimeSeriesSubsettings), 59

- [, timeSeries, index\_timeSeries, character-method (TimeSeriesSubsettings), 59

- [, timeSeries, index\_timeSeries, index\_timeSeries-method (TimeSeriesSubsettings), 59

- [, timeSeries, index\_timeSeries, missing-method (TimeSeriesSubsettings), 59

- [, timeSeries, matrix, index\_timeSeries-method

- (TimeSeriesSubsettings), 59
- [, timeSeries, matrix, missing-method (TimeSeriesSubsettings), 59
- [, timeSeries, missing, character-method (TimeSeriesSubsettings), 59
- [, timeSeries, missing, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [, timeSeries, missing, missing-method (TimeSeriesSubsettings), 59
- [, timeSeries, timeDate, character-method (TimeSeriesSubsettings), 59
- [, timeSeries, timeDate, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [, timeSeries, timeDate, missing-method (TimeSeriesSubsettings), 59
- [, timeSeries, timeSeries, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [, timeSeries, timeSeries, missing-method (TimeSeriesSubsettings), 59
- [, timeSeries, time\_timeSeries, ANY-method (TimeSeriesSubsettings), 59
- [, timeSeries, time\_timeSeries, character-method (TimeSeriesSubsettings), 59
- [, timeSeries, time\_timeSeries, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [, timeSeries, time\_timeSeries, missing-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, character, ANY-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, character, character-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, character, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, character, missing-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, index\_timeSeries, character-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, matrix, character-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeDate, ANY-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeDate, character-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeDate, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeDate, missing-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeSeries, character-method (TimeSeriesSubsettings), 59
- (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeSeries, index\_timeSeries-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, timeSeries, missing-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, ANY, ANY-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, ANY, factor-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, ANY, numeric-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, ANY-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, factor-method (TimeSeriesSubsettings), 59
- [<-, timeSeries, numeric-method (TimeSeriesSubsettings), 59
- %\*%, ANY, timeSeries-method (math), 27
- %\*%, timeSeries, ANY-method (math), 27
- %\*%, timeSeries, vector-method (math), 27
- aggregate (aggregate-methods), 6
- aggregate, timeSeries-method (aggregate-methods), 6
- aggregate-methods, 6
- aggregate.timeSeries (aggregate-methods), 6
- align, timeSeries-method (align-methods), 7
- align-methods, 7
- alignDailySeries (SpecialDailySeries), 48
- apply, 8
- apply, timeSeries-method (apply), 8
- applySeries (apply), 8
- as, 9
- attach, 11
- attach, timeSeries-method (attach), 11
- base-methods, 12
- bind, 13
- 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), 9
- coerce, character, timeSeries-method (as), 9
- coerce, data.frame, timeSeries-method (as), 9
- coerce, timeSeries, data.frame-method (as), 9
- coerce, timeSeries, list-method (as), 9
- coerce, timeSeries, matrix-method (as), 9
- coerce, timeSeries, ts-method (as), 9
- coerce, timeSeries, tse-method (as), 9
- coerce, ts, timeSeries-method (as), 9
- colAvg (colStats), 14
- colCum, 13
- colCummax (colCum), 13
- colCummax, matrix-method (colCum), 13
- colCummax, timeSeries-method (colCum), 13
- colCummins (colCum), 13
- colCummins, matrix-method (colCum), 13
- colCummins, timeSeries-method (colCum), 13
- colCumprods (colCum), 13
- colCumprods, matrix-method (colCum), 13
- colCumprods, timeSeries-method (colCum), 13
- colCumreturns (colCum), 13
- colCumreturns, matrix-method (colCum), 13
- colCumreturns, timeSeries-method (colCum), 13
- colCumsums (colCum), 13
- colCumsums, matrix-method (colCum), 13
- colCumsums, timeSeries-method (colCum), 13
- colKurtosis (colStats), 14
- colMaxs (colStats), 14
- colMeans, timeSeries-method (colStats), 14
- colMins (colStats), 14
- colnames, timeSeries-method (dimnames), 18
- colnames<-, timeSeries-method (dimnames), 18
- colProds (colStats), 14
- colQuantiles (colStats), 14
- colSds (colStats), 14
- colSkewness (colStats), 14
- colStats, 14
- colStdevs (colStats), 14
- colSums, timeSeries-method (colStats), 14
- colVars (colStats), 14
- comment, 16
- comment, timeSeries-method (comment), 16
- comment<-, timeSeries-method (comment), 16
- cor, timeSeries-method (timeSeries-method-stats), 55
- cor-methods (timeSeries-method-stats), 55
- countMonthlyRecords (monthly), 29
- cov, timeSeries-method (timeSeries-method-stats), 55
- cov-methods (timeSeries-method-stats), 55
- cummax, timeSeries-method (math), 27
- cummin, timeSeries-method (math), 27
- cumprod, timeSeries-method (math), 27
- cumsum, timeSeries-method (math), 27
- cumulated, 16
- cut, timeSeries-method (TimeSeriesSubsettings), 59
- cut.timeSeries (TimeSeriesSubsettings), 59
- DataPart, timeSeries-method, 17
- dcauchy, timeSeries-method (timeSeries-method-stats), 55
- dcauchy-methods (timeSeries-method-stats), 55
- description, 18
- diff, 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), 55

- dnorm-methods
  - (timeSeries-method-stats), 55
- drawdowns, 19
- drawdownsStats (drawdowns), 19
- dt, timeSeries-method
  - (timeSeries-method-stats), 55
- dt-methods (timeSeries-method-stats), 55
- dummyDailySeries (SpecialDailySeries), 48
- dummySeries (SpecialDailySeries), 48
- durations, 20
- durationSeries (durations), 20
- end, timeSeries-method (start), 52
- end.timeSeries (start), 52
- extract, 21
- fapply (apply), 8
- filter, 22
- filter, timeSeries-method (filter), 22
- finCenter, 22
- finCenter, timeSeries-method
  - (finCenter), 22
- finCenter<-, timeSeries-method
  - (finCenter), 22
- frequency, timeSeries-method
  - (isRegular), 24
- getDataPart, timeSeries-method
  - (DataPart, timeSeries-method), 17
- getFinCenter (finCenter), 22
- getReturns (returns), 40
- getTime (time), 54
- getUnits (units), 63
- hclustColnames (orderColnames), 33
- head, timeSeries-method
  - (TimeSeriesSubsettings), 59
- head.timeSeries
  - (TimeSeriesSubsettings), 59
- index\_timeSeries (TimeSeriesClass), 56
- index\_timeSeries-class
  - (TimeSeriesClass), 56
- initialize, timeSeries-method
  - (TimeSeriesClass), 56
- 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), 24
- isMonthly, timeSeries-method
  - (isRegular), 24
- isMultivariate (isUnivariate), 25
- isQuarterly, timeSeries-method
  - (isRegular), 24
- isRegular, 24
- isRegular, timeSeries-method
  - (isRegular), 24
- isUnivariate, 25
- lag, 26
- lag, timeSeries-method (lag), 26
- lag.timeSeries (lag), 26
- lines, timeSeries-method (plot-methods), 36
- log, timeSeries-method (math), 27
- LPP2005REC (TimeSeriesData), 59
- math, 27
- Math, timeSeries-method (math), 27
- Math2, timeSeries-method (math), 27
- mean, timeSeries-method (base-methods), 12
- merge, 28
- 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), 51
- midquoteSeries (spreads), 51
- model.frame, 28, 29

- monthly, 29
- MSFT (TimeSeriesData), 59
- na, 30
- na.contiguous, 32
- na.contiguous, timeSeries-method (na.contiguous), 32
- names, timeSeries-method (dimnames), 18
- names<-, timeSeries-method (dimnames), 18
- newPositions<- (timeSeries-deprecated), 55
- ohlcdailyPlot (SpecialDailySeries), 48
- 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, 35
- outlier (TimeSeriesSubsettings), 59
- outlier, ANY-method (TimeSeriesSubsettings), 59
- outlier, timeSeries-method (TimeSeriesSubsettings), 59
- pcaColnames (orderColnames), 33
- periodical, 35
- plot (plot-methods), 36
- plot, timeSeries-method (plot-methods), 36
- plot-methods, 36
- points, timeSeries-method (plot-methods), 36
- pretty.timeSeries (plot-methods), 36
- print, timeSeries-method (print-methods), 38
- print-methods, 38
- quantile, timeSeries-method (math), 27
- quantile.timeSeries (math), 27
- rank, 39
- rank, timeSeries-method (rank), 39
- 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, 39
- removeNA (na), 30
- returns, 40
- returns, ANY-method (returns), 40
- returns, timeSeries-method (returns), 40
- returnSeries (returns), 40
- rev, 41
- rev, timeSeries-method (rev), 41
- rev.timeSeries (rev), 41
- rollDailySeries (SpecialDailySeries), 48
- rollMax (rollMean), 42
- rollMean, 42
- rollMedian (rollMean), 42
- rollMin (rollMean), 42
- rollMonthlySeries (monthly), 29
- rollMonthlyWindows (monthly), 29
- rollStats (rollMean), 42
- rowCum, 43
- rowCumsums (rowCum), 43
- rowCumsums, ANY-method (rowCum), 43
- rowCumsums, timeSeries-method (rowCum), 43
- rownames, timeSeries-method (dimnames), 18
- rownames<-, timeSeries, ANY-method (dimnames), 18
- rownames<-, timeSeries, timeDate-method (dimnames), 18
- runlengths, 44
- sample, 45
- sample, timeSeries-method (time), 54
- sampleColnames (orderColnames), 33
- scale, 45
- scale, timeSeries-method (math), 27
- scale.timeSeries (math), 27
- sd, timeSeries-method (timeSeries-method-stats), 55
- sd-methods (timeSeries-method-stats), 55
- series (series-methods), 45
- series, timeSeries-method (series-methods), 45
- series-methods, 45

- series<- (series-methods), 45
- series<-, timeSeries, ANY-method  
(series-methods), 45
- series<-, timeSeries, data.frame-method  
(series-methods), 45
- series<-, timeSeries, matrix-method  
(series-methods), 45
- series<-, timeSeries, vector-method  
(series-methods), 45
- seriesData (TimeSeriesClass), 56
- seriesPositions  
(timeSeries-deprecated), 55
- setDataPart, timeSeries-method  
(DataPart, timeSeries-method),  
17
- setFinCenter<- (finCenter), 22
- setTime<- (time), 54
- setUnits<- (units), 63
- show, timeSeries-method (print-methods),  
38
- smooth, 46
- smoothLowess (smooth), 46
- smoothSpline (smooth), 46
- smoothSupsmu (smooth), 46
- sort, 47
- sort, timeSeries-method (sort), 47
- sort.timeSeries (sort), 47
- sortColnames (orderColnames), 33
- SpecialDailySeries, 48
- splits, 50
- spreads, 51
- spreadSeries (spreads), 51
- start, 52
- start, timeSeries-method (start), 52
- start.timeSeries (start), 52
- statsColnames (orderColnames), 33
- str (str-methods), 53
- str, timeSeries-method (str-methods), 53
- str-methods, 53
- substituteNA (na), 30
- Summary, timeSeries-method (math), 27
- summary, timeSeries-method  
(base-methods), 12
  
- t, 53
- t, timeSeries-method (t), 53
- tail, timeSeries-method  
(TimeSeriesSubsettings), 59
  
- tail.timeSeries  
(TimeSeriesSubsettings), 59
- time, 54
- time, timeSeries-method (time), 54
- time.timeSeries (time), 54
- time<- (time), 54
- time\_timeSeries (TimeSeriesClass), 56
- time\_timeSeries-class  
(TimeSeriesClass), 56
- timeSeries (TimeSeriesClass), 56
- timeSeries, ANY, ANY-method  
(TimeSeriesClass), 56
- timeSeries, ANY, missing-method  
(TimeSeriesClass), 56
- timeSeries, ANY, timeDate-method  
(TimeSeriesClass), 56
- timeSeries, matrix, ANY-method  
(TimeSeriesClass), 56
- timeSeries, matrix, missing-method  
(TimeSeriesClass), 56
- timeSeries, matrix, numeric-method  
(TimeSeriesClass), 56
- timeSeries, matrix, timeDate-method  
(TimeSeriesClass), 56
- timeSeries, missing, ANY-method  
(TimeSeriesClass), 56
- timeSeries, missing, missing-method  
(TimeSeriesClass), 56
- timeSeries, missing, timeDate-method  
(TimeSeriesClass), 56
- timeSeries-class (TimeSeriesClass), 56
- timeSeries-deprecated, 55
- timeSeries-method-stats, 55
- timeSeries-package, 3
- TimeSeriesClass, 56
- TimeSeriesData, 59
- TimeSeriesSubsettings, 59
- trunc, timeSeries-method (math), 27
- turns, 61
- turnsStats (turns), 61
  
- units, 63
- USDCHF (TimeSeriesData), 59
  
- var, timeSeries-method  
(timeSeries-method-stats), 55
- var-methods (timeSeries-method-stats),  
55

window, [63](#)  
window, timeSeries-method  
    (TimeSeriesSubsettings), [59](#)  
window.timeSeries  
    (TimeSeriesSubsettings), [59](#)