Package 'fTrading'

September 29, 2009

Version 2100.76

Revision 4406

Date 2009-09-28

Title Technical Trading Analysis

Author Diethelm Wuertz and many others, see the SOURCE file

Depends R (>= 2.4.0), methods, timeDate, timeSeries, fBasics

Suggests RUnit, tcltk

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

Description Environment for teaching "Financial Engineering and Computational Finance"

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

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL http://www.rmetrics.org

Repository CRAN

Date/Publication 2009-09-29 18:44:32

R topics documented:

`BenchmarkAnalysis` *Utilities and Benchmark Analysis*

Description

A collection and description of utility and benchmark functions for the analysis of financial markets. The collection provides a set of functions for the computation of returns, for the display of price charts, and for benchmark measurements.

The functions are:

`ohlcPlot`	Plots open–high–low–close bar charts,
`sharpeRatio`	Computes Sharpe Ratio,
`sterlingRatio`	Computes Sterling Ratio,
`maxDrawDown`	Computes maximum drawdown.

Usage

```
ohlcPlot(x, xlim = NULL, ylim = NULL, xlab = "Time", ylab, col = par("col"),
    bg = par("bg"), axes = TRUE, frame.plot = axes, ann = par("ann"),
    main = NULL, date = c("calendar", "julian"), format = "%Y-%m-%d",
    origin = "1899-12-30", ...)

sharpeRatio(x, r = 0, scale = sqrt(250))
sterlingRatio(x)

maxDrawDown(x)
```

Arguments

`date, format, origin`	[ohlcPlot] - date elements, `date`, a string indicating the type of x axis annotation. Default is calendar dates. `format`, a string indicating the format of the x axis annotation if `date == "calendar"`. For details see `format.POSIXct`. `origin` an R object specifying the origin of the Julian dates if `date == "calendar"`. Defaults to 1899-12-30 (Popular spreadsheet programs internally also use Julian dates with this origin).
`r`	[sharpeRatio] - the risk free rate. Default corresponds to using portfolio returns not in excess of the riskless return.
`scale`	[sharpeRatio] - a scale factor. Default corresponds to an annualization when working with daily financial time series data.

`x` a numeric vector of prices. For `ohlcPlot` a multivariate time series object of class `mts` is required.

`xlim, ylim, xlab, ylab, col, bg, axes, frame.plot, ann, main` [ohlcPlot] - graphical arguments, see `plot`, `plot.default` and `par`.

`...` [ohlcPlot] - further graphical arguments passed to `plot.window`, `title`, `axis`, and `box`.

Details

Open–High–Low–Close Chart:

Within an open–high–low–close bar chart, each bar represents price information for the time interval between the open and the close price. The left tick for each bar indicates the open price for the time interval. The right tick indicates the closing price for the time interval. The vertical length of the bar represents the price range for the time interval. The time scale of `x` must be in Julian dates (days since the `origin`).
`[tseries:plotOHLC]`

Sharpe and Sterling Ratios:

The Sharpe ratio is defined as a portfolio's mean return in excess of the riskless return divided by the portfolio's standard deviation. In finance the Sharpe Ratio represents a measure of the portfolio's risk-adjusted (excess) return. The Sterling ratio is defined as a portfolio's overall return divided by the portfolio's maximum drawdown statistic. In finance the Sterling Ratio represents a measure of the portfolio's risk-adjusted return.
`[tseries:sharpe]`

Maximum Drawdown:

The maximum drawdown or maximum loss statistic is defined as the maximum value drop after one of the peaks of `x`. For financial instruments the maximum drawdown represents the worst investment loss for a buy–and–hold strategy invested in `x`.
`[tseries:maxdrawdown]`

Get Returns:

The function computes the return series given a financial security price series. The price series may be an object of class `numeric` or a time series object. This includes objects of classes `"ts"`, `"its"` and/or `"timeSeries"`.

Value

`ohlcPlot`
creates an Open–High–Low–Close chart.

`sharpeRatio`
`sterlingRatio`
return the Sharpe or Sterling ratio, a numeric value.

`maxDrawDown`
returns a list containing the following three components: `maxDrawDown`, double representing the max drawdown or max loss statistic; `from`, the index (or vector of indices) where the maximum drawdown period starts; `to`, the index (or vector of indices) where the max drawdown period ends.

Author(s)

Adrian Trapletti for the ohlcPlot,*Ratio and maxDrawDown functions,
Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## ohlcPlot -
   # Plot OHLC for SP500
   # ohlcPlot(x, ylab = "price", main = instrument)

## sharpeRatio -
   # Sharpe Ratio for DAX and FTSE:
   data(EuStockMarkets)
   dax = log(EuStockMarkets[, "DAX"])
   ftse = log(EuStockMarkets[, "FTSE"])
   # Ratios:
   sharpeRatio(dax)
   sharpeRatio(ftse)

## maxDrawDown -
   data(EuStockMarkets)
   dax = log(EuStockMarkets[, "DAX"])
   mdd = maxDrawDown(dax)
   mdd
   # Plot DAX:
   plot(dax)
   grid()
   segments(time(dax)[mdd$from], dax[mdd$from],
     time(dax)[mdd$to], dax[mdd$from])
   segments(time(dax)[mdd$from], dax[mdd$to],
     time(dax)[mdd$to], dax[mdd$to])
   mid = time(dax)[(mdd$from + mdd$to)/2]
   arrows(mid, dax[mdd$from], mid, dax[mdd$to], col = 2)
   title(main = "DAX: Max Drawdown")
```

`RollingAnalysis` *Rolling Analysis*

Description

A collection and description of functions to perform a rolling analysis. A rolling analysis is often required in building trading models.

The functions are:

`rollFun`	Rolling or moving sample statistics,
`rollMin`	Rolling or moving sample minimum,
`rollMax`	Rolling or moving sample maximum,
`rollMean`	Rolling or moving sample mean,
`rollVar`	Rolling or moving sample variance.

Usage

```
rollFun(x, n, trim = TRUE, na.rm = FALSE, FUN, ...)
rollMin(x, n = 9, trim = TRUE, na.rm = FALSE)
rollMax(x, n = 9, trim = TRUE, na.rm = FALSE)
rollMean(x, n = 9, trim = TRUE, na.rm = FALSE)
rollVar(x, n = 9, trim = TRUE, unbiased = TRUE, na.rm = FALSE)
```

Arguments

`FUN`	the rolling function, arguments to this function can be passed through the `...` argument.
`n`	an integer specifying the number of periods or terms to use in each rolling/moving sample.
`na.rm`	a logical flag: if TRUE, missing values in x will be removed before computation. The default is FALSE.
`trim`	a logical flag: if TRUE, the first n-1 missing values in the returned object will be removed; if FALSE, they will be saved in the returned object. The default is TRUE.
`unbiased`	a logical flag. If TRUE, the unbiased sample variance will be returned. The default is TRUE.
`x`	an univariate `timeSeries` object or a numeric vector.
`...`	additional arguments to be passed.

Value

The functions return a `timeSeries` object or a numeric vector, depending on the argument `x`.

`rollMax` returns the rolling sample maximum,
`rollMin` returns the rolling sample minimum,
`rollMean` returns the rolling sample mean, and
`rollVar` returns the biased/unbiased rolling sample variance.

Note, that the function `rollFun` always returns a numeric vector, independent of the argument `x`.

If you like to operate for `x` with rectangular objects, you have to call the functions columnwise within a loop.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

See Also

var.

Examples

```
## Rolling Analysis:
   x = (1:10)^2
   x
   trim =  c(TRUE, TRUE, FALSE, FALSE)
   na.rm = c(TRUE, FALSE, TRUE, FALSE)
   for (i in 1:4)
     print(rollMin(x, 5, trim[i], na.rm[i]))
   for (i in 1:4)
     print(rollMax(x, 5, trim[i], na.rm[i]))
   for (i in 1:4)
     print(rollVar(x, 5, trim[i], unbiased = TRUE, na.rm[i]))
   for (i in 1:4)
     print(rollVar(x, 5, trim[i], unbiased = FALSE, na.rm[i]))
```

TechnicalAnalysis *Tools for the Technical Analysis*

Description

A collection and description of functions for the technical analysis of stock markets. The collection provides a set of the most common technical indicators.

Utility Functions:

emaTA	Exponential Moving Average,
biasTA	Bias Indicator,
medpriceTA	Medium Price Indicator,
typicalpriceTA	Typical Price Indicator,
wcloseTA	Weighted Close Indicator,
rocTA	Rate of Change,
oscTA	Oscillator Indicator.

Oscillator Indicators:

momTA	Momentum Indicator,
macdTA	MACD Indicator,
cdsTA	MACD Signal Line,
cdoTA	MACD Oscillator,

`vohlTA`	High/Low Volatility,
`vorTA`	Volatility Ratio.

`stochasticTA`	Stochastics Oscillator,
`fpkTA`	Fast Percent K,
`fpdTA`	Fast Percent D,
`spdTA`	Slow Percent D,
`apdTA`	Averaged Percent D,
`wprTA`	William's Percent R,
`rsiTA`	Relative Strength Index.

S-Plus Like Moving Averages:

`SMA`	Simple Moving Average,
`EWMA`	Exponentially Weighted Moving Average.

Usage

```
emaTA(x, lambda, startup = 0)
biasTA(x, lag)
medpriceTA(high, low)
typicalpriceTA(high, low, close)
wcloseTA(high, low, close)
rocTA(x, lag)
oscTA(x, lag1 = 25, lag2 = 65)

momTA(x, lag)
macdTA(x, lag1, lag2)
cdsTA(x, lag1 = 12, lag2 = 26, lag3 = 9)
cdoTA(x, lag1 = 12, lag2 = 26, lag3 = 9)
vohlTA(high, low)
vorTA(high, low)

stochasticTA(close, high, low, lag1 = 5, lag2 = 3, lag3 = 5,
    type = c("fast", "slow"))
fpkTA(close, high, low, lag)
fpdTA(close, high, low, lag1, lag2)
spdTA(close, high, low, lag1, lag2, lag3)
apdTA(close, high, low, lag1, lag2, lag3, lag4)
wprTA(close, high, low, lag)
rsiTA(close, lag)

SMA(x, n = 5)
EWMA(x, lambda, startup = 0)
```

Arguments

`lag, lag1, lag2, lag3, lag4`
: integer values, time lags.

`n`
: [SMA] -
an integer value, time lag.

`lambda`
: [emaTA][EWMA] -
a numeric value between zero and one giving the decay length of the exponential moving average. If an integer value greater than one is given, lambda is used as a lag of "n" periods to calculate the decay parameter.

`startup`
: [emaTA][EWMA] -
an integer value, the startup position of the exponential moving average, by default 0.

`type`
: [stochasticTA] -
a character string, either `"fast"` or `""slow"` characterizing the type of the percent K and percent D indicator. By default `type="fast"`

`x, high, low, close`
: a numeric vector of prices, either opening, closing, or high and low values. For `ohlcPlot` a multivariate time series object of class `mts`.

Value

`*TA`
The technical Indicators return the following numeric vectors (or matrix):

`emaTA` returns the Exponential Moving Average, EMA
`biasTA` returns the EMA-Bias,
`medpriceTA` returns the Medium Price,
`typicalpriceTA` returns the Typical Price,
`wcloseTA` returns the Weighted Closing Price,
`rocTA` returns the Rate of Change Indicator,
`oscTA` returns the EMA Oscillator Indicator,
`momTA` returns the Momentum Oscillator,

`macdTA` returns the MACD Oscillator,
`cdsTA` returns the MACD Signal Line,
`cdo` returns the MACD Oscillator,
`vohlTA` returns the High/Low Volatility Oscillator,
`vorTA` returns Volatility Ratio Oscillator,

`stochasticTA` returns a 2-column matrix with percent K and D Indicator,
`fpkTA` returns the Fast Percent-K Stochastics Indicator,
`fpdTA` returns the Fast Percent-D Stochastics Indicator,
`spdTA` returns the Slow Percent-D Stochastics Indicator,
`apdTA` returns the Averaged Percent-D Stochastics Indicator,
`wprTA` returns the Williams Percent-R Stochastics Indicator,
`rsiTA` returns the Relative Strength Index Stochastics Indicator.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## data -
   # Load MSFT Data:
   x = MSFT
   colnames(x)
   x = x[, "Close"]
   head(x)

## emaTA -
   # Exponential Moving Average:
   y = emaTA(x, lambda = 9)
   seriesPlot(x)
   lines(y, col = "red")
```

Index