

The fCalendar Package

October 22, 2006

Version 240.10068

Date 1996 - 2006

Title Rmetrics - Chronological and Calendarical Objects

Author Diethelm Wuertz and many others, see the SOURCE file

Depends R (>= 2.2.0), methods, fEcofin

Maintainer Diethelm Wuertz and Rmetrics Core Team <wuertz@itp.phys.ethz.ch>

Description Environment for teaching “Financial Engineering and Computational Finance”

License GPL Version 2 or later

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

R topics documented:

RmetricsUtilities	2
DaylightSavingTime	4
TimeDateClass	5
TimeDateSubsets	10
TimeDateMathOps	13
TimeDateSpecDates	16
TimeDateCoercion	19
TimeSeriesClass	21
TimeSeriesData	27
TimeSeriesPositions	33
TimeSeriesCoercion	35
HolidayDates	37
HolidayCalendars	40
TimeSeriesImport	42
Index	47

Description

A collection and description of functions which mask function from R's base installation to generate generic functions and some additional generic functions which may be useful.

Usage

```
align.default(x, y, xout, method = "linear", n = 50, rule = 1, f = 0,
  ties = mean, ...)

log.default(x, base = exp(1))

outlier.default(x, sd = 5, complement = TRUE, ...)

round.default(x, digits, ...)

sample.default(x, size, replace = FALSE, prob = NULL, ...)

sort.default(x, partial = NULL, na.last = NA, decreasing = FALSE,
  method = c("shell", "quick"), index.return = FALSE, ...)

var.default(x, y = NULL, na.rm = FALSE, use)

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

as.POSIXlt(x, tz = "")

as.matrix.ts(x)
as.matrix.mts(x)

currentYear
myUnits
```

Arguments

base	[log] - additional argument to the log function.
digits	[round] - an integer indicating the precision to be used.
f	[align] - for method="constant" the value of f takes a number between 0 and 1 inclusive, indicating a compromise between left- and right-continuous step functions. If y0 and y1 are the values to the left and right of the point then the value is $y_0 * (1-f) + y_1 * f$ so that f=0 is right-continuous and f=1 is left-continuous.

method	[align] - a character string which specifies the alignment method to be used. Choices are "linear" or "constant".
n	[align] - if <code>xout</code> is not specified, alignment takes place at <code>n</code> equally spaced points spanning the interval <code>range(x)</code> .
partial, na.last, decreasing, index.return	[sort] - additional arguments to the <code>sort</code> function.
rule	[align] - an integer describing how alignment is to take place outside the interval <code>range(x)</code> . If <code>rule=1</code> then NA's are returned for such points and if <code>rule=2</code> , the value at the closest data extreme is used.
sd, complement	[outlier] - <code>sd</code> - a numeric value of standard deviations, e.g. 5 means that values larger or smaller than five times the standard deviation of the series will be detected. <code>complement</code> - a logical flag, should the outlier series or its complements be returned?
size, replace, prob	[sample] - <code>size</code> - is a non-negative integer giving the number of items to choose, <code>replace</code> - a logical flag. Should sampling be with replacement? <code>prob</code> - a vector of probability weights for obtaining the elements of the vector being sampled.
ties	[align] - handles tied <code>x</code> values. Either a function with a single vector argument returning a single number result or the string "ordered".
tz	[as.POSIXlt] - time zone specification.
value	[colnames][rownames] - additional arguments to the <code>colnames</code> and <code>rownames</code> functions.
x	[align] - x-coordinates of the points to be aligned. [log][sort][var] - first argument. [as.matrix.ts][as.matrix.mts] - an univariate or multivariate time series object of class "ts" or "mts" which will be transformed into an one-column or multi-column rectangular object of class "matrix". [as.POSIXlt] - an object to be converted.
xout	[align] - a set of values specifying where interpolation is to take place.
y, na.rm, use	[align] - y-coordinates of the points to be aligned. [var] - additional arguments to the <code>var</code> function.
...	arguments to be passed.

Details

For details we refer to the original help pages.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## SOURCE("fCalendar.1A-RmetricsUtilities")
```

DaylightSavingTime *Daylight Saving Time Rules*

Description

A collection and description of functions for 92 cities and regions, which return time zone offsets and daylight saving time rules.

The functions are:

Adelaide Algiers Amsterdam Anchorage Andorra Athens Auckland Bahrain Bangkok Beirut Belfast Belgrade Berlin Bogota Bratislava Brisbane Brussels Bucharest Budapest BuenosAires Cairo Calcutta Caracas Casablanca Cayman Chicago Copenhagen Darwin Denver Detroit Dubai Dublin Eastern Edmonton Frankfurt Helsinki HongKong Honolulu Indianapolis Istanbul Jakarta Jerusalem Johannesburg Kiev KualaLumpur Kuwait Lagos Lisbon Ljubljana London LosAngeles Luxembourg Madrid Manila Melbourne MexicoCity Monaco Montreal Moscow Nairobi Nassau NewYork Nicosia Oslo Pacific Paris Perth Prague Riga Riyadh Rome Seoul Shanghai Singapore Sofia Stockholm Sydney Taipei Tallinn Tehran Tokyo Tunis Vaduz Vancouver Vienna Vilnius Warsaw Winnipeg Zagreb Zurich.

Note

There are currently two synonyms available "Pacific" for Los Angeles and "Eastern" for New York.

Note that we leave the space in all double named cities like New York or Hong Kong and don't use an underscore for it.

All the entries are retrieved from the Ical library which is available under GNU's GPL licence.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## SOURCE("fCalendar.2B-DaylightSavingTime")

## ...
# DST Rules for Zurich:
head(Zurich())
tail(Zurich())
```

TimeDateClass *timeDate Class*

Description

A collection and description of functions and methods for managing date and time around the globe for any financial center. The concept allows for dealing with several time zones at the same time, day light saving time and holiday calendars independently of the date and time specifications of the operating system implemented on a specific computer.

The functions for Financial Centers are:

<code>rulesFinCenter</code>	Returns DST rules for a financial center,
<code>listFinCenter</code>	Lists all supported financial centers.

The functions for the Generation of 'timeDate' objects are:

<code>timeDate</code>	S4: Creates 'timeDate' object from a character vector,
<code>is.timeDate</code>	Tests if the object is of class 'timeDate',
<code>timeCalendar</code>	Creates 'timeDate' object from calendar atoms,
<code>timeSequence</code>	Creates regularly spaced object of class 'timeDate',
<code>seq</code>	A synonyme function for timeSequence,
<code>Sys.timeDate</code>	Returns system time as an object of class 'timeDate'.

The functions to represent 'timeDate' objects:

<code>print.timeDate</code>	Prints 'timeDate' Object,
<code>plot.timeDate</code>	Plots with GMT 'timeDate' x-axis,
<code>points.timeDate</code>	Adds points to a 'timeDate' plot,
<code>lines.timeDate</code>	Adds lines to a 'timeDate' plot,
<code>summary.timeDate</code>	Summarizes details of a 'timeDate' object,
<code>format.timeDate</code>	Formats 'timeDate' as ISO conform character string,

Usage

```
myFinCenter
rulesFinCenter(FinCenter = myFinCenter)
listFinCenter(pattern = "*")

timeDate(charvec = Sys.timeDate(), format = NULL, zone = myFinCenter,
         FinCenter = myFinCenter)
timeCalendar(y = currentYear, m = 1:12, d = 1, h = NULL, min = NULL,
            s = NULL, zone = myFinCenter, FinCenter = myFinCenter)
timeSequence(from = "2004-01-01", to = format(Sys.time(), "%Y-%m-%d"),
            by = c("day", "year", "quarter", "month", "week", "hour", "min", "sec"),
            length.out = NULL, format = "", zone = myFinCenter, FinCenter = myFinCenter)
Sys.timeDate(FinCenter = myFinCenter)

## S3 method for class 'timeDate':
seq(from, to, by = c("day", "year", "quarter", "month",
```

```

    "week", "hour", "min", "sec"), length.out = NULL, ...)

is.timeDate(object)

## S3 method for class 'timeDate':
print(x, ...)
## S3 method for class 'timeDate':
plot(x, y, ...)
## S3 method for class 'timeDate':
summary(object, ...)
## S3 method for class 'timeDate':
format(x, ...)

currentYear
myUnits

```

Arguments

<code>by</code>	a character string, containing one of "sec", "min", "hour", "day", "week", "month" or "year". This can optionally be preceded by an integer and a space, or followed by "s".
<code>charvec</code>	a character vector of dates and times.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>format</code>	the format specification of the input character vector.
<code>from, to</code>	[timeSequence] - starting date, required, and end date, optional. If supplied <code>to</code> must be after <code>from</code> , [seq] - <code>cr</code> in this case the <code>from</code> and <code>to</code> dates must be objects of class <code>timeDate</code> .
<code>h, min, s</code>	hours of the days (0-23), defaults are 0, minutes of the days (0-59), defaults are 0, and seconds of the days (0-59), defaults are 0.
<code>length.out</code>	<code>length.out</code> integer, optional. Desired length of the sequence, if specified "to" will be ignored.
<code>method</code>	[modify] - a character string defining the modification method, one of "sort", "round", or "trunc".
<code>object</code>	[is.timeData][summary] - an object of class <code>timeDate</code> .
<code>pattern</code>	a pattern character string which can be recognized by the <code>grep</code> function. Wild cards are allowed.
<code>x</code>	[isWeekday][isWeekend][isBizday][weekDay] - an object of class <code>timeDate</code> . [format][print][plot] - an object of class <code>timeDate</code> .
<code>y, m, d</code>	[timeCalendar] - calendar years (e.g. 1997), defaults are 1960, calendar months (1-12), defaults are 1, and calendar days (1-31), defaults are 1, [plot] - a numeric vector.
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	arguments passed to other methods.

Details

For the management of chronological objects under R three concepts are available: The first is the implementation of date and time in R's `chron` package neglecting locals, time zones and day light saving times. This approach is appropriate for economic time series. The second approach, available in R's base package implements the POSIX standard to date and time objects, named "POSIXt". Unfortunately, the representation of these objects is operating system dependent and especially under MS Windows several problems appear in the management of time zones and day light saving times. Here we present a solution to overcome these difficulties with POSIX objects and introduce a new S4 class of 'timeDate' objects which allow for powerful methods to represent dates and times in different financial centers around the world. Many of the basic functionalities of these objects are in common with S-Plus' `timeDate` objects and thus many of your privately written functions for `FinMetrics` may also be used within R's environment.

A major difference is the time zone concept which is replaced by the "Financial Center" concept. The `FinCenter` character variable specifies where you are living and at which financial center you are working. With the variable `myFinCenter` you can overwrite the default setting with your personal settings. With the specification of the `FinCenter` your system knows what rules rules for day light saving times should be applied, what is your holiday calendar, what are your interest rate conventions. (Not all specifications are already implemented.) Many other aspects can be easily accessed when a financial center is named. So we can distinguish between Frankfurt and Zurich, which both belong to the same time zone, but differed in DST changes in the eighties and have different holiday calendars. Furthermore, since the underlying time refers to "GMT" and DST rules and all other information is available in local (ASCII) databases, we are sure, that R delivers with such a date/time concept on every computer independent of the operating system in use, identical results.

Another important feature of the "timeDate" concept used here is the fact that we don't rely on American or European ways to write dates. We use consequently the ISO-8601 standard for date and time notations.

Financial Centers

There are two functions concerned with the financial centers. The first, `rulesFinCenter`, lists the daylight saving rules for a selected financial center, and the second, `listFinCenter`, lists all centers available in the database. There is no dependency on the POSIX implementation of your operating system because all time zone and day light saving time information is stored locally in ASCII files. It is important to say, that the `TZ` environment variable must set to "GMT" in your system environment that there are no conflicts with the POSIX time zone management.

Through the definition of financial centers it becomes possible to introduce in the future a specification structure for financial centers, which includes further information for a center like holiday calendars, currency and interest rate conventions or many others.

Generation of timeDate Objects

We have defined a 'timeDate' class which is in many aspects similar to the S-Plus class with the same name, but has also some important advantageous differences. The S4 class has four Slots, the `Data` slot which holds date and time as 'POSIXt' objects in the standard ISO-8601 format, the `Dim` slot which gives the dimension of the data object, the `format` specification slot and the `FinCenter` slot which holds the name of the financial center.

Three functions allow to generate date/time objects: `timeDate` from character vectors, `timeCalendar` from date and time atoms, and `timeSequence` from a "from/to" or from a "from/length" sequence specification. Note, time zone transformations are easily handled by by the `timeDate` functions

which can also take `timeDate` and `POSIXt` objects as inputs, while transforming them between financial centers and/or time zones specified by the arguments `zone` and `FinCenter`. Finally the function `Sys.timeDate` returns current system time in form of a `timeDate` object.

Tests and Representation of timeDate Objects:

We have implemented four methods to test and represent `timeDate` objects. The method `is.timeDate` checks if a given object is of class `"timeDate"`. The `print` method returns the date/time in square `"[]"` brackets to distinguish the output from other date and time objects. On top of the date and time output the name of the `FinCenter` is printed. The `summary` method returns a printed report with information about the `"timeDate"` object. Finally, the `format` methods allows to transform objects into a ISO conform formatted character strings.

Mathematical Operations:

This is a collection of S3 methods for objects of class `timeDate` to perform mathematical operations. Included are methods to extract or replace subsets from `timeDate` objects, to perform arithmetic `"+"` and `"-"` operations, to group 'Ops' generic functions, to return suitably lagged and iterated differences, to return differences of two `timeDate` objects, to concatenate objects, to replicate objects, to rounds objects, to truncates objects, to extract the first or last entry of a vector, to sort the objects of the elements of a date/time vector, and to revert `timeDate` vector objects.

Transformation of Objects:

This is a collection of S3 methods for objects of class `timeDate` to transform those objects between different representations. Included are methods to transform `timeDate` objects to character strings, to data frames, to `POSIXct` or `POSIXlt` objects, to Julian counts, to extract date/time atoms from calendar dates, and to extract the months atom from a `timeDate` object.

Value

`rulesFinCenter`

`listFinCenter`

the first function returns a printed list of DST rules, the second lists time zones available in the database.

`timedate`

`timeCalendar`

`timeSequence`

return a S4 object of class 'timeDate'.

`Sys.timeDate`

returns the system time as an object of class 'timeDate'.

`is.timeDate`

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

print

prints the financial center and date and time vector for a `timeDate` object.

summary

returns a summary report of the details of a `timeDate` object. This includes the starting and end

date, the number of dates the format and financial center in use.

format

returns an ISO conform formatted character string.

Note

originally, these functions were written for Rmetrics users using R and Rmetrics under Microsoft's Windows XP operating system where time zones, daylight saving times and holiday calendars are not or insufficiently supported. The functions are untested for other system environments, but may be used.

The usage of the Ical Library and the introduction of the FinCenter concept was originally developed for R Version 1.5. The `timeDate` and `timeSeries` objects were added for R Version 1.8.1. Minor changes were made to adapt the functions for R Version 1.9.1. As a consequence, newer concepts like the `Date` objects were not yet considered and included in this collection of date and time concepts. With R Version 2.3.0 a major update has been made adding many new generic functions and renaming a few already existing functions, please be aware of this.

Note, the date/time conversion from an arbitrary timezone to GMT cannot be unique, since date/time objects appear twice during the hour when DST changes. A bookkeeping which takes care if DST is effective or not is not yet included. However, in most applications this is not necessary since the markets are closed on weekends, especially at times when DST usually changes. It is planned for the future to implement the DST change properly.

The ISO-8601 midnight standard has been implemented. Note, that for example "2005-01-01 24:00:00" is a valid date/time string.

Beside the examples given in the manual pages additional demo and test files are available with much more examples including also those from the book of Zivot and Wang (2003).

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

References

Bateman R., (2000); *Time Functionality in the Standard C Library*, Novell AppNotes, September 2000 Issue, 73–85.

ISO-8601, (1988); *Data Elements and Interchange Formats - Information Interchange, Representation of Dates and Time*, International Organization for Standardization, Reference Number ISO 8601, 14 pages.

James D.A., Pregibon D. (1992), *Chronological Objects for Data Analysis*, Reprint.

Ripley B.D., Hornik K. (2001); *Date-Time Classes*, R-News, Vol. 1/2 June 2001, 8–12.

Zivot, E., Wang J. (2003); *Modeling Financial Time Series with S-Plus*, Springer, New-York.

See Also

`timeDateCoercion`, `timeDateSpecDates`

We also recommend to inspect the help pages for the POSIX time and date class, `?Dates`, and the help pages from the contributed R packages `chron` and `date`.

Examples

```
## SOURCE("fCalendar.3A-TimeDateClass")

## myFinCenter -
myFinCenter

## - Character Vecor Strings:
dts = c("1989-09-28", "2001-01-15", "2004-08-30", "1990-02-09")
dts
tms = c("23:12:55", "10:34:02", "08:30:00", "11:18:23")
tms

## timeDate -
timeDate(dts, format = "%Y-%m-%d", FinCenter = "GMT" )
timeDate(dts, format = "%Y-%m-%d", FinCenter = "Europe/Zurich")
timeDate(paste(dts, tms), format = "%Y-%m-%d %H:%M:%S",
  zone = "GMT", FinCenter = "GMT")
timeDate(paste(dts, tms), format = "%Y-%m-%d %H:%M:%S",
  zone = "Europe/Zurich", FinCenter = "Europe/Zurich")
timeDate(paste(dts, tms), format = "%Y-%m-%d %H:%M:%S",
  zone = "GMT", FinCenter = "Europe/Zurich")

## timeCalendar -
timeCalendar(m = c(9, 1, 8, 2), d = c(28, 15, 30, 9),
  y = c(1989, 2001, 2004, 1990), FinCenter = "GMT")
timeCalendar(m = c(9, 1, 8, 2), d = c(28, 15, 30, 9),
  y = c(1989, 2001, 2004, 1990), FinCenter = "Europe/Zurich")
timeCalendar(h = c(9, 14), min = c(15, 23))

## timeSequence -
timeSequence(from = "2004-03-12", to = "2004-04-11",
  format = "%Y-%m-%d", FinCenter = "GMT")
timeSequence(from = "2004-03-12", to = "2004-04-11",
  format = "%Y-%m-%d", FinCenter = "Europe/Zurich")

## timeDate -
# Note, ISO and American Formats are Auto-Detected
timeDate("2004-12-11", FinCenter = "GMT")
timeDate("12/11/2004", FinCenter = "GMT")

## print | summary | format -
tC = timeCalendar()
print(tC)
summary(tC)
format(tC)
```

TimeDateSubsets *timeDate Class, Subsetting*

Description

A collection and description of functions and methods for subsetting 'timeDate' objects.

Functions for testing 'timeDate' objects:

<code>isWeekday</code>	Tests if a date is a weekday or not,
<code>isWeekend</code>	Tests if a date falls on a weekend or not,
<code>isBizday</code>	Tests if a date is a business day or not,
<code>isHoliday</code>	Tests if a date is a non-business day or not.

Functions for getting additional information on 'timeDate' objects:

<code>getDayOfWeek</code>	Returns the day of the week to a 'timeDate' object,
<code>getDayOfYear</code>	Returns the day of the year to a 'timeDate' object.

Functions to extract subsets from 'timeDate' objects:

<code>[.timeDate</code>	Extracts or replaces subsets from 'timeDate' objects,
<code>cut.timeDate</code>	Extracts a piece from a 'timeDate' object,
<code>start.timeDate</code>	Extracts the first entry of a 'timeDate' object,
<code>end.timeDate</code>	Extracts the last entry of a 'timeDate' object,
<code>length.timeDate</code>	Gets the length of a 'timeDate' object,
<code>blockStart</code>	Creates start dates for equally sized blocks,
<code>blockEnd</code>	Creates end dates for equally sized blocks.

Usage

```
isWeekday(x)
isWeekend(x)
isBizday(x, holidays = holidayNYSE())
isHoliday(x, holidays = holidayNYSE())

getDayOfWeek(x)
getDayOfYear(x)

## S3 method for class 'timeDate':
x[..., drop = TRUE]
## S3 method for class 'timeDate':
cut(x, from , to, ...)
## S3 method for class 'timeDate':
start(x, ...)
## S3 method for class 'timeDate':
end(x, ...)
## S3 method for class 'timeDate':
length(x)

blockStart(x, block = 20)
blockEnd(x, block = 20)
```

Arguments

<code>block</code>	an integer value specifying the length in number of records for numerically sized blocks of dates.
<code>drop</code>	<code>[TRUE]</code> - a logical flag, by default <code>TRUE</code> .
<code>from, to</code>	starting date, required, and end date, optional. If supplied <code>to</code> must be after <code>from</code> .

holidays	[isBizday] - holiday dates from a holiday calendar. An object of class <code>timeDate</code> .
method	[modify] - a character string defining the modification method, one of "sort", "round", or "trunc".
x	[isWeekday][isWeekend][isBizday][weekDay] - an object of class <code>timeDate</code> . [format][print] - an object of class <code>timeDate</code> .
...	arguments passed to other methods.

Value

```
isWeekday
isWeekend
isBizday
isHoliday
```

the functions return logical vectors indicating if a date is a weekday, a weekend day, a business day, or a holiday. Note for business and holidays extraction an holiday/business calendar has to be specified.

```
getDayOfWeek
```

`getDayOfWeek` the function `getDayOfWeek` returns a three letter character string with the names of the day of the week, and the function `getDayOfWeek` returns the day count as integer value starting January, 1st.

```
[.timeDate
cut.timeDate
start.timeDate
end.timeDate
blockStart
blockEnd
```

these are functions for subsetting "timeDate" objects. The function `[.timeDate` extracts or replaces subsets, the function `cut` extracts a piece, the functions `start` and `end` extract the first and last element, the functions `blockStart` and `blockEnd` create vectors of start and end values for equally sized blocks. All functions return an object of class "timeDate".

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

```
timeDateCoercion, timeDateCoercion, timeDateMathOps, timeDateSpecDates
```

We also recommend to inspect the help pages for the POSIX time and date class, `?Dates`, and the help pages from the contributed R packages `chron` and `date`.

Examples

```
## SOURCE ("fCalendar.3B-TimeDateSubsets")
## Easter -
```

```

currentYear

## Easter -
Easter(currentYear)
tS = timeSequence(from = Easter(currentYear)-7*24*3600, length.out = 8)
tS

## Weekdays and Weekend Days:
isWeekday(tS)
isWeekend(tS)
getDayOfWeek(tS)

## Business Days and Holidays:
holidayNYSE()
isBizday(tS, holidayNYSE())
isHoliday(tS, holidayNYSE())

## [ - Subsetting:
tS[c(1, 6:8)]
tS[isBizday(tS)]
tS[isHoliday(tS)]

## cut -
GoodFriday(currentYear)
EasterMonday(currentYear)
cut(tS, from = GoodFriday(currentYear), to = EasterMonday(currentYear))

## start | end -
start(tS)
end(tS)

```

TimeDateMathOps

timeDate Class, Functions and Methods

Description

A collection and description of functions and methods for mathematical and logical operations on 'timeDate' objects.

The functions to perform mathematical operations:

Ops.timeDate	Group 'Ops' generic functions for 'timeDate' objects,
+.timeDate	Performs arithmetic + operation on 'timeDate' objects,
-.timeDate	Performs arithmetic - operation on 'timeDate' objects,
diff.timeDate	Returns suitably lagged and iterated differences,
diff.timeDate	Returns a difference of two 'timeDate' objects,
round.timeDate	Rounds objects of class 'timeDate',
trunc.timeDate	Truncates objects of class 'timeDate'.

The functions to concatenate, sort and order 'timeDate' objects:

c.timeDate	Concatenates 'timeDate' objects,
rep.timeDate	Replicates a 'timeDate' object,

sort.timeDate	Sorts a 'timeDate' object,
sample.timeDate	Resamples a 'timeDate' object,
unique.timeDate	Makes a 'timeDate' object unique,
rev.timeDate	Reverts a 'timeDate' object.

Usage

```
## S3 method for class 'timeDate':
Ops(e1, e2)
## S3 method for class 'timeDate':
e1 + e2
## S3 method for class 'timeDate':
e1 - e2
## S3 method for class 'timeDate':
diff(x, lag = 1, differences = 1, ...)
difftimeDate(time1, time2,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"))
## S3 method for class 'timeDate':
round(x, units = c("days", "hours", "mins"), ...)
## S3 method for class 'timeDate':
trunc(x, units = c("days", "hours", "mins"), ...)

## S3 method for class 'timeDate':
c(..., recursive = FALSE)
## S3 method for class 'timeDate':
rep(x, ...)
## S3 method for class 'timeDate':
sample(x, ...)
## S3 method for class 'timeDate':
sort(x, ...)
## S3 method for class 'timeDate':
unique(x, ...)
## S3 method for class 'timeDate':
rev(x)
```

Arguments

differences	[lag] - an integer indicating the order of the difference.
e1, e2	["+"][["-"]]["Ops"] - usually objects of class timeDate, in the case of addition and subtraction e2 may be of class numeric.
lag	[lag] - an integer indicating which lag to use.
method	[modify] - a character string defining the modification method, one of "sort", "round", or "trunc".
recursive	[c] - a logical. If recursive is set to TRUE, the function recursively descends through lists combining all their elements into a vector.
time1, time2	[difftime] - two objects objects of class timeDate.

<code>units</code>	a character string denoting the date/time units in which the results are desired.
<code>x</code>	<code>[isWeekday][isWeekend][isBizday][weekDay]</code> - an object of class <code>timeDate</code> . <code>[format][print]</code> - an object of class <code>timeDate</code> .
<code>...</code>	arguments passed to other methods.

Value

```
Ops.timeDate
+.timeDate
-.timeDate
diff.timeDate
difftimeDate
round.timeDate
trunc.timeDate
```

these are functions for mathematical operations. Group codeOps are generic functions which manage mathematical operations. The plus operator "+" performs arithmetic "+" operation on `timeDate` objects, and the minus operator "-" returns a `difftime` object if both arguments `e1` and `e2` are "timeDate" objects, or returns a "timeDate" object `e2` seconds earlier than `e1`. For the function, `diff.timeDate`, if `x` is a vector of length `n` and `differences=1`, then the computed result is equal to the successive differences `x[(1+lag):n] - x[1:(n-lag)]`. If `difference` is larger than one this algorithm is applied recursively to `x`. Note that the returned value is a vector which is shorter than `x`. The function, `difftimeDate`, takes a difference of two `timeDate` objects and returns an object of class "difftime" with an attribute indicating the units. The two functions `round` and `trunc` allow to round or to truncate "timedate" objects to the specified unit and return them as "timedate" objects.

```
c
rep
sample
sort
unique
rev
```

these are functions to concatenate, to replicate, to resample, to sort, to make unique, and to revert `timeDate` objects. The functions return an object of class "timeDate".

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

```
timeDateClass, timeDateCoercion, timeDateSpecDates
```

We also recommend to inspect the help pages for the POSIX time and date class, `?Dates`, and the help pages from the contributed R packages `chron` and `date`.

Examples

```
## SOURCE("fCalendar.3C-TimeDateMathOps")
## c -
```

```

# Create Character Vectors:
dts = c("1989-09-28", "2001-01-15", "2004-08-30", "1990-02-09")
dts
tms = c("23:12:55", "10:34:02", "08:30:00", "11:18:23")
tms

## "+/-" -
# Add One Day to a Given timeDate Object:
GMT = timeDate(dts, zone = "GMT", FinCenter = "GMT")
GMT
ZUR = timeDate(dts, zone = "GMT", FinCenter = "Europe/Zurich")
ZUR
GMT + 24*3600
ZUR[2] - ZUR[1]

## "[" -
# Subsets from and Lops for timeDate Objects:
GMT[GMT < GMT[2]]
ZUR[ZUR < ZUR[3]] == ZUR[1:3]

## diff -
# Suitably Lagged and Iterated Differences:
diff(GMT)
diff(GMT, lag = 2)
diff(GMT, lag = 1, diff = 2)
difftimeDate(GMT[1:2], GMT[-(1:2)])

## c | rep -
# Concatenate and Replicate timeDate Objects:
c(GMT[1:2], ZUR[1:2])
c(ZUR[1:2], GMT[1:2])
rep(ZUR[2], times = 3)
rep(ZUR[2:3], times = 2)

```

TimeDateSpecDates *timeDate Class, Special Date Functions*

Description

A collection and description of functions for the generation of sequences of dates according to special date rules.

The functions for special 'timeDate' sequences are:

timeFirstDayInMonth	Computes the first day in a given month and year,
timeLastDayInMonth	Computes the last day in a given month and year,
timeFirstDayInQuarter	Computes the first day in a given quarter and year,
timeLastDayInQuarter	Computes the last day in a given quarter and year,
timeNdayOnOrAfter	Computes date that is a "on-or-after" n-day,
timeNdayOnOrBefore	Computes date that is a "on-or-before" n-day,
timeNthNdayInMonth	Computes n-th occurrence of a n-day in year/month,
timeLastNdayInMonth	Computes the last n-day in year/month.

Usage

```

timeFirstDayInMonth(charvec, format = "%Y-%m-%d", zone = myFinCenter,
  FinCenter = myFinCenter)
timeLastDayInMonth(charvec, format = "%Y-%m-%d", zone = myFinCenter,
  FinCenter = myFinCenter)

timeFirstDayInQuarter(charvec, format = "%Y-%m-%d", zone = myFinCenter,
  FinCenter = myFinCenter)
timeLastDayInQuarter(charvec, format = "%Y-%m-%d", zone = myFinCenter,
  FinCenter = myFinCenter)

timeNdayOnOrAfter(charvec, nday = 1, format = "%Y-%m-%d",
  zone = myFinCenter, FinCenter = myFinCenter)
timeNdayOnOrBefore(charvec, nday = 1, format = "%Y-%m-%d",
  zone = myFinCenter, FinCenter = myFinCenter)

timeNthNdayInMonth(charvec, nday = 1, nth = 1, format = "%Y-%m-%d",
  zone = myFinCenter, FinCenter = myFinCenter)
timeLastNdayInMonth(charvec, nday = 1, format = "%Y-%m-%d",
  zone = myFinCenter, FinCenter = myFinCenter)

```

Arguments

<code>charvec</code>	a character vector of dates and times.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>format</code>	the format specification of the input character vector.
<code>nday</code>	an integer vector with entries ranging from 0 (Sunday) to 6 (Saturday).
<code>nth</code>	an integer vector numbering the n-th occurrence.
<code>zone</code>	the time zone or financial center where the data were recorded.

Details

We have implemented functions to generate special "timeDate" sequences. These are functions to compute the last day in a given month and year, to compute the dates in a month that is a n-day on or after a given date, to compute the dates in a month that is a n-day on or before a specified date, to compute the n-th occurrences of a n-day for a specified year/month vectors, and finally to compute the last n-day for a specified year/month value or vector. n-days are numbered from 0 to 6 where 0 correspond to the Sunday and 6 to the Saturday.

Value

```

timeLastDayInMonth
timeFirstDayInMonth
timeLastDayInQuarter
timeFirstDayInQuarter
timeNdayOnOrAfter
timeNdayOnOrBefore
timeNthNdayInMonth
timeLastNdayInMonth

```

these functions return `timeDate` objects on special dates. For the functions `timeLastDayInMonth`

and `timeLastDayInMonth` return the last or first day respectively in a given month and year. The same functionality for quarterly time horizons is returned by the functions `timeLastDayInQuarter` and `timeLastDayInQuarter`. For the function `timeNdayOnOrAfter` the date in the specified month that is a n-day (e.g. Sun-day) on or after a given date, for `timeNdayOnOrBefore` the date that is a n-day on or before a given date, for `timeNthNdayInMonth` the nth occurrence of a n-day ($n = 1, \dots, 5$) in year, month, and for `timeLastNdayInMonth` the last nday in year, month will be returned as "timeDate" objects.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

`timeDate`

Examples

```
## SOURCE ("fCalendar.3D-TimeDateSpecDates")

## Date as character String:
charvec = "2006-04-16"

## timeLastDayInMonth-
# What date has the last day in a month for a given date ?
timeLastDayInMonth(charvec, format = "%Y-%m-%d", zone = myFinCenter, FinCenter = myFinCenter)
timeLastDayInMonth(charvec)
timeLastDayInMonth(charvec, FinCenter = "Zurich")

## timeFirstDayInMonth -
# What date has the first day in a month for a given date ?
timeFirstDayInMonth(charvec)

## timeLastDayInQuarter -
# What date has the last day in a quarter for a given date ?
timeLastDayInQuarter(charvec)

## timeFirstDayInQuarter -
# What date has the first day in a quarter for a given date ?
timeFirstDayInQuarter(charvec)

## timeNdayOnOrAfter
# What date has the first Monday on or after March 15, 1986 ?
timeNdayOnOrAfter("1986-03-15", 1)

## timeNdayOnOrBefore
# What date has Friday on or before April 22, 1977 ?
timeNdayOnOrBefore("1986-03-15", 5)

## timeNthNdayInMonth -
# What date is the second Monday in April 2004 ?
timeNthNdayInMonth("2004-04-01", 1, 2)

## timeLastNdayInMonth -
# What date has the last Tuesday in May, 1996 ?
timeLastNdayInMonth("1996-05-01", 2)
```

TimeDateCoercion *timeDate Class, Coercion and Transformation*

Description

A collection and description of functions and methods for coercion and transformation of objects of class 'timeDate'.

The functions are:

<code>as.timeDate.POSIXt</code>	Returns a 'POSIX' object as 'timeDate' object,
<code>as.timeDate.Date</code>	Returns a 'POSIX' object as 'timeDate' object,
<code>as.character.timeDate</code>	Returns a 'timeDate' object as character string,
<code>as.data.frame.timeDate</code>	Returns a 'timeDate' object as data frame,
<code>as.POSIXct.timeDate</code>	Returns a 'timeDate' object as POSIXct object,
<code>julian.timeDate</code>	Returns Julian day counts since 1970-01-01,
<code>atoms.timeDate</code>	Returns date/time atoms from a 'timeDate' object,
<code>months.timeDate</code>	Extract months atom from a 'timeDate' object.

Usage

```
## Default S3 method:
as.timeDate(x, zone = myFinCenter, FinCenter = myFinCenter)
## S3 method for class 'POSIXt':
as.timeDate(x, zone = myFinCenter, FinCenter = myFinCenter)
## S3 method for class 'Date':
as.timeDate(x, zone = myFinCenter, FinCenter = myFinCenter)

## S3 method for class 'timeDate':
as.character(x, ...)
## S3 method for class 'timeDate':
as.double(x, units = c("auto", "secs", "mins", "hours", "days", "weeks"), ...)
## S3 method for class 'timeDate':
as.data.frame(x, ...)
## S3 method for class 'timeDate':
as.POSIXct(x, tz = "")
## S3 method for class 'timeDate':
as.POSIXlt(x, tz = "")
## S3 method for class 'timeDate':
as.Date(x, method = c("trunc", "round", "next"), ...)

## S3 method for class 'timeDate':
julian(x, origin = timeDate("1970-01-01"), units = c("auto",
  "secs", "mins", "hours", "days", "weeks"), zone = NULL, FinCenter = NULL, ..)
## S3 method for class 'timeDate':
atoms(x, ...)
## S3 method for class 'timeDate':
months(x, abbreviate = NULL)
```

Arguments

<code>abbreviate</code>	<code>[months]</code> - currently not used.
<code>tz</code>	inputs the time zone to POSIX objects, i.e. the time zone, zone, or financial center string, <code>FinCenter</code> , as used by <code>timeDate</code> objects.
<code>FinCenter</code>	a character with the the location of the financial center named as "continent/city".
<code>method</code>	a character string denoting the method how to determine the dates.
<code>origin</code>	a length-one object inheriting from class " <code>timeDate</code> " setting the origin for the julian counter.
<code>units</code>	a character string denoting the date/time units in which the results are desired.
<code>x</code>	an object of class <code>timeDate</code> .
<code>zone</code>	the time zone or financial center where the data were recorded.
<code>...</code>	arguments passed to other methods.

Value

`as.character`
`as.data.frame`
 return a `timeDate` object trnasformed into a character or a data frame formatted object.

`as.POSIXct`
 return a `timeDate` object transformed into a POSIX type formatted object.

`julian`
 return a `timeDate` object as a Julian count.

`atoms`
`months`
 extrac from a `timeDate` object the calendar atoms, i.e, the year, month, day, and optionally hour, minute and second.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

`timeDateClass`

Examples

```
## SOURCE("fCalendar.3E-TimeDateCoercion")

## as.character
tD = timeDate()
as.character(tD)

## as.data.frame -
as.data.frame(tD)
```

```
## as.POSIXct -
  as.POSIXct(tD)

## julian -
  tC = timeCalendar()
  julian(tC)[1:3]

## atoms -
  atoms(tC)

## months -
  months(tC)
```

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>returnSeries</code>	Computes returns from a 'timeSeries' object,
<code>durationSeries</code>	Computes durations from a 'timeSeries' object,
<code>midquoteSeries</code>	Computes mid quotes from a 'timeSeries' object,
<code>spreadSeries</code>	Computes spreads from a 'timeSeries' object,
<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.

Functions to print and plot 'timeSeries' objects:

<code>print.timeSeries</code>	S3: Print method for a 'timeSeries' object,
<code>summary.timeSeries</code>	S3: Summary method for a 'timeSeries' object,
<code>plot.timeSeries</code>	S3: Plot method for a 'timeSeries' object,
<code>lines.timeSeries</code>	S3: Lines method for a 'timeSeries' object,
<code>points.timeSeries</code>	S3: Points method for a 'timeSeries' object.

Special Functions for daily 'timeSeries' objects:

<code>dummyDailySeries</code>	Creates a dummy daily 'timeSeries' object,
-------------------------------	--

alignDailySeries Aligns a daily 'timeSeries' to new positions,
 ohlcDailyPlot Plots open high low close bar chart.

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, ...)
returnSeries(x, type = c("continuous", "discrete"), percentage = FALSE,
  trim = TRUE, digits = 4, units = NULL)
durationSeries(x, trim = FALSE, units = c("secs", "mins", "hours"))
midquoteSeries(x, which = c("Bid", "Ask"))
spreadSeries(x, which = c("Bid", "Ask"), tickSize = NULL)

applySeries(x, from = NULL, to = NULL, by = c("monthly", "quarterly"),
  FUN = colAvgs, 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)

## S3 method for class 'timeSeries':
print(x, recordIDs = FALSE, ...)
## S3 method for class 'timeSeries':
summary(object, ...)
## S3 method for class 'timeSeries':
plot(x, ...)
## S3 method for class 'timeSeries':
lines(x, ...)
## S3 method for class 'timeSeries':
points(x, ...)

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)
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

by [applySeries] -
 a character either "monthly" or "quarterly". The default value is "monthly".
 Only operative when both arguments from and to have their default values

	NULL. In this case the function FUN will be applied to monthly or quarterly periods.
charvec	a character vector of dates and times.
colOrder	[ohlcDailyPlot] - an integer vector which gives the order of the prices and the volume in the input object. By default the following order of columns from 1 to 5 is assumed: Open, high, low, close, and volume.
data	a <code>data.frame</code> or a <code>matrix</code> object of numeric data.
digits	[returnSeries] - an integer value. The number of digits to be printed in the output.
documentation	optional documentation string, or a vector of character strings.
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	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.
from, to	starting date and end date, <code>to</code> must be after <code>from</code> .
FUN	[applySeries] - a function to use for aggregation, by default <code>colAves</code> .
grid.lty, grid.nx	[ohlcDailyPlot] - The type of grid line and the number of grid lines used in the plot.
include.weekends	[alignDailySeries] - a logical value. Should weekend dates be included or removed from the series.
tickSize	[spreadSeries] - the default is NULL 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.
main	[ohlcDailyPlot] - a character string to title the price and volume plot.
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> .
percentage	[returnSeries] - a logical value. By default FALSE, if TRUE the series will be expressed in percentage changes.

recordIDs	a data frame which can be used for record identification information. [print] - a logical value. Should the recordIDs printed together with the data matrix and time series positions?
sep	[readSeries] - the field separator used in the spreadsheet file to separate columns.
title	an optional title string, if not specified the inputs data name is departed.
trim	[diffSeries][returnSeries] - a logical value. By default TRUE, the first missing observation in the return series will be removed.
type	[returnSeries] - a character string specifying how to compute the returns. Valid choices are: continuous and discrete. For the default type="continuous", the returns are calculated as the logarithmic differences, otherwise if type="discrete", the returns are calculated as percentage changes.
units	[applySeries][lag][alignDailySeries][returnSeries][mergeSeries] - 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. [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. [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.
volume	[ohlcDailyPlot] - a logical value. Should a volume plot added to the OHLC Plot. By default TRUE.
which	[midquoteSeries][spreadSeries] - a vector with two character strings naming the column names of the time series from which to compute the mid quotes and spreads. By default these are bid and ask prices with column names c("Bid", "Ask").
x	[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 timeSeries.
xlab, ylab	[ohlcDailyPlot] - 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 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 ...
```

```
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

```
## SOURCE("fCalendar.4A-TimeSeriesClass")

## data - Data Frame:
# Microsoft Data:
MSFT.df = data.frame(matrix(c(
  20010326, 57.1250, 57.5000, 55.5625, 56.0625, 31559300,
  20010327, 56.0625, 58.5625, 55.8750, 58.2500, 47567800,
```

```

20010328, 57.3750, 57.9375, 55.3750, 55.5625, 39340800,
20010329, 55.3750, 57.1875, 54.5625, 55.3750, 43492500,
20010330, 55.7500, 56.1875, 53.8750, 54.6875, 45600800,
20010402, 54.8125, 56.9375, 54.6250, 55.8125, 37962000,
20010403, 55.3125, 55.3125, 52.7500, 53.3750, 47093800,
20010404, 53.3750, 55.0000, 51.0625, 51.9375, 52023300,
20010405, 53.7500, 57.3750, 53.5000, 56.7500, 56682000,
20010406, 56.3750, 57.1875, 55.0625, 56.1875, 46311000,
20010409, 56.5700, 57.4200, 55.6600, 57.1500, 28147800,
20010410, 57.9500, 60.0900, 57.7800, 59.6800, 54599700,
20010411, 60.6500, 61.5000, 59.7000, 60.0400, 54939800,
20010412, 59.5600, 62.3100, 59.3500, 62.1800, 43760000,
20010416, 61.4000, 61.5800, 60.1200, 60.7900, 32928700,
20010417, 60.5200, 62.1100, 60.0400, 61.4800, 42574600,
20010418, 63.3900, 66.3100, 63.0000, 65.4300, 78348200,
20010419, 65.8100, 69.0000, 65.7500, 68.0400, 79687800,
20010420, 70.3000, 71.1000, 68.5000, 69.0000, 96459800,
20010423, 68.1100, 68.4700, 66.9000, 68.2500, 46085600,
20010424, 68.2000, 69.9300, 67.1400, 67.5500, 44588300,
20010425, 67.5700, 69.7900, 67.2500, 69.6900, 38372000,
20010426, 70.0700, 71.0000, 68.2500, 69.1300, 59368800,
20010427, 69.5300, 69.6800, 66.2100, 67.1200, 60786200,
20010430, 68.5300, 69.0600, 67.6800, 67.7500, 37184100,
20010501, 67.6600, 70.3000, 67.6000, 70.1700, 41851400,
20010502, 71.0000, 71.1500, 69.3500, 69.7600, 46432200,
20010503, 69.2500, 70.1800, 68.1400, 68.5300, 33136700,
20010504, 68.0000, 71.0500, 67.9600, 70.7500, 59769200,
20010507, 70.8300, 72.1500, 70.7000, 71.3800, 54678100),
byrow = TRUE, ncol = 6))
colnames(MSFT.df) = c("YMMDD", "Open", "High", "Low", "Close", "Volume")

# Load Microsoft Data:
myFinCenter <- "GMT"
MSFT = as.timeSeries(MSFT.df)
head(MSFT)

## timeSeries - Conversion:
# 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))

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

## Return Series -
# Compute Returns:
args(returnSeries)
# Continuous Returns:
returnSeries(tsApril01)

```

```

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

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

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

## 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)

```

TimeSeriesData

timeSeries Class, Functions and Methods

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 and methods to modify 'timeSeries' objects:

diff	Takes differences from a 'timeSeries' object,
lag	Lags a 'timeSeries' object,
merge	Merges two 'timeSeries' objects,
rbinds	Binds rows of two 'timeSeries' objects,
cumsum	Returns the cumulated sum of a 'timeSeries' object,
lag	Scales and centers a 'timeSeries' object.

Functions and methods for classifying 'timeSeries' objects:

var	Computes Variance from a 'timeSeries' object.
-----	---

Functions and methods for mathematical operations on 'timeSeries' objects:

<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>quantile.timeSeries</code>	Computes quantiles of a 'timeSeries' object.

Functions and methods for subsetting 'timeSeries' objects:

<code>"["</code>	"[" method for a 'timeSeries' object,
<code>cut</code>	Cuts a block from a 'timeSeries' object,
<code>head</code>	Returns the head of a 'timeSeries' object,
<code>tail</code>	Returns the tail of a 'timeSeries' object,
<code>outliers</code>	Removes outliers from a 'timeSeries' object.

Functions and methods for dimensional operations on 'timeSeries' objects:

<code>dim</code>	Returns the dimension of a 'timeSeries' object,
<code>dimnames</code>	Returns the dimension names of a 'timeSeries' object,
<code>colnames<-</code>	Assigns column names to a 'timeSeries' object,
<code>rownames<-</code>	Assigns row names to a 'timeSeries' object,
<code>is.array</code>	Allows that NCOL and NROW work properly.

Usage

```
## S3 method for class 'timeSeries':
diff(x, lag = 1, diff = 1, trim = FALSE, pad = NA, ...)
## S3 method for class 'timeSeries':
lag(x, k = 1, trim = FALSE, units = NULL, ...)
## S3 method for class 'timeSeries':
merge(x, y, units = NULL, ...)
## S3 method for class 'timeSeries':
rbind(x, y)
## S3 method for class 'timeSeries':
cumsum(x)
## S3 method for class 'timeSeries':
scale(x, center = TRUE, scale = TRUE)
## S3 method for class 'timeSeries':
var(x, y = NULL, na.rm = FALSE, use)

## 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':
quantile(x, probs = 0.95, ...)

## 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':
cut(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':
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

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 x .
complement	[outlierSeries] - a logical flag, should the outlier series or its complement be returns, by default TRUE which returns the series free of outliers.
diff	[diffSeries] - an integer indicating the order of the difference. By default 1.
dimnames	[as.timeSeries] - a logical, if TRUE the dimension names of the matrix are assigned to the time series object.
e1, e2	[Ops] - two objects of class <code>timeSeries</code> .
from, to	starting date and end date, <code>to</code> must be after <code>from</code> .
i, j	[[""] - index arguments used for subsettings.
k	[lagSeries] - an integer value. The number of lags (in units of observations). By default 1.

lag	[diffSeries] - an integer indicating which lag to use. By default 1.
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".
n	[head][tail] - an integer specifying the number of lines to be returned. By default n=6.
na.rm	[var] - a logical flag. Should missing values be removed? By default FALSE.
pad	[diffSeries] - which value should get the padded values? By default NA. Another choice often used would be zero.
probs	a numeric value or numeric vector of probabilities with values in [0, 1].
recordIDs	[head][tail] - a logical value. Should the recordIDs returned together with the data matrix and time series positions?
sd	[outlierSeries] - a numeric value of standard deviations, e.g. 10 means that values larger or smaller than ten times the standard deviation will be removed from the series.
trim	[diffSeries][returnSeries] - a logical value. By default TRUE, the first missing observation in the return series will be removed.
units	[applySeries][lag][alignDailySeries][returnSeries][mergeSeries] - 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. [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.
use	[var] - 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".
value	[colnames<-][rownames<-] - a valid value for column names component of dimnames(x). For a "timeSeries" object this is either NULL or a character vector of length the column dimension. Not, row names cannot be assigned for a "timeSeries" object, the function rownames() will stop and return an error message.
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 timeSeries.
y	[mergeSeries][var] - a matrix type object to be merged with an object of class timeSeries. Must have the same number of rows.

... 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`
`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`.

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

```
## SOURCE("fCalendar.4B-TimeSeriesData")

## 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
par(ask = FALSE)
plot(TS, type = "o", pch = 19)

## diff | scale -
# Difference and scale the timeSeries Object:
scale(diff(TS, lag = 2, trim = TRUE))

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

## merge -
# Merge two timeSeries Object:
merge(TS, diff(log(TS), units = c("ts", "diffLog")))

## var -
# Variance of a timeSeries Object:
var(returnSeries(TS, percentage = TRUE))

## Ops | +/- * ^ ...
# Mathematical Operations:
TS^2
TS[2:4]
OR = returnSeries(TS)
OR
OR > 0
quantile(OR, probs = c(0.90, 0.95))
```

```
TimeSeriesPositions
      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>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':
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

<code>method</code>	<code>[alignDailySeries]</code> - 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>object</code>	<code>[is][seriesData][seriesPositions][summary]</code> - an object of class <code>timeSeries</code> .
<code>value</code>	a valid value for that component of <code>newPositions(x)</code> , i.e. an object of class "timeDate" with appropriate length.
<code>x</code>	<code>[as]</code> - a matrix type object to be converted. <code>[as.vector][as.matrix][as.data.frame]</code> -

[applySeries] -
 [cut][end][mergeSeries][plot][print][rev][start] -
 an object of class `timeSeries`.
 ... 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

```
## SOURCE("fCalendar.4C-TimeSeriesPositions")

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

## seriesPositions -
seriesPositions(X)
```

TimeSeriesCoercion *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

```
## SOURCE("fCalendar.4D-TimeSeriesCoercion")

## 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)

```

HolidayDates

Public and Ecclesiastical Holidays

Description

A collection and description of functions and methods dealing with holiday dates in the G7 countries and Switzerland.

Usage

```

Septuagesima(year)
Quinquagesima(year)
AshWednesday(year)
PalmSunday(year)
GoodFriday(year)
Easter(year)
EasterSunday(year)
EasterMonday(year)
RogationSunday(year)
Ascension(year)
Pentecost(year)
PentecostMonday(year)
TrinitySunday(year)
CorpusChristi(year)
ChristTheKing(year)
Advent1st(year)
Advent2nd(year)
Advent3rd(year)
Advent4th(year)
ChristmasEve(year)
ChristmasDay(year)
BoxingDay(year)
NewYearsDay(year)
SolemnityOfMary(year)
Epiphany(year)
PresentationOfLord(year)

```

Annunciation (year)
TransfigurationOfLord (year)
AssumptionOfMary (year)
BirthOfVirginMary (year)
CelebrationOfHolyCross (year)
MassOfArchangels (year)
AllSaints (year)
AllSouls (year)
LaborDay (year)
CHBerchtoldsDay (year)
CHSechselaeuten (year)
CHAscension (year)
CHConfederationDay (year)
CHKnabenschiessen (year)
GBMayDay (year)
GBBankHoliday (year)
GBSummerBankHoliday (year)
GBMilleniumDay (year)
DEAscension (year)
DECorpusChristi (year)
DEGermanUnity (year)
DEChristmasEve (year)
DENewYearsEve (year)
FRFetDeLaVictoire1945 (year)
FRAscension (year)
FRBastilleDay (year)
FRAssumptionVirginMary (year)
FRAllSaints (year)
FRArmisticeDay (year)
ITEpiphany (year)
ITLiberationDay (year)
ITAssumptionOfVirginMary (year)
ITAllSaints (year)
ITStAmrose (year)
ITImmaculateConception (year)
USDecorationMemorialDay (year)
USPresidentsDay (year)
USNewYearsDay (year)
USInaugurationDay (year)
USMLKingsBirthday (year)
USLincolnsBirthday (year)
USWashingtonsBirthday (year)
USMemorialDay (year)
USIndependenceDay (year)
USLaborDay (year)
USColumbusDay (year)
USElectionDay (year)
USVeteransDay (year)
USThanksgivingDay (year)
USChristmasDay (year)
USCPulaskisBirthday (year)
USGoodFriday (year)

CAVictoriaDay (year)
CACanadaDay (year)
CACivicProvincialHoliday (year)
CALabourDay (year)
CATHanksgivingDay (year)
CaRemembranceDay (year)
JPNewYearsDay (year)
JPGantan (year)
JPBankHolidayJan2 (year)
JPBankHolidayJan3 (year)
JPComingOfAgeDay (year)
JPSeijinNoHi (year)
JPNatFoundationDay (year)
JPKenkokuKinenNoHi (year)
JPGreeneryDay (year)
JPMidoriNoHi (year)
JPConstitutionDay (year)
JPKenpouKinenBi (year)
JPNationHoliday (year)
JPKokuminNoKyujitu (year)
JPChildrensDay (year)
JPKodomoNoHi (year)
JPMarineDay (year)
JPUmiNoHi (year)
JPRespectForTheAgedDay (year)
JPKeirouNOhi (year)
JPAutumnalEquinox (year)
JPShuubunNoHi (year)
JPHealthandSportsDay (year)
JPTaiikuNoHi (year)
JPNationalCultureDay (year)
JPBunkaNoHi (year)
JPThanksgivingDay (year)
JPKinrouKanshaNoHi (year)
JPEmperorsBirthday (year)
JPTennouTanjyouBi (year)
JPBankHolidayDec31 (year)

Arguments

`year` an integer value or vector of year numbers including the century. These are integers of the form CCYY, e.g. 2000.

Value

Returns an ISO-8601 formatted calendar date CCYYMMDD, an object of class "sdate" which represents an integer value or vector.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## SOURCE("fCalendar.5C-HolidayDates")

## CHSechselaeuten -
# Sechselaeuten a half Day Bank Holiday in Switzerland
CHSechselaeuten(2000:2010)
CHSechselaeuten(currentYear)

## German Unification Day:
DEGermanUnity(currentYear)
```

HolidayCalendars *Holiday Calendars and Utilities*

Description

A collection and description of functions and methods dealing with holiday dates and holiday calendars.

The functions for the generation of holiday calendars are:

holiday	Returns a holiday date of G7 and CH as 'sdate',
holidayNYSE	Computes 'timeDate' object for full-day NYSE holidays,
holidayZURICH	Computes 'timeDate' object for Zurich holidays.

Usage

```
holiday(year, Holiday = Easter)
holidayNYSE(year = currentYear)
holidayZURICH(year = currentYear)
```

Arguments

Holiday	the unquoted function name of an ecclestial or public holiday in the G7 countries or Switzerland, see the list below.
year	an integer vector of years, [CCYY]. [day.of.week] - cr vector of integer year numbers.

Details**Holiday Calendar:**

Easter is the central ecclestial holiday. Many other holidays are related to this feast. The function `easter` computes the dates of easter and related ecclestial holidays for the requested year vector. `holiday` calculates the dates of ecclestial or public holidays in the G7 countries, e.g. `holiday(2003, "GoodFriday")`. `holidays.db` is our Data Base of World Wide Holidays. The database contains holiday functions automatically loaded by the `fBasics` library at startup time. The user can add additional holiday functions to this data base. The information for the holiday data base were collected from several web pages about holiday calendars. The following ecclestial and public `[HOLIDAY]` functions in the G7 countries and Switzerland are available:

Holidays Related to Easter:

Septuagesima, Quinquagesima, AshWednesday, PalmSunday, GoodFriday, EasterSunday, Easter, EasterMonday, RogationSunday, Ascension, Pentecost, PentecostMonday, TrinitySunday CorpusChristi.

Holidays Related to Christmas:

ChristTheKing, Advent1st, Advent1st, Advent3rd, Advent4th, ChristmasEve, ChristmasDay, BoxingDay, NewYearsDay.

Other Ecclesiastical Feasts:

SolemnityOfMary, Epiphany, PresentationOfLord, Annunciation, TransfigurationOfLord, AssumptionOfMary, AssumptionOfMary, BirthOfVirginMary, CelebrationOfHolyCross, MassOfArchangels, AllSaints, AllSouls.

CHZurich - Public Holidays:

CHBerchtoldsDay, CHSechselaeuten, CHAscension, CHConfederationDay, CHKnabenschuessen.

GBLondon - Public Holidays:

GBMayDay, GBBankHoliday, GBSummerBankHoliday, GBNewYearsEve.

DEFrankfurt - Public Holidays:

DEAscension, DECorpusChristi, DEGermanUnity, DEChristmasEve, DENewYearsEve.

FRParis - Public Holidays:

FRFetDeLaVictoire1945, FRAscension, FRBastilleDay, FRAssumptionVirginMary, FRAllSaints, FRArmisticeDay.

ITMilano - Public Holidays:

ITEpiphany, ITLiberationDay, ITRepublicAnniversary, ITAssumptionOfVirginMary, ITAllSaints, ITWWIVictoryAnniversary, ITStAmrose, ITImmaculateConception.

USNewYork/USChicago - Public Holidays:

USNewYearsDay, USInaugurationDay, USMLKingsBirthday, USLincolnsBirthday, USWashingtonsBirthday, USMemorialDay, USIndependenceDay, USLaborDay, USColumbusDay, USElectionDay, USVeteransDay, USThanksgivingDay, USChristmasDay, USCPulaskisBirthday, USGoodFriday.

CAToronto/CAMontreal - Public Holidays:

CAVictoriaDay, CACanadaDay, CACivicProvincialHoliday, CALabourDay, CAThanksgivingDay, CaRemembranceDay.

JPTokyo/JPOsaka - Public Holidays:

JPNewYearsDay, JPGantan, JPBankHolidayJan2, JPBankHolidayJan3, JPComingOfAgeDay, JPSeijinNoHi, JPNatFoundationDay, JPKenkokuKinenNoHi, JPGreeneryDay, JPMidoriNoHi, JPConstitutionDay, JPKenpouKinenBi, JPNationHoliday, JPKokuminNoKyujitu, JPChildrensDay, JPKodomoNoHi, JPMarineDay, JPUmiNoHi, JPRespectForTheAgedDay, JPKeirouNoHi, JPAutumnalEquinox, JPShuubun-no-hi, JPHealthandSportsDay, JPTaiikuNoHi, JPNationalCultureDay, JPBunkaNoHi, JPThanksgivingDay, JPKinrouKanshaNohi, JPKinrou-kansha-no-hi, JPEmperorsBirthday, JPTennou-tanjyou-bi, JPTennou-tanjyou-bi.

All the holiday functions are listed in the data file `holidays.R`. Additional holidays, which are not yet available there, can be added to this data base file.

Value

holiday

holidayNYSE holidayZURICH

return an ISO-8601 formatted calendar date `CCYYMMDD`, an object of class `"sdate"` which represents an integer value or vector.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

References

ISO-8601 (1988); *Data Elements and Interchange Formats - Information Interchange, Representation of Dates and Time*, International Organization for Standardization, Reference Number ISO 8601, 14 pages.

Examples

```
## SOURCE("fCalendar.5B-HolidayCalendars")

## easter -
# Dates for GoodFriday from 2000 until 2010:
holiday(2000:2010, "GoodFriday")

## holidays -
Easter(2000:2010)
GoodFriday(2000:2010)

## holidayNYSE -
holidayNYSE(currentYear)
```

TimeSeriesImport *Import Market Data from the Internet*

Description

A collection and description of functions to import financial and economic market data from the Internet. Download functions are available for economic and financial market data from Economagic's, from Yahoo's, from the Federal Reserve's, and from the the forecasts.org Internet sites.

The functions are:

economagicImport	Economic series from Economagic's Web site,
yahooImport	daily stock market data from Yahoo's Web site,
yahooSeries	easy to use download from Yahoo,
keystatsImport	key statistics from Yahoo's Web site,
fredImport	time series from St. Louis FRED Web site,
forecastsImport	monthly data from the Financial Forecast Center.

Usage

```
economagicImport(query, file = "tempfile",
  source = "http://www.economagic.com/em-cgi/data.exe/",
  frequency = c("quarterly", "monthly", "daily"), save = FALSE,
  colname = "VALUE", try = TRUE)
yahooImport(query, file = "tempfile",
  source = "http://chart.yahoo.com/table.csv?", save = FALSE,
```

```

    sep = ";", swap = 20, try = TRUE)
keystatsImport(query, file = "tempfile",
  source = "http://finance.yahoo.com/q/ks?s=", save = FALSE,
  try = TRUE)
fredImport(query, file = "tempfile",
  source = "http://research.stlouisfed.org/fred2/series/",
  frequency = "daily", save = FALSE, sep = ";", try = TRUE)
forecastsImport(query, file = "tempfile",
  source = "http://www.forecasts.org/data/data/", save = FALSE,
  try = TRUE)

yahooSeries(symbols = c("^DJI", "IBM"), from = NULL, to = NULL,
  nDaysBack = 365, quote = c("Open", "High", "Low", "Close", "Volume"),
  aggregation = c("d", "w", "m"), returnClass = c("timeSeries", "ts",
  "matrix", "data.frame"), getReturns = FALSE, ...)

show.fWEBDATA(object)
## S3 method for class 'keystats':
print(x, ...)

```

Arguments

aggregation	[yahooSeries] - a character string denoting the aggregation level of the downloaded data records, d for daily, w for weekly and m for monthly data records.
colname	[economagicImport] - a character string which defines the name of the value column. By default "VALUE".
file	a character string with filename, usually having extension ".csv", where to save the downloaded data.
frequency	a character string, one of "quarterly", "monthly", or "daily", defining the frequency of the data records.
from, to	[yahooSeries] - an ISO-8601 formatted character string of the starting (end) date, e.g. "2005-01-01".
getReturns	[yahooSeries] - a logical flag. Should return values be computed using the function returnSeries?
nDaysBack	[yahooSeries] - an integer giving the length of the download period in number of days starting n days back from today. Only in use if from and to are not specified.
object	an S4 object of class "fWEBDATA".
query	a character string, denoting the location of the data at the web site.
save	a logical value, if set to TRUE the downloaded data file will be stored under the path and file name specified by the string file. By default FALSE.
quote	[yahooSeries] - a character value or vector of strings giving the column name(s) of those instruments to be extracted from the download.
returnClass	[yahooSeries] - a character string naming the class of the object to be returned. By default the

	function <code>yahooSeries</code> returns a "timeSeries" object, alternatives are: "ts", "matrix", or "data.frame".
<code>sep</code>	a character value, defining the field separator for the destination file which saves the downloaded data records. By default a semicolon.
<code>source</code>	a character string with the download URL.
<code>symbols</code>	[<code>yahooSeries</code>] - a character string value or vector, the Yahoo symbol name(s).
<code>swap</code>	[<code>yahooImport</code>] - an integer value which determines when we swap from the 19th to 20th century, by default 20, i.e. we swap 1920. This is necessary since Yahoo does not list the century in its dates, e.g. "15-Aug-02".
<code>try</code>	a logical value, if set to TRUE the Internet access will be checked.
<code>x</code>	[<code>print.keystats</code>] - an object of class <code>keystats</code> as returned by the function <code>keystatsImport</code> .
<code>...</code>	optional arguments to be passed.

Details

Import data from www.economagic.com

Frequently requested data files from Economagic for the US economy include:

[query]	Description:
<code>var/leading-ind-long</code>	Index of Leading Economic Indicators
<code>beana/t102101</code>	Real Gross Domestic Product
<code>fedstl/trsp500</code>	SP 500 Total Return
<code>fedstl/gnp</code>	Gross National Product in Current Dollars
<code>var/cpiu-long</code>	Consumer Price Index - All Urban Consumers
<code>feddal/ru</code>	Unemployment Rate
<code>fedstl/indpro</code>	Total Industrial Production Index
<code>fedstl/exjpus+2</code>	FX Rate: Japanese Yen to one US Dollar
<code>fedstl/fedfunds+2</code>	Federal Funds Rate
<code>fedstl/mdiscrt+2</code>	Discount Rate
<code>fedbog/tcm30y+2</code>	30-Year Treasury Constant Maturity Rate
<code>fedstl/mprime+2</code>	Bank Prime Loan Rate
<code>fedstl/tb3ms+2</code>	3-Month Treasury Bills - Secondary Market
<code>fedstl/tb6ms+2</code>	6-Month Treasury Bills - Secondary Market
<code>fedbog/cm+2</code>	30 Year Federal Home Loan Mortgages
<code>var/west-texas-crude-long</code>	Price of West Texas Intermediate Crude

Import data from chart.yahoo.com:

The query string is given as

```
s=SYMBOL&a=DD&b=MM&c=CCYY&g=d&q=q&z=SYMBOL&x=.csv
```

where `SYMBOL` has to be replaced by the symbol name of the instrument, and `DD`, `MM`, and `CCYY` by the day, month-1 and century/year when the time series should start.

Here are some examples of symbols:

[query]	Description:
^DJI	Dow Jones 30 Industrial Averages
^NYA	New York Stock Exchange Composite
^NDX	Nasdaq 100 Index
^IXIC	Nasdaq Composite Index
^TYX	US 30Y Treasury Bond Index
IBM	BM DJIA Stock
KO	Coca-Cola DJIA Stock

The meaning of the tokens in the query string are the following:

Token	Description
s	Selected Ticker-Symbol
a	First Quote starts with Month (mm)
b	First Quote starts with Day (dd)
c	First Quote starts with Year (ccyy)
d	Last Quote ends with Month (mm)
e	Last Quote ends with Day (dd)
f	Last Quote ends with Year (ccyy)
z	Selected Ticker-Symbol

Note, that month tokens range between 0 and 11 for January to December!

Value

The functions `economagicImport`, `fredImport`, and `yahooImport` return an S4 object of class `fWEBDATA` with the following slots:

<code>@call</code>	the function call.
<code>@data</code>	the data as downloaded formatted as a <code>data.frame</code> .
<code>@param</code>	a character vector whose elements contain the values of selected parameters of the argument list.
<code>@title</code>	a character string with the name of the download. This can be overwritten specifying a user defined input argument.
<code>@description</code>	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The function `yahooSeries` returns by default an S4 object of class `timeSeries` or alternatively an object specified by the function argument `returnClass`.

Note

Internet Download:

Note, that if the service provider changes the data file format it may become necessary to modify and update the functions.

The R package `tseries` from Adrian Trapletti offers an alternative function to download stock market data and indexes from Yahoo's Internet site.

Functions added to Splus Version:

These functions are only used when running Rmetrics under SPlus. Note, that only part of the functionality is supported which is required running these R functions for Rmetrics under Splus.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## Not run:
## SOURCE("fcALENDAR.6A-TimeSeriesImport")

## economagicImport -
USDEUR = economagicImport(query = "fedny/day-fxus2eu",
  frequency = "daily", colname = "USDEUR")
# Print Data Slot if Internet Download was Successful:
if (!is.null(USDEUR)) print(USDEUR@data[1:20, ])

## economagicImport -
USFEDFUNDS = economagicImport(query = "fedst1/fedfunds+2",
  frequency = "monthly", colname = "USFEDFUNDS")
if (!is.null(USFEDFUNDS)) print(USFEDFUNDS@data[1:20, ])

## economagicImport -
USGNP = economagicImport(query = "fedst1/gnp",
  frequency = "quarterly", colname = "USGNP")
if (!is.null(USGNP)) print(USGNP@data[1:20, ])

## yahooImport -
# [test 19/20 century change 01-12-1999 -- 31-01-2000]
query = "s=IBM&a=11&b=1&c=1999&d=0&q=31&f=2000&z=IBM&x=.csv"
IBM = yahooImport(query)
if (!is.null(IBM)) print(IBM@data[1:20, ])

## fredImport -
DPRIME = fredImport("DPRIME")
if (!is.null(DPRIME)) print(DPRIME@data[1:20, ])
## End(Not run)
```

Index

*Topic **chron**

- HolidayCalendars, 40
- HolidayDates, 37
- TimeDateClass, 4
- TimeDateCoercion, 18
- TimeDateMathOps, 13
- TimeDateSpecDates, 16
- TimeDateSubsets, 10
- TimeSeriesClass, 21
- TimeSeriesCoercion, 35
- TimeSeriesData, 27
- TimeSeriesPositions, 32

*Topic **data**

- DaylightSavingTime, 3
- TimeSeriesImport, 42

*Topic **programming**

- RmetricsUtilities, 1
- + .timeDate (*TimeDateMathOps*), 13
- .timeDate (*TimeDateMathOps*), 13
- [.timeDate (*TimeDateSubsets*), 10
- [.timeSeries (*TimeSeriesData*), 27
- abs.timeSeries (*TimeSeriesData*), 27
- Adelaide (*DaylightSavingTime*), 3
- Advent1st (*HolidayDates*), 37
- Advent2nd (*HolidayDates*), 37
- Advent3rd (*HolidayDates*), 37
- Advent4th (*HolidayDates*), 37
- Algiers (*DaylightSavingTime*), 3
- align (*RmetricsUtilities*), 1
- alignDailySeries (*TimeSeriesClass*), 21
- AllSaints (*HolidayDates*), 37
- AllSouls (*HolidayDates*), 37
- Amsterdam (*DaylightSavingTime*), 3
- Anchorage (*DaylightSavingTime*), 3
- Andorra (*DaylightSavingTime*), 3
- Annunciation (*HolidayDates*), 37
- applySeries (*TimeSeriesClass*), 21
- as.character.timeDate (*TimeDateCoercion*), 18
- as.data.frame.timeDate (*TimeDateCoercion*), 18

- as.data.frame.timeSeries (*TimeSeriesCoercion*), 35
- as.Date.timeDate (*TimeDateCoercion*), 18
- as.double.timeDate (*TimeDateCoercion*), 18
- as.matrix.mts (*RmetricsUtilities*), 1
- as.matrix.timeSeries (*TimeSeriesCoercion*), 35
- as.matrix.ts (*RmetricsUtilities*), 1
- as.POSIXct.timeDate (*TimeDateCoercion*), 18
- as.POSIXlt (*RmetricsUtilities*), 1
- as.POSIXlt.timeDate (*TimeDateCoercion*), 18
- as.timeDate (*TimeDateCoercion*), 18
- as.timeSeries (*TimeSeriesCoercion*), 35
- as.ts.timeSeries (*TimeSeriesCoercion*), 35
- as.vector.timeSeries (*TimeSeriesCoercion*), 35
- Ascension (*HolidayDates*), 37
- AshWednesday (*HolidayDates*), 37
- AssumptionOfMary (*HolidayDates*), 37
- Athens (*DaylightSavingTime*), 3
- atoms (*RmetricsUtilities*), 1
- atoms.timeDate (*TimeDateCoercion*), 18
- Auckland (*DaylightSavingTime*), 3
- Bahrain (*DaylightSavingTime*), 3
- Bangkok (*DaylightSavingTime*), 3
- Beirut (*DaylightSavingTime*), 3
- Belfast (*DaylightSavingTime*), 3
- Belgrade (*DaylightSavingTime*), 3
- Berlin (*DaylightSavingTime*), 3
- BirthOfVirginMary (*HolidayDates*), 37
- blockEnd (*TimeDateSubsets*), 10
- blockStart (*TimeDateSubsets*), 10

- Bogota (*DaylightSavingTime*), 3
- BoxingDay (*HolidayDates*), 37
- Bratislava (*DaylightSavingTime*), 3
- Brisbane (*DaylightSavingTime*), 3
- Brussels (*DaylightSavingTime*), 3
- Bucharest (*DaylightSavingTime*), 3
- Budapest (*DaylightSavingTime*), 3
- BuenosAires (*DaylightSavingTime*), 3
- c.timeDate (*TimeDateMathOps*), 13
- CACanadaDay (*HolidayDates*), 37
- CACivicProvincialHoliday (*HolidayDates*), 37
- Cairo (*DaylightSavingTime*), 3
- CALabourDay (*HolidayDates*), 37
- Calcutta (*DaylightSavingTime*), 3
- Caracas (*DaylightSavingTime*), 3
- CaRemembranceDay (*HolidayDates*), 37
- Casablanca (*DaylightSavingTime*), 3
- CAThanksgivingDay (*HolidayDates*), 37
- CAVictoriaDay (*HolidayDates*), 37
- Cayman (*DaylightSavingTime*), 3
- CelebrationOfHolyCross (*HolidayDates*), 37
- CHAscension (*HolidayDates*), 37
- CHBerchtoldsDay (*HolidayDates*), 37
- CHConfederationDay (*HolidayDates*), 37
- Chicago (*DaylightSavingTime*), 3
- CHKnabenschiessen (*HolidayDates*), 37
- ChristmasDay (*HolidayDates*), 37
- ChristmasEve (*HolidayDates*), 37
- ChristTheKing (*HolidayDates*), 37
- CHSechselaeuten (*HolidayDates*), 37
- colnames<- (*RmetricsUtilities*), 1
- colnames<- .timeSeries (*TimeSeriesData*), 27
- Copenhagen (*DaylightSavingTime*), 3
- CorpusChristi (*HolidayDates*), 37
- cumsum.timeSeries (*TimeSeriesData*), 27
- currentYear (*RmetricsUtilities*), 1
- cut.timeDate (*TimeDateSubsets*), 10
- cut.timeSeries (*TimeSeriesData*), 27
- Darwin (*DaylightSavingTime*), 3
- DaylightSavingTime, 3
- DEAscension (*HolidayDates*), 37
- DEChristmasEve (*HolidayDates*), 37
- DECorpusChristi (*HolidayDates*), 37
- DEGermanUnity (*HolidayDates*), 37
- DENewYearsEve (*HolidayDates*), 37
- Denver (*DaylightSavingTime*), 3
- Detroit (*DaylightSavingTime*), 3
- diff.timeDate (*TimeDateMathOps*), 13
- diff.timeSeries (*TimeSeriesData*), 27
- difftimeDate (*TimeDateMathOps*), 13
- dim.timeSeries (*TimeSeriesData*), 27
- dimnames.timeSeries (*TimeSeriesData*), 27
- Dubai (*DaylightSavingTime*), 3
- Dublin (*DaylightSavingTime*), 3
- dummyDailySeries (*TimeSeriesClass*), 21
- durationSeries (*TimeSeriesClass*), 21
- Easter (*HolidayDates*), 37
- EasterMonday (*HolidayDates*), 37
- Eastern (*DaylightSavingTime*), 3
- EasterSunday (*HolidayDates*), 37
- economagicImport (*TimeSeriesImport*), 42
- Edmonton (*DaylightSavingTime*), 3
- end.timeDate (*TimeDateSubsets*), 10
- end.timeSeries (*TimeSeriesPositions*), 32
- Epiphany (*HolidayDates*), 37
- exp.timeSeries (*TimeSeriesData*), 27
- forecastsImport (*TimeSeriesImport*), 42
- format.timeDate (*TimeDateClass*), 4
- FRAllSaints (*HolidayDates*), 37
- Frankfurt (*DaylightSavingTime*), 3
- FRArmisticeDay (*HolidayDates*), 37
- FRAscension (*HolidayDates*), 37
- FRAssumptionVirginMary (*HolidayDates*), 37
- FRBastilleDay (*HolidayDates*), 37
- fredImport (*TimeSeriesImport*), 42
- FRFetDeLaVictoire1945 (*HolidayDates*), 37
- fWEBCDATA (*TimeSeriesImport*), 42
- fWEBCDATA-class (*TimeSeriesImport*), 42

- GBBankHoliday (*HolidayDates*), 37
- GBMayDay (*HolidayDates*), 37
- GBMilleniumDay (*HolidayDates*), 37
- GBSummerBankHoliday
(*HolidayDates*), 37
- getDayOfWeek (*TimeDateSubsets*), 10
- getDayOfYear (*TimeDateSubsets*), 10
- GoodFriday (*HolidayDates*), 37

- head.timeSeries (*TimeSeriesData*),
27
- Helsinki (*DaylightSavingTime*), 3
- holiday (*HolidayCalendars*), 40
- HolidayCalendars, 40
- HolidayDates, 37
- holidayNYSE (*HolidayCalendars*), 40
- holidayZURICH (*HolidayCalendars*),
40
- HongKong (*DaylightSavingTime*), 3
- Honolulu (*DaylightSavingTime*), 3

- Indianapolis
(*DaylightSavingTime*), 3
- is.array.timeSeries
(*TimeSeriesData*), 27
- is.timeDate (*TimeDateClass*), 4
- is.timeSeries
(*TimeSeriesCoercion*), 35
- isBizday (*TimeDateSubsets*), 10
- isHoliday (*TimeDateSubsets*), 10
- isMultivariate (*TimeSeriesClass*),
21
- Istanbul (*DaylightSavingTime*), 3
- isUnivariate (*TimeSeriesClass*), 21
- isWeekday (*TimeDateSubsets*), 10
- isWeekend (*TimeDateSubsets*), 10
- ITAllSaints (*HolidayDates*), 37
- ITAssumptionOfVirginMary
(*HolidayDates*), 37
- ITEpiphany (*HolidayDates*), 37
- ITImmaculateConception
(*HolidayDates*), 37
- ITLiberationDay (*HolidayDates*), 37
- ITStAmrose (*HolidayDates*), 37

- Jakarta (*DaylightSavingTime*), 3
- Jerusalem (*DaylightSavingTime*), 3
- Johannesburg
(*DaylightSavingTime*), 3
- JPAutumnalEquinox (*HolidayDates*),
37
- JPBankHolidayDec31
(*HolidayDates*), 37
- JPBankHolidayJan2 (*HolidayDates*),
37
- JPBankHolidayJan3 (*HolidayDates*),
37
- JPBunkaNoHi (*HolidayDates*), 37
- JPChildrensDay (*HolidayDates*), 37
- JPComingOfAgeDay (*HolidayDates*),
37
- JPConstitutionDay (*HolidayDates*),
37
- JPEmperorsBirthday
(*HolidayDates*), 37
- JPGantan (*HolidayDates*), 37
- JPGreeneryDay (*HolidayDates*), 37
- JPHealthandSportsDay
(*HolidayDates*), 37
- JPKeirouNOhi (*HolidayDates*), 37
- JPKenkokuKinenNoHi
(*HolidayDates*), 37
- JPKenpouKinenBi (*HolidayDates*), 37
- JPKinrouKanshaNoHi
(*HolidayDates*), 37
- JPKodomoNoHi (*HolidayDates*), 37
- JPKokuminNoKyujitu
(*HolidayDates*), 37
- JPMarineDay (*HolidayDates*), 37
- JPMidoriNoHi (*HolidayDates*), 37
- JPNatFoundationDay
(*HolidayDates*), 37
- JPNationalCultureDay
(*HolidayDates*), 37
- JPNationHoliday (*HolidayDates*), 37
- JPNewYearsDay (*HolidayDates*), 37
- JPRespectForTheAgedDay
(*HolidayDates*), 37
- JPSeijinNoHi (*HolidayDates*), 37
- JPShuubunNoHi (*HolidayDates*), 37
- JPTaiikuNoHi (*HolidayDates*), 37
- JPTennouTanjyouBi (*HolidayDates*),
37
- JPThanksgivingDay (*HolidayDates*),
37
- JPUmiNoHi (*HolidayDates*), 37
- julian.timeDate
(*TimeDateCoercion*), 18

- keystatsImport
(*TimeSeriesImport*), 42
- Kiev (*DaylightSavingTime*), 3
- KualaLumpur (*DaylightSavingTime*),
3
- Kuwait (*DaylightSavingTime*), 3

- LaborDay (*HolidayDates*), 37
- lag.timeSeries (*TimeSeriesData*), 27
- Lagos (*DaylightSavingTime*), 3
- length.timeDate (*TimeDateSubsets*), 10
- lines.timeDate (*TimeDateClass*), 4
- lines.timeSeries (*TimeSeriesClass*), 21
- Lisbon (*DaylightSavingTime*), 3
- listFinCenter (*TimeDateClass*), 4
- Ljubljana (*DaylightSavingTime*), 3
- log (*RmetricsUtilities*), 1
- log.timeSeries (*TimeSeriesData*), 27
- London (*DaylightSavingTime*), 3
- LosAngeles (*DaylightSavingTime*), 3
- Luxembourg (*DaylightSavingTime*), 3

- Madrid (*DaylightSavingTime*), 3
- Manila (*DaylightSavingTime*), 3
- MassOfArchangels (*HolidayDates*), 37
- Melbourne (*DaylightSavingTime*), 3
- merge.timeSeries (*TimeSeriesData*), 27
- MexicoCity (*DaylightSavingTime*), 3
- midquoteSeries (*TimeSeriesClass*), 21
- modify (*RmetricsUtilities*), 1
- Monaco (*DaylightSavingTime*), 3
- months.timeDate (*TimeDateCoercion*), 18
- Montreal (*DaylightSavingTime*), 3
- Moscow (*DaylightSavingTime*), 3
- myFinCenter (*TimeDateClass*), 4
- myUnits (*RmetricsUtilities*), 1

- Nairobi (*DaylightSavingTime*), 3
- Nassau (*DaylightSavingTime*), 3
- newPositions<- (*TimeSeriesPositions*), 32
- NewYearsDay (*HolidayDates*), 37
- NewYork (*DaylightSavingTime*), 3
- Nicosia (*DaylightSavingTime*), 3

- ohlcdailyPlot (*TimeSeriesClass*), 21
- Ops.timeDate (*TimeDateMathOps*), 13
- Ops.timeSeries (*TimeSeriesData*), 27
- orderStatistics (*TimeSeriesClass*), 21

- Oslo (*DaylightSavingTime*), 3
- outlier (*RmetricsUtilities*), 1
- outlier.timeSeries (*TimeSeriesData*), 27

- Pacific (*DaylightSavingTime*), 3
- PalmSunday (*HolidayDates*), 37
- Paris (*DaylightSavingTime*), 3
- Pentecost (*HolidayDates*), 37
- PentecostMonday (*HolidayDates*), 37
- Perth (*DaylightSavingTime*), 3
- plot.timeDate (*TimeDateClass*), 4
- plot.timeSeries (*TimeSeriesClass*), 21
- points.timeDate (*TimeDateClass*), 4
- points.timeSeries (*TimeSeriesClass*), 21
- Prague (*DaylightSavingTime*), 3
- PresentationOfLord (*HolidayDates*), 37
- print.keystats (*TimeSeriesImport*), 42
- print.timeDate (*TimeDateClass*), 4
- print.timeSeries (*TimeSeriesClass*), 21

- quantile.timeSeries (*TimeSeriesData*), 27
- Quinquagesima (*HolidayDates*), 37

- rbind.timeSeries (*TimeSeriesData*), 27
- readSeries (*TimeSeriesClass*), 21
- rep.timeDate (*TimeDateMathOps*), 13
- returnSeries (*TimeSeriesClass*), 21
- rev.timeDate (*TimeDateMathOps*), 13
- rev.timeSeries (*TimeSeriesPositions*), 32
- Riga (*DaylightSavingTime*), 3
- Riyadh (*DaylightSavingTime*), 3
- RmetricsUtilities, 1
- RogationSunday (*HolidayDates*), 37
- Rome (*DaylightSavingTime*), 3
- round (*RmetricsUtilities*), 1
- round.timeDate (*TimeDateMathOps*), 13
- rownames<- (*RmetricsUtilities*), 1
- rownames<- .timeSeries (*TimeSeriesData*), 27
- rulesFinCenter (*TimeDateClass*), 4
- sample (*RmetricsUtilities*), 1
- sample.timeDate (*TimeDateMathOps*), 13

- sample.timeSeries
(TimeSeriesPositions), 32
- scale.timeSeries
(TimeSeriesData), 27
- Seoul (DaylightSavingTime), 3
- Septuagesima (HolidayDates), 37
- seq.timeDate (TimeDateClass), 4
- seriesData (TimeSeriesClass), 21
- seriesPositions
(TimeSeriesPositions), 32
- Shanghai (DaylightSavingTime), 3
- show, fWEBDATA-method
(TimeSeriesImport), 42
- show.fWEBDATA (TimeSeriesImport),
42
- sign.timeSeries (TimeSeriesData),
27
- Singapore (DaylightSavingTime), 3
- Sofia (DaylightSavingTime), 3
- SolemnityOfMary (HolidayDates), 37
- sort (RmetricsUtilities), 1
- sort.timeDate (TimeDateMathOps),
13
- sort.timeSeries
(TimeSeriesPositions), 32
- spreadSeries (TimeSeriesClass), 21
- sqrt.timeSeries (TimeSeriesData),
27
- start.timeDate (TimeDateSubsets),
10
- start.timeSeries
(TimeSeriesPositions), 32
- Stockholm (DaylightSavingTime), 3
- summary.timeDate (TimeDateClass),
4
- summary.timeSeries
(TimeSeriesClass), 21
- Sydney (DaylightSavingTime), 3
- Sys.timeDate (TimeDateClass), 4
- tail.timeSeries (TimeSeriesData),
27
- Taipei (DaylightSavingTime), 3
- Tallinn (DaylightSavingTime), 3
- Tehran (DaylightSavingTime), 3
- timeCalendar (TimeDateClass), 4
- timeDate (TimeDateClass), 4
- timeDate-class (TimeDateClass), 4
- TimeDateClass, 4
- TimeDateCoercion, 18
- TimeDateMathOps, 13
- TimeDateSpecDates, 16
- TimeDateSubsets, 10
- timeFirstDayInMonth
(TimeDateSpecDates), 16
- timeFirstDayInQuarter
(TimeDateSpecDates), 16
- timeLastDayInMonth
(TimeDateSpecDates), 16
- timeLastDayInQuarter
(TimeDateSpecDates), 16
- timeLastNdayInMonth
(TimeDateSpecDates), 16
- timeNdayOnOrAfter
(TimeDateSpecDates), 16
- timeNdayOnOrBefore
(TimeDateSpecDates), 16
- timeNthNdayInMonth
(TimeDateSpecDates), 16
- timeSequence (TimeDateClass), 4
- timeSeries (TimeSeriesClass), 21
- timeSeries-class
(TimeSeriesClass), 21
- TimeSeriesClass, 21
- TimeSeriesCoercion, 35
- TimeSeriesData, 27
- TimeSeriesImport, 42
- TimeSeriesPositions, 32
- Tokyo (DaylightSavingTime), 3
- TransfigurationOfLord
(HolidayDates), 37
- TrinitySunday (HolidayDates), 37
- trunc.timeDate (TimeDateMathOps),
13
- Tunis (DaylightSavingTime), 3
- unique.timeDate
(TimeDateMathOps), 13
- USChristmasDay (HolidayDates), 37
- USColumbusDay (HolidayDates), 37
- USCPulaskisBirthday
(HolidayDates), 37
- USDecorationMemorialDay
(HolidayDates), 37
- USElectionDay (HolidayDates), 37
- USGoodFriday (HolidayDates), 37
- USInaugurationDay (HolidayDates),
37
- USIndependenceDay (HolidayDates),
37
- USLaborDay (HolidayDates), 37
- USLincolnsBirthday
(HolidayDates), 37
- USMemorialDay (HolidayDates), 37
- USMLKingsBirthday (HolidayDates),
37

USNewYearsDay (*HolidayDates*), 37
USPresidentsDay (*HolidayDates*), 37
USThanksgivingDay (*HolidayDates*),
37
USVeteransDay (*HolidayDates*), 37
USWashingtonsBirthday
(*HolidayDates*), 37

Vaduz (*DaylightSavingTime*), 3
Vancouver (*DaylightSavingTime*), 3
var (*RmetricsUtilities*), 1
var.timeSeries (*TimeSeriesData*),
27
Vienna (*DaylightSavingTime*), 3
Vilnius (*DaylightSavingTime*), 3

Warsaw (*DaylightSavingTime*), 3
Winnipeg (*DaylightSavingTime*), 3

yahooImport (*TimeSeriesImport*), 42
yahooSeries (*TimeSeriesImport*), 42

Zagreb (*DaylightSavingTime*), 3
Zurich (*DaylightSavingTime*), 3