

Package ‘fSeries’

May 25, 2009

Version 270.76.3

Revision 4192

Date 2009-05-25

Title Financial Time Series Objects

Author Diethelm Wuertz and many others, see the SOURCE file

Depends R (>= 2.4.0), robustbase, methods, fUtilities, fEcofin, fCalendar

Suggests RUnit

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

Description Environment for teaching “Financial Engineering and Computational Finance”

NOTE SEVERAL PARTS ARE STILL PRELIMINARY AND MAY BE CHANGED IN THE FUTURE. THIS TYPICALLY INCLUDES FUNCTION AND ARGUMENT NAMES, AS WELL AS DEFAULTS FOR ARGUMENTS AND RETURN VALUES.

LazyLoad yes

LazyData yes

License GPL (>= 2)

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

Repository CRAN

Date/Publication 2009-05-25 20:24:26

R topics documented:

aggregate	2
apply	3
as	4
attach	6
bind	7
colCum	8
cor	9
dim	10
durations	11
lag	12
math	13
merge	15
model.fram	16
monthly	17
na	18
orderColnames	20
plot	22
print	23
returns	24
rowCum	25
SpecialDailySeries	26
spreads	28
summary	29
time	30
TimeSeriesClass	32
TimeSeriesSubsettings	37
Index	40

 aggregate

timeSeries Class, Functions and Methods

Description

Computes Summary Statistics of Data Subsets

Usage

```
## S3 method for class 'timeSeries':
aggregate(x, by = c("monthly", "quarterly"),
          FUN = colMeans, units = NULL, ...)
```

Arguments

<code>by</code>	[aggregate] - a character string denoting the aggregation period, either "monthly" or "quarterly".
<code>FUN</code>	the function to be applied.
<code>units</code>	an optional character string, which allows to overwrite the current column names of a <code>timeSeries</code> object. By default <code>NULL</code> which means that the column names are selected automatically.
<code>x</code>	an object of class <code>timeSeries</code> .
<code>...</code>	arguments passed to other methods.

Value

returns an aggregated S4 object of class `timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
  x = as.timeSeries(data(msft.dat))

## aggregate -
  aggregate(x)
```

apply

Applies Functions Over timeSeries Margins

Description

Applies Functions Over timeSeries Margins

Usage

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

Arguments

<code>from, to</code>	starting date and end date, <code>to</code> must be after <code>from</code> .
<code>FUN</code>	the function to be applied.
<code>x</code>	an object of class <code>timeSeries</code> .
<code>...</code>	arguments passed to other methods.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
```

as *timeSeries Class, Coercion and Transformation*

Description

A collection and description of 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 a 'timeSeries' object,
<code>as.timeSeries.default</code>	Returns unchanged the 'timeSeries' 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.vector.timeSeries</code>	Coerces a 'timeSeries' to a vector,
<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.

Usage

```
is.timeSeries(object)
## S3 method for class 'numeric':
as.timeSeries(x, ...)
## S3 method for class 'data.frame':
as.timeSeries(x, ...)
## S3 method for class 'matrix':
as.timeSeries(x, ...)
## S3 method for class 'ts':
as.timeSeries(x, ...)
## S3 method for class 'character':
```

```

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

## S3 method for class 'timeSeries':
as.vector(x, mode = "any")
## S3 method for class 'timeSeries':
as.matrix(x, ...)
## S3 method for class 'timeSeries':
as.data.frame(x, row.names = NULL, optional = NULL, ...)
## S3 method for class 'timeSeries':
as.ts(x, ...)

```

Arguments

mode	a character string giving an atomic mode or "list", or (not for 'vector') "any".
object	an object of class <code>timeSeries</code> .
optional	A logical value. If TRUE, setting row names and converting column names (to syntactic names) is optional.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
x	an object which is coerced according to the generic function.
...	arguments passed to other methods.

Value

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

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

`as.vector`
`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`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data - timeSeries:
# Create an artificial timeSeries object:
myFinCenter <- "GMT"
charvec = timeCalendar()
data = matrix(rnorm(12))
TS = timeSeries(data, charvec, units = "RAND")
TS

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

## As Vector:
as.vector(TS)

## As Matrix or Data Frame:
as.matrix(TS)
as.data.frame(TS)

## As Univariate Object of Class 'ts':
as.ts(TS)
```

attach

Attach a timeSeries to the search path

Description

A collection and description of functions and methods dealing with the attachment of timeSeries objects to the search path.

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

Usage

```
## S3 method for class 'timeSeries':
attach(what, pos = 2, name = deparse(substitute(what)),
       warn.conflicts = TRUE)
```

Arguments

`name` [attach] -
alternative way to specify the database to be attached. See for details `help(attach, package=base)`

`pos` [attach] -
an integer specifying position in `search()` where to attach the database. See for details `help(attach, package=base)`.

```
warn.conflicts
    [attach] -
    a logical value. If TRUE, warnings are printed about conflicts from attaching the
    database, unless that database contains an object .conflicts.OK. A conflict
    is a function masking a function, or a non-function masking a non-function. See
    for details help(attach, package=base).

what
    [attach] -
    database to be attached. This may currently be a timeSeries object, a data.frame
    or a list or a R data file created with save or NULL or an environment. See for
    details help(attach, package=base).
```

Note

Preliminary, further work has to be done.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
  x = as.timeSeries(data(msft.dat))[1:10, ]

## attach -
  attach(x)
  High - Low
```

 bind

Bind two timeSeries objects

Description

Binds two timeSeries objects either by column or row.

Usage

```
## S3 method for class 'timeSeries':
cbind(x, y, units = NULL)
## S3 method for class 'timeSeries':
rbind(x, y, units = NULL)
```

Arguments

```
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, y      two objects of class timeSeries.
```

Value

returns a S4 object of class `timedate`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
  x = as.timeSeries(data(msft.dat))[1:12, ]

## cbind -
  cbind(x[, "Open"], returnSeries(x[, "Open"], units = c("Open", "Return")))

## rbind -
  rbind(x[1:3, "Open"], x[10:12, "Open"])
```

colCum

Cumulated Column Statistics

Description

Functions to compute cumulative column statistics.

Usage

```
## Default S3 method:
colCumsums(x, na.rm = FALSE, ...)
## S3 method for class 'timeSeries':
colCumsums(x, na.rm = FALSE, ...)

## Default S3 method:
colCummaxs(x, na.rm = FALSE, ...)
## S3 method for class 'timeSeries':
colCummaxs(x, na.rm = FALSE, ...)

## Default S3 method:
colCumprods(x, na.rm = FALSE, ...)
## S3 method for class 'timeSeries':
colCumprods(x, na.rm = FALSE, ...)

## Default S3 method:
colCumreturns(x, method = c("geometric", "simple"), na.rm = FALSE, ...)
## S3 method for class 'timeSeries':
colCumreturns(x, method = c("geometric", "simple"), na.rm = FALSE, ...)

## S3 method for class 'timeSeries':
cumsum(x)
```

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", "timeSeries", or "zoo".

... arguments to be passed.

Value

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

Note

The method `cumsum.timeSeries` will be no longer supported use instead `colCumsums`.

The methods for the "zoo" methods are still preliminary and untested. `na.rm` is not yet supported in all case, please test.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## Simulated Monthly Return Data:
x = matrix(rnorm(24), ncol = 2)

## colStats -
colCumsums(x)
```

cor

timeSeries Correlations

Description

A collection and description of functions and methods dealing with correlations between 'timeSeries' objects.

`cov` Computes Covariance from a 'timeSeries' object,
`cor` Computes Correlations from a 'timeSeries' object.

Usage

```
## S3 method for class 'timeSeries':
cov(x, y = NULL, use = "all.obs",
```

```

method = c("pearson", "kendall", "spearman")

## S3 method for class '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 <code>timeSeries</code> .
y	NULL (default) or a <code>timeSeries</code> object with compatible dimensions to x. The default is equivalent to <code>y = x</code> (but more efficient).

Value

returns the covariance or correlation matrix.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```

## data -
x = as.timeSeries(data(msft.dat))[, 1:4]
x = 100*returnSeries(x)

## cov -
cov(x[, "Open"], x[, "Close"])
cov(x)

```

dim

timeSeries Columns and Rows

Description

A collection and description of functions and methods dealing with 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,
is.array	Allows that NCOL and NROW work properly.

Usage

```
## S3 method for class 'timeSeries':
dim(x)
## S3 method for class 'timeSeries':
dimnames(x)

colnames<-.timeSeries(x) <- value
rownames<-.timeSeries(x) <- value

## S3 method for class 'timeSeries':
is.array(x)
```

Arguments

value	a valid value for column names component of <code>dimnames(x)</code> . For a "timeSeries" object this is either <code>NULL</code> or a character vector of length the column dimension. Not, row names cannot be assigne for a "timeSeries" object, the function <code>rownames()</code> will stop and return an error message.
x	an object of class <code>timeSeries</code> .

Value

NA

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## NYI
```

durations

Computes durations from a timeSeries

Description

Computes durations from an object of class 'timeSeries'.

Usage

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

Arguments

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

Value

returns an object of class `timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -  
# Microsoft Data:  
myFinCenter <- "GMT"  
MSFT = as.timeSeries(data(msft.dat)) [1:20, "Open"]  
head(MSFT)  
  
## durations -  
# Durations in hours: Continuous Returns:  
durations(MSFT, units = "hours")
```

lag

Lag a timeSeries Object

Description

Computes a lagged version of a `timeSeries` object.

Usage

```
## S3 method for class 'timeSeries':  
lag(x, k = 1, trim = FALSE, units = NULL, ...)
```

Arguments

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

Value

returns a lagged S4 object of class `timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
  x = as.timeSeries(data(msft.dat))[1:20, "Open"]

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

 math

Mathematical timeSeries Operations

Description

A collection and description of functions and methods dealing with mathematical `timeSeries` operations .

<code>Ops.timeSeries</code>	S3: Arith method for a 'timeSeries' object,
<code>abs</code>	Returns absolute values of a 'timeSeries' object,
<code>sqrt</code>	Returns square root of a 'timeSeries' object,
<code>exp</code>	Returns the exponential values of a 'timeSeries' object,
<code>log</code>	Returns the logarithm of a 'timeSeries' object,
<code>sign</code>	Returns the signs of a 'timeSeries' object,
<code>diff</code>	Differences a 'timeSeries' object,
<code>scale</code>	Centers and/or scales a 'timeSeries' object,
<code>quantile</code>	Returns quantiles of an univariate 'timeSeries'.

Usage

```
## S3 method for class 'timeSeries':
Ops(e1, e2)
## S3 method for class 'timeSeries':
abs(x)
## S3 method for class 'timeSeries':
sqrt(x)
## S3 method for class 'timeSeries':
exp(x)
## S3 method for class 'timeSeries':
log(x, base = exp(1))
## S3 method for class 'timeSeries':
sign(x)
## S3 method for class 'timeSeries':
diff(x, lag = 1, diff = 1, trim = FALSE, pad = NA, ...)
## S3 method for class 'timeSeries':
scale(x, center = TRUE, scale = TRUE)
## S3 method for class 'timeSeries':
quantile(x, ...)
```

Arguments

base	[log] - a positive number. The base with respect to which logarithms are computed. Defaults to $e = \exp(1)$.
center, scale	[scale] - either a logical value or a numeric vector of length equal to the number of columns of <i>x</i> .
diff	an integer indicating the order of the difference. By default 1.
e1, e2	[Ops] - two objects of class <code>timeSeries</code> .
lag	an integer indicating which lag to use. By default 1.
pad	[diffSeries] - which value should get the padded values? By default <code>NA</code> . Another choice often used would be zero.
trim	a logical value. By default <code>TRUE</code> , the first missing observation in the return series will be removed.
x	an object of class <code>timeSeries</code> .
...	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.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
# Create an artificial timeSeries object:
myFinCenter <- "GMT"
charvec = timeCalendar()
set.seed(4711)
data = matrix(exp(cumsum(rnorm(12, sd = 0.1))))
TS = timeSeries(data, charvec, units = "TS")
TS

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

merge

Merge two timeSeries objects

Description

Merges two timeSeries objects.

Usage

```
## S3 method for class 'timeSeries':
merge(x, y, units = NULL, ...)
```

Arguments

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, y	two objects of class timeSeries.
...	arguments passed to other methods.

Value

returns an S4 object of class timeSeries.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -  
x = as.timeSeries(data(msft.dat))[1:20, "Open"]  
  
## aggregate -  
merge(x, returnSeries(x))
```

model.frame

Model Frames for timeSeries Objects

Description

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

Usage

```
## S3 method for class 'timeSeries':  
model.frame(formula, data, ...)
```

Arguments

formula	a model formula object.
data	an object of class <code>timeSeries</code> .
...	arguments passed to the function <code>stats::model.frame</code> .

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.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

[model.frame](#).

Examples

```
## data -
# Microsoft Data:
myFinCenter <- "GMT"
MSFT = as.timeSeries(data(msft.dat)) [1:12, ]

## model.frame -
# Extract High's and Low's:
model.frame( ~ High + Low, data = MSFT)
# Extract Open Prices and their log10's:
base = 10
Open = model.frame(Open ~ log(Open, base = `base`), data = MSFT)
colnames(Open) <- c("MSFT", "log10(MSFT)")
Open
```

monthly

*Special Monthly Series***Description**

A collection and description of functions and methods dealing with special monthly 'timeSeries' objects.

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.

Usage

```
countMonthlyRecords(x)
isMonthly(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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
# Microsoft Daily Data Set:
x = as.timeSeries(data(msft.dat))
countMonthlyRecords(x)
isMonthly(x)

## data -
# EDHEC Hedge Funds Monthly Data Set
x = as.timeSeries(data(edhec.tS))
isMonthly(x)
```

 na

Handling Missing Values

Description

A collection and description of 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.

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

Usage

```
## S3 method for class '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] - Three alternative methods are provided to remove NAs from the data: `type="zeros"` replaces the missing values by zeros, `type="mean"` replaces the missing values by the column mean, `type="median"` replaces the missing values by the column median.

`method` [na.omit] - Specifies the method how to handle NAs. One of the applied vector strings: `method="s"` `na.rm = FALSE`, skip, i.e. do nothing, `method="r"` remove NAs, `method="z"` substitute NAs by zeros, `method="ir"` interpolate NAs and remove NAs at the beginning and end of the series, `method="iz"` interpolate NAs and substitute NAs at the beginning and end of the series, `method="ie"` interpolate NAs and extrapolate NAs at the beginning and end of the series, [interpNA] - Specifies the method how to interpolate the matrix column by column. One of the applied vector strings: `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 `x = as.matrix(x, ...)`. If `x` is a vector, it will be transformed into a one-dimensional matrix.

`...` arguments to be passed to the function `as.matrix`.

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`.

Author(s)

Raphael Gottardo for the `knn` function,
Diethelm Wuertz for the Rmetrics R-port.

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 with NAs:
X = matrix(rnorm(100), ncol = 5)
# a single NA inside:
X[3, 5] = NA
# three in a row inside:
X[17, 2:4] = c(NA, NA, NA)
# three in a column inside:
X[13:15, 4] = c(NA, NA, NA)
# two at the right border:
X[11:12, 5] = c(NA, NA)
# one in the lower left corner:
X[20, 1] = NA
print(X)

## removeNA -
# Remove rows with NA's
removeNA(X)
# Now we have only 12 lines!

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

## interpNA -
# 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
```

Description

A collection and description of functions and methods dealing with the rearrangement of column names of 'timeSeries' objects.

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

Usage

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

Arguments

FUN	a character string indicating which statistical function should be applied. By default statistical ordering operates on the column means of the time series.
method	a character string with two elements. The first determines the choice of the distance measure, see <code>dist</code> , and the second determines the choice of the agglomeration method, see <code>hclust</code> .
robust	a logical flag which indicates if robust correlations should be used.
x	an object of class <code>timeSeries</code> or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a numeric matrix.
...	further arguments to be passed, see details.

Details**Statistically Motivated Rearrangement**

The function `statsColnames` rearranges the column names according to a 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="kendall"` to the argument list calculates Kendall's rank correlations instead the default which calculates Person's correlations.

Ordering by Hierarchical Clustering

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

Value

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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
edhec = as.timeSeries(data(edhec.tS))
colnames(edhec) = abbreviate(colnames(edhec), 6)

## sortColnames -
# Sort alphabetically
sortColnames(edhec)

## hclustColnames -
head(edhec[, hclustColnames(edhec)])
```

plot

Plot Reports

Description

Functions to plot timeSeries objects.

Usage

```
## S3 method for class 'timeSeries':
plot(x, ...)
## S3 method for class 'timeSeries':
lines(x, ...)
## S3 method for class 'timeSeries':
points(x, ...)
```

Arguments

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

Value

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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
myFinCenter <- "GMT"
EDHEC = as.timeSeries(data(edhec.ts)) [1:12, 1:4]
colnames(EDHEC) <- abbreviate(colnames(EDHEC), 6)

## plot -
plot(EDHEC[,1], type = "o", col = "steelblue",
     main = "EDHEC", xlab = "1997", ylab = "Return")
```

print	<i>Prints timeSeries objects</i>
-------	----------------------------------

Description

Prints objects of class `timeSeries`.

Usage

```
show.timeSeries(object)
```

Arguments

`object` an object of class `timeSeries`.

Value

prints an object of class `timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## print -
myFinCenter <- "GMT"
EDHEC = as.timeSeries(data(edhec.ts)) [1:12, 1:4]
colnames(EDHEC) <- abbreviate(colnames(EDHEC), 6)
print(EDHEC)
```

Description

Functions to calculate financial returns.

Usage

```
returns(x, ...)  
  
## Default S3 method:  
returns(x, method = c("continuous", "discrete", "compound", "simple"),  
        percentage = FALSE, ...)  
## S3 method for class '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	...
na.rm	...
trim	...
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 synonyms for `returns.timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
# Microsoft Data:
myFinCenter <- "GMT"
MSFT = as.timeSeries(data(msft.dat)) [1:10, 1:4]
head(MSFT)

## returnSeries -
# Continuous Returns:
returns(MSFT)
# Discrete Returns:
returns(MSFT, type = "discrete")
# Don't trim:
returns(MSFT, trim = FALSE)
# Use Percentage Values:
returns(MSFT, percentage = TRUE, trim = FALSE)
```

rowCum

Cumulated Column Statistics

Description

Functions to compute cumulative row Statistics.

Usage

```
## Default S3 method:
rowCumsums(x, na.rm = FALSE, ...)
## S3 method for class 'timeSeries':
rowCumsums(x, na.rm = FALSE, ...)
```

Arguments

<code>na.rm</code>	a logical. Should missing values be removed?
<code>x</code>	a time series, may be an object of class "matrix", "timeSeries", or "zoo".
<code>...</code>	arguments to be passed.

Value

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

Note

The methods for the "zoo" methods are still preliminary and untested. `na.rm` is not yet supported in all case, please test.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## Simulated Monthly Return Data:
x = matrix(rnorm(24), ncol = 2)

## colStats -
rowCumsums(x)
```

SpecialDailySeries *SpecialDailySeries*

Description

A collection and description of special daily timeSeries functions.

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

Usage

```
dummyDailySeries(x = rnorm(365), units = "X", zone = myFinCenter,
  FinCenter = myFinCenter)
alignDailySeries(x, method = c("before", "after", "interp", "fillNA"),
  include.weekends = FALSE, units = NULL, zone = myFinCenter,
  FinCenter = myFinCenter)
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

<code>colOrder</code>	<code>[ohlcDailyPlot]</code> - an integer vector which gives the order of the prices and the volume in the input object. By default the following order of columns from 1 to 5 is assumed: Open, high, low, close, and volume.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>FUN</code>	the function to be applied. <code>[applySeries]</code> - a function to use for aggregation, by default <code>colAvgs</code> .

<code>grid.lty, grid.nx</code>	[ohlcDailyPlot] - The type of grid line and the number of grid lines used in the plot.
<code>include.weekends</code>	[alignDailySeries] - a logical value. Should weekend dates be included or removed from the series.
<code>main</code>	[ohlcDailyPlot] - a character string to title the price and volume plot.
<code>method</code>	[alignDailySeries] - the method to be used for the alignment. A character string, one of "before", use the data from the row whose position is just before the unmatched position, or "after", use the data from the row whose position is just after the unmatched position, or "linear", interpolate linearly between "before" and "after".
<code>period</code>	[rollDailySeries] - a character string specifying the rolling period composed by the length of the period and its unit, e.g. "7d" represents one week.
<code>units</code>	[alignDailySeries] - an optional character string, which allows to overwrite the current column names of a <code>timeSeries</code> object. By default NULL which means that the column names are selected automatically. [ohlcDailyPlot] - a numeric value, specifying in which multiples the volume should be referenced on the plot labels. By default 1e6, i.e. in units of 1 Million.
<code>volume</code>	[ohlcDailyPlot] - a logical value. Should a volume plot added to the OHLC Plot. By default TRUE.
<code>x</code>	an object of class <code>timeSeries</code> .
<code>xlab, ylab</code>	[ohlcDailyPlot] - two string vectors to name the x and y axis of the price and volume plot.
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	arguments passed to other methods.

Value

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

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

`rollDailySeries`

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

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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data - Data Frame:
# To work with daily data, the best choice is "GMT"
myFinCenter <- "GMT"
MSFT = as.timeSeries(data(msft.dat))
head(MSFT)

## Align Daily Series -
# Cut out April Data from 2001:
Close = MSFT[, "Close"]
tsApril01 = window(Close, "2001-04-01", "2001-04-30")
tsApril01
# Align with NA:
tsRet = returnSeries(tsApril01, trim = TRUE)
GoodFriday(2001)
EasterMonday(2001)
alignDailySeries(tsRet, method = "fillNA", include.weekends = FALSE)
alignDailySeries(tsRet, method = "fillNA", include.weekends = TRUE)
# Interpolate:
alignDailySeries(tsRet, method = "interp", include.weekend = FALSE)
alignDailySeries(tsRet, method = "interp", include.weekend = TRUE)

## ohlcdailyPlot -
# ohlcdailyPlot(Close)
```

spreads

Calculations of spreads and mid quotes

Description

Functions to calculate spreads and midquotes from price streams.

Usage

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

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

Arguments

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

Value

all functions return an object of class `timeSeries`.

Note

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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
# Microsoft Data:
myFinCenter <- "GMT"
MSFT = as.timeSeries(data(msft.dat)) [1:10, 1:4]
head(MSFT)

## midquotes -

## spreads -
```

summary

Object Summary

Description

produces a result summary of a `timeSeries` object.

Usage

```
## S3 method for class 'timeSeries':
summary(object, ...)
```

Arguments

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

Value

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

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
EDHEC = as.timeSeries(data(edhec.tS))[1:12, 1:4]
colnames(EDHEC) <- abbreviate(colnames(EDHEC), 4)

## summary -
summary(EDHEC)
```

time

timeSeries, Positions

Description

A collection and description of functions and methods extracting and modifying positions on 'timeSeries' objects.

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

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

Usage

```
seriesPositions(object)
newPositions(object) <- value

## S3 method for class 'timeSeries':
time(x, ...)
```

```
## S3 method for class 'timeSeries':
start(x, ...)
## S3 method for class 'timeSeries':
end(x, ...)

## S3 method for class 'timeSeries':
sample(x, ...)
## S3 method for class 'timeSeries':
sort(x, ...)
## S3 method for class 'timeSeries':
rev(x)
```

Arguments

method	[alignDailySeries] - the method to be used for the alignment. A character string, one of "before", use the data from the row whose position is just before the unmatched position, or "after", use the data from the row whose position is just after the unmatched position, or "linear", interpolate linearly between "before" and "after".
object	[is][seriesData][seriesPositions][summary] - an object of class <code>timeSeries</code> .
value	a valid value for that component of <code>newPositions(x)</code> , i.e. an object of class "timeDate" with appropriate length.
x	[as] - a matrix type object to be converted. [as.vector][as.matrix][as.data.frame] - [applySeries] - [cut][end][mergeSeries][plot][print][rev][start] - an object of class <code>timeSeries</code> .
...	arguments passed to other methods.

Value

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

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

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

```

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

```

```

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

```

```

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

```

```

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

```

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```

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

## seriesPositions -
seriesPositions(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. Included are functions and methods for the generation and representation of 'timeSeries' objects, and for mathematical operations.

Functions to generate and modify 'timeSeries' objects:

<code>timeSeries</code>	Creates a 'timeSeries' object from scratch,
<code>readSeries</code>	Reads a 'timeSeries' from a spreadsheet file,
<code>applySeries</code>	Applies a function to margins of a 'timeSeries',
<code>orderStatistics</code>	Computes order statistic of a 'timeSeries'.

Data Slot and classification of 'timeSeries' objects:

<code>seriesData</code>	Extracts data slot from a 'timeSeries',
<code>isUnivariate</code>	Tests if a 'timeSeries' object is univariate,
<code>isMultivariate</code>	Tests if a 'timeSeries' object is multivariate.

Usage

```
timeSeries(data, charvec, units = NULL, format = NULL, zone = myFinCenter,
           FinCenter = myFinCenter, recordIDs = data.frame(), title = NULL,
           documentation = NULL, ...)
readSeries(file, header = TRUE, sep = ";", zone = myFinCenter,
           FinCenter = myFinCenter, title = NULL, documentation = NULL, ...)

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

orderStatistics(x)

seriesData(object)
isUnivariate(x)
isMultivariate(x)
```

Arguments

<code>by</code>	[<code>applySeries</code>] - a character either "monthly" or "quarterly". The default value is "monthly". Only operative when both arguments <code>from</code> and <code>to</code> have their default values NULL. In this case the function <code>FUN</code> will be applied to monthly or quarterly periods.
<code>charvec</code>	a character vector of dates and times.
<code>data</code>	a <code>data.frame</code> or a matrix object of numeric data.
<code>documentation</code>	optional documentation string, or a vector of character strings.
<code>file</code>	the filename of a spreadsheet data set from which to import the data records.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>header</code>	a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: 'header' is set to 'TRUE' if and only if the first row contains one fewer field than the number of columns.

<code>format</code>	the format specification of the input character vector, [as.timeSeries] - a character string with the format in POSIX notation to be passed to the time series object.
<code>from, to</code>	starting date and end date, <code>to</code> must be after <code>from</code> .
<code>FUN</code>	the function to be applied. [applySeries] - a function to use for aggregation, by default <code>colAverages</code> .
<code>object</code>	[is][seriesData][seriesPositions][show][summary] - an object of class <code>timeSeries</code> .
<code>recordIDs</code>	a data frame which can be used for record identification information. [print] - a logical value. Should the <code>recordIDs</code> printed together with the data matrix and time series positions?
<code>sep</code>	[readSeries] - the field separator used in the spreadsheet file to separate columns.
<code>title</code>	an optional title string, if not specified the inputs data name is deparsed.
<code>units</code>	[applySeries][lag][returnSeries][mergeSeries] - 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. [durationSeries] - a character value or vector which allows to set the units in which the durations are measured. By default durations are measured in seconds.
<code>x</code>	[as] - a <code>matrix</code> type object to be converted. [as.vector][as.matrix][as.data.frame] - [applySeries] - [cut][end][mergeSeries][plot][print][rev][start] - an object of class <code>timeSeries</code> .
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	arguments passed to other methods.

Details

Generation of Time Series Objects:

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

Value

timeSeries
 readSeries
 returnSeries
 applySeries
 return a S4 object of class timeSeries.

orderStatistics
 returns ...

seriesData

extracts the @Data slot from a timeSeries object. Thus, seriesData returns an object of class matrix.

isUnivariate
 isMultivariate

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

plot
 lines
 points
 print
 plot and print methods for an object of class timeSeries.

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.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
# Microsoft Data:
myFinCenter <- "GMT"
MSFT = as.timeSeries(data(msft.dat))
head(MSFT)

## timeSeries -
# Create a timeSeries Objec - The Direct Way ...
Close = MSFT[, 5]
```

```

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

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

## returnSeries -
# Compute Returns:
args(returnSeries)
# Continuous Returns:
returnSeries(tsApril01)
# Discrete Returns:
returnSeries(tsApril01, type = "discrete")
# Don't trim:
returnSeries(tsApril01, trim = FALSE)
# Use Percentage Values:
tsRet = returnSeries(tsApril01, percentage = TRUE, trim = FALSE)
tsRet

## applySeries -
# 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)

```

TimeSeriesSubsettings

timeSeries Subsetting

Description

A collection and description of functions and methods for subsetting timeSeries objects.

"["	"[" method for 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

```
## S3 method for class 'timeSeries':
x[i = min(1, nrow(x@Data)):nrow(x@Data),
  j = min(1, ncol(x@Data)):ncol(x@Data)]
## S3 method for class 'timeSeries':
window(x, from, to, ...)
## S3 method for class 'timeSeries':
head(x, n = 6, recordIDs = FALSE, ...)
## S3 method for class 'timeSeries':
tail(x, n = 6, recordIDs = FALSE, ...)
## S3 method for class 'timeSeries':
outlier(x, sd = 10, complement = TRUE, ...)

## S3 method for class 'timeSeries':
cut(x, from, to, ...)
```

Arguments

<code>complement</code>	[<code>outlierSeries</code>] - a logical flag, should the outlier series or its complement be returns, by default TRUE which returns the series free of outliers.
<code>from, to</code>	starting date and end date, <code>to</code> must be after <code>from</code> .
<code>i, j</code>	[<code>"</code>] - index arguments used for subsettings.
<code>n</code>	[<code>head</code>][<code>tail</code>] - an integer specifying the number of lines to be returned. By default <code>n=6</code> .
<code>recordIDs</code>	[<code>head</code>][<code>tail</code>] - a logical value. Should the <code>recordIDs</code> returned together with the data matrix and time series positions?
<code>sd</code>	[<code>outlierSeries</code>] - 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.
<code>x</code>	an object of class <code>timeSeries</code> .
<code>...</code>	arguments passed to other methods.

Value

all functions return an object of class `timeSeries`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -  
# Create an artificial timeSeries object:  
myFinCenter <- "GMT"  
charvec = timeCalendar()  
set.seed(4711)  
data = matrix(exp(cumsum(rnorm(12, sd = 0.1))))  
tS = timeSeries(data, charvec, units = "tS")  
tS  
  
## "[" -  
tS[1:3, ]  
  
## head -  
head(tS)
```

Index

*Topic **chron**

- aggregate, 2
- apply, 3
- as, 3
- attach, 5
- bind, 6
- cor, 9
- dim, 10
- durations, 11
- lag, 12
- math, 13
- merge, 15
- model.fram, 15
- monthly, 17
- orderColnames, 20
- plot, 22
- print, 23
- returns, 24
- SpecialDailySeries, 26
- spreads, 28
- summary, 29
- time, 30
- TimeSeriesClass, 32
- TimeSeriesSubsettings, 37

*Topic **math**

- na, 18

*Topic **univar**

- colCum, 7
- rowCum, 25

- [.timeSeries
(TimeSeriesSubsettings), 37

- abs.timeSeries (math), 13
- aggregate, 2
- alignDailySeries
(SpecialDailySeries), 26
- apply, 3
- applySeries (TimeSeriesClass), 32
- as, 3
- as.data.frame.timeSeries (as), 3

- as.matrix.timeSeries (as), 3
- as.timeSeries (as), 3
- as.ts.timeSeries (as), 3
- as.vector.timeSeries (as), 3
- attach, 5

- bind, 6

- cbind (bind), 6
- colCum, 7
- colCummaxs (colCum), 7
- colCummins (colCum), 7
- colCumprods (colCum), 7
- colCumreturns (colCum), 7
- colCumsums (colCum), 7
- colnames<- .timeSeries (dim), 10
- cor, 9
- countMonthlyRecords (monthly), 17
- cov (cor), 9
- cumsum.timeSeries (colCum), 7
- cut.timeSeries
(TimeSeriesSubsettings), 37

- diff.timeSeries (math), 13
- dim, 10
- dimnames.timeSeries (dim), 10
- dummyDailySeries
(SpecialDailySeries), 26
- durations, 11
- durationSeries (durations), 11

- end.timeSeries (time), 30
- exp.timeSeries (math), 13

- fapply (apply), 3

- getReturns (returns), 24

- hclustColnames (orderColnames), 20
- head.timeSeries
(TimeSeriesSubsettings), 37

interpNA(*na*), 18
 is.array.timeSeries(*dim*), 10
 is.timeSeries(*as*), 3
 isMonthly(*monthly*), 17
 isMultivariate(*TimeSeriesClass*),
 32
 isUnivariate(*TimeSeriesClass*), 32

 lag, 12
 lines.timeSeries(*plot*), 22
 log.timeSeries(*math*), 13

 math, 13
 merge, 15
 midquotes(*spreads*), 28
 midquoteSeries(*spreads*), 28
 model.fram, 15
 model.frame, 16
 model.frame(*model.fram*), 15
 monthly, 17

 na, 18
 na.omit.timeSeries(*na*), 18
 newPosition<-(*time*), 30

 ohlcDailyPlot
 (*SpecialDailySeries*), 26
 Ops.timeSeries(*math*), 13
 orderColnames, 20
 orderStatistics
 (*TimeSeriesClass*), 32
 outlier.timeSeries
 (*TimeSeriesSubsettings*), 37

 pcaColnames(*orderColnames*), 20
 plot, 22
 points.timeSeries(*plot*), 22
 print, 23

 quantile.timeSeries(*math*), 13

 rbind(*bind*), 6
 readSeries(*TimeSeriesClass*), 32
 removeNA(*na*), 18
 returns, 24
 returnSeries(*returns*), 24
 rev.timeSeries(*time*), 30
 rollDailySeries
 (*SpecialDailySeries*), 26
 rollMonthlySeries(*monthly*), 17

 rollMonthlyWindows(*monthly*), 17
 rowCum, 25
 rowCumsums(*rowCum*), 25
 rownames<-*.timeSeries*(*dim*), 10

 sample.timeSeries(*time*), 30
 sampleColnames(*orderColnames*), 20
 scale.timeSeries(*math*), 13
 seriesData(*TimeSeriesClass*), 32
 seriesPositions(*time*), 30
 show,timeSeries-method(*print*), 23
 show.timeSeries(*print*), 23
 sign.timeSeries(*math*), 13
 sort.timeSeries(*time*), 30
 sortColnames(*orderColnames*), 20
 SpecialDailySeries, 26
 spreads, 28
 spreadSeries(*spreads*), 28
 sqrt.timeSeries(*math*), 13
 start.timeSeries(*time*), 30
 statsColnames(*orderColnames*), 20
 substituteNA(*na*), 18
 summary, 29

 tail.timeSeries
 (*TimeSeriesSubsettings*), 37
 time, 30
 timeSeries(*TimeSeriesClass*), 32
 timeSeries-class
 (*TimeSeriesClass*), 32
 TimeSeriesClass, 32
 TimeSeriesSubsettings, 37

 window.timeSeries
 (*TimeSeriesSubsettings*), 37