Package ‘RAdamant’

August 2, 2011

Type Package

Title Financial Technical Analysis and Risk Management

Version 0.8.3.1

Date 2011-08-02

Author RAdamant Development Team

Maintainer RAdamant Development Team <team@r-adamant.org>

Depends R (>= 2.11.1), utils, grDevices

Description R-Adamant is a collection of functions and algorithms for processing of Financial Time Series, Risk Management and Econometrics.

License GPL>=2

LazyLoad yes

R topics documented:

`3dptelem` *3D Plot Elements*

Description

Add elements to 3D Plot

Usage

```
lines3d(x, y, z, pmat = getProjectionMatrix(), ...)

points3d(x, y, z, pmat = getProjectionMatrix(), ...)

rect3d(xrange, yrange, z, pmat = getProjectionMatrix(), ...)

text3d(x, y, z, pmat = getProjectionMatrix(), ...)
```

Arguments

`x`	X axis
`y`	Y axis
`z`	Z axis
`pmat`	pamt
`...`	Further arguments to or from other methods
`xrange`	xrange
`yrange`	yrange

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

3dptpars *3D Plot Axis Formatting*

Description

Add and format labels for 3D Plot

Usage

```
x.axis3d(xlim = getPlotLimits(1)
    , ylim = getPlotLimits(2)
    , zlim = getPlotLimits(3)
    , pmat = getProjectionMatrix()
    , at = NULL
    , labels = NULL
    , theme.params = getCurrentTheme()
    , show.labels = TRUE
    , grid = theme.params[["xgrid"]]
    , overrides = list(...)
    , ...
    )

y.axis3d(xlim = getPlotLimits(1)
    , ylim = getPlotLimits(2)
    , zlim = getPlotLimits(3)
    , pmat = getProjectionMatrix()
    , at = NULL
    , labels = NULL
    , theme.params = getCurrentTheme()
    , show.labels = TRUE
    , grid = theme.params[["ygrid"]]
    , overrides = list(...)
    , ...
    )

z.axis3d(xlim = getPlotLimits(1)
    , ylim = getPlotLimits(2)
    , zlim = getPlotLimits(3)
    , pmat = getProjectionMatrix()
    , at = NULL
    , labels =NULL
    , theme.params = getCurrentTheme()
    , show.labels = TRUE
    , grid = theme.params[["zgrid"]]
    , overrides = list(...)
    , ...
    )

x.title3d(xlim = getPlotLimits(1)
    , ylim = getPlotLimits(2)
```

```
     , zlim = getPlotLimits(3)
     , pmat = getProjectionMatrix()
     , title = ""
     , theme.params = getCurrentTheme()
     , ...
     )

y.title3d(xlim = getPlotLimits(1)
     , ylim = getPlotLimits(2)
     , zlim = getPlotLimits(3)
     , pmat = getProjectionMatrix()
     , title = ""
     , theme.params = getCurrentTheme()
     , ...
     )

z.title3d(xlim = getPlotLimits(1)
     , ylim = getPlotLimits(2)
     , zlim = getPlotLimits(3)
     , pmat = getProjectionMatrix()
     , title = ""
     , theme.params = getCurrentTheme()
     , ...
     )

getPlotLimits(which = 1:3
            , env = getOption("RAdamant")
            )

setPlotLimits(xlim = NULL
            , ylim = NULL
            , zlim = NULL
            , env = getOption("RAdamant")
)
```

Arguments

`xlim`	xlim
`ylim`	ylim
`zlim`	zlim
`pmat`	pmat
`at`	at
`which`	which
`env`	environment
`labels`	labels
`title`	title
`theme.params`	theme.params
`show.labels`	show.labels
`grid`	grid

`overrides`	Overrides list
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`abi` *Absolute Breath Index - ABI*

Description

Compute Absolute Breath Index (Technical Analysis)

Usage

```
Abi(X, lag = 5, plot=FALSE, ...)
```

Arguments

`X`	Input numerical series
`lag`	Number of lags
`plot`	Logical. Return plot.
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`absrs` *Absolute Relative Strenght*

Description

Compute Absolute Relative Strenght (Technical Analysis)

Usage

```
absrs(X, lag = 14, na.rm = FALSE, plot = FALSE, ...)
```

Arguments

`x`	x
`lag`	Integer. Number of lag periods.
`na.rm`	na.rm
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`acdi` *Acceleration Deceleration*

Description

Acceleration Deceleration Technical Indicator

Usage

```
acdi(Close
    , High = NULL
    , Low = NULL
    , Vol = NULL
    , plot = TRUE
    , ...
    )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Vol`	Vector. Asset traded Volume.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

adi — *Advance-Decline Indicator*

Description

Advance-Decline Indicator (Technical Analysis)

Usage

```
ADind(close, high, low, lag = 5)
```

Arguments

`close`	Vector. Close price.
`high`	Vector. high price.
`low`	Vector. Low price.
`lag`	Integer. Number of lag periods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

adrating — *Average Directional Rating*

Description

Compute Average Directional Rating index (Technical Analysis)

Usage

```
ADrating(close, high, low, lag)
```

Arguments

`close`	Vector. Close price.
`high`	Vector. high price.
`low`	Vector. Low price.
`lag`	Integer. Number of lag periods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`adratio` *Advance Decline ratio*

Description

Compute Advance Decline ratio (Technical Analysis)

Usage

```
ADratio(X, lag, plot, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`advdec` *Advance Decline issues*

Description

Compute Advance Decline issues (Technical Analysis)

Usage

```
AdvDec(X, lag = 5, ret.idx = TRUE, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`ret.idx`	ret.idx
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`ama` *General Adaptive Moving Average*

Description

General Adaptive Moving Average, computed on each column of the input data X.

Usage

```
ama(X
    , ar.ord = 1
    , ma.ord = 1
    , func = NULL
    , padding = 0
    , type = "AMA"
    , plot = FALSE
    , ...
    )
```

Arguments

`X`	X
`ar.ord`	ar.ord
`ma.ord`	ma.ord
`func`	func
`padding`	padding
`type`	type
`plot`	LOGICAL. If TRUE plot is returned.
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

apo *Apo - Absolute price indicator*

Description

Apo - Absolute price indicator

Usage

```
apo(X, fast.lag = 10, slow.lag = 30, plot = FALSE, ...)
```

Arguments

`X`	X
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

apprais *Appraisal ratio*

Description

`Appraisal`: Calculate Jensen index for a portfolio `Appraisal.Capm`: Get Jensen index from an object of class "Capm".

Usage

```
Appraisal(PTF, ...)
## Default S3 method:
Appraisal(PTF, PTF_M, rf = NULL, rfr = 0, ...)

## S3 method for class 'Capm'
Appraisal(PTF, rfr = 0, ...)
```

Arguments

`PTF`	Input portfolio or an object of class "Capm"
`PTF_M`	Market/benchmark portfolio
`rfr`	Risk free rate
`rf`	Risk free asset
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Sharpe`, `Treynor`, `Jensen`

`armaspc` *Arma spectral representation*

Description

Spectral representation based on ARMA models

Usage

```
Arma.Spec(X, ar_ord = 1, ma_ord = 1, vfreq = NULL)
```

Arguments

`X`	X
`ar_ord`	ar_ord
`ma_ord`	ma_ord
`vfreq`	vfreq

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

arms — *Arms index*

Description

Compute Arms index (Technical Analysis)

Usage

```
Arms(X, Volume, lag, plot = FALSE, ...)
```

Arguments

`X`	X
`Volume`	Vector. Asset traded Volume.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

arodown — *Aroon Down oscillator*

Description

Compute Aroon Down oscillator (Technical Analysis)

Usage

```
arodown(X, lag = 5, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`aroon` *Aroon oscillator*

Description

Compute Aroon oscillator (Technical Analysis)

Usage

```
aroon(X, lag = 5, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`aroud` *Aroon Down oscillator*

Description

Compute Aroon Down oscillator (Technical Analysis)

Usage

```
aroud(X, lag = 5, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	lag
`plot`	plot
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`aroup` *Aroon Up oscillator*

Description

Compute Aroon Up oscillator (Technical Analysis)

Usage

```
aroup(X, lag = 5, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`asfs` *Convert Yahoo! Data into Financial Series object*

Description

Converts a stock data series (dataframe) into a Financial Series (fs) object.

Usage

```
as.fs(X, SName = "", Symbol = "")
```

Arguments

`X`	Input dataframe with columns (Open, High, Low, Close, Volume, Adj.Close).
`SName`	The name assigned to the fs object.
`Symbol`	The symbol assigned to the fs object.

Value

A financial Time Series object. This is a matrix with columns (Open, High, Low, Close, Volume, Adj.Close).
The following attributes are attached to the object:

SName	The Name/Description of the financial series.
Symbol	The input stock symbol.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Load sample financial series data
data(ex_fs)
# Subset data and create another fs object
as.fs(as.data.frame(ex_fs[1:10,])
    , SName = "My Financial Series"
    , Symbol = "My Symbol"
    )
```

`assmeas` *Association measures*

Description

Measures of Association of Predicted Probabilities and Observed Responses

Usage

```
KendallTau(target, pred, ...)

GKgamma(target, pred, ...)

CalcPairs(target, pred, segm_fact = 0.002)

SomerD(target, pred, ...)

confusionM(target, ...)

## Default S3 method:
confusionM(target, pred, th=0.5, ...)

## S3 method for class 'scorecard'
confusionM(target, th=0.5, ...)

accuracy(x, ...)

## S3 method for class 'scorecard'
```

```
accuracy(x, th=0.5, ...)
```

Arguments

`target`	Vector. Observed target value
`pred`	Vector. Predicted values
`x`	An object of class "scorecard"
`segm_fact`	Segmentation factor used for pairs calculation
`th`	Threshold value for the predicted values (Defaults = 0.5)
`...`	Further arguments to or from other methods

Details

- **KendallTau**: calculate Kendall rank correlation coefficient;
- **GKgamma**: calculate Goodman and Kruskal's gamma;
- **Somerd**: calculate Somer D statistic;
- **CalcPairs**: calculate number of *Concordant* and *Discordant* pairs;
- **confusionM**: calculate confusion matrix predicted VS original values
- **accuracy**: get accuracy measure from the results of a classification model

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# load example data set
data(ex_credit)

## Generate Score Card
data = ex_credit[ ,-1]
target = ex_credit[ ,1]
# Example of scorecard
sc3 = Score.card(X=data, Y=target, nseg = c(2,3,4))
sc3

# get confusion matrix for an object of class "scorecard"
confusionM(sc3, 0.5)
# extract accuracy measures
accuracy(sc3, 0.4)

# get predicted values
pred = predict(sc3)

# calculate association measures
SomerD(target, pred)
KendallTau(target, pred)
GKgamma(target, pred)
```

barthann — *Bartlet-Hann window*

Description

Computes Bartlet-Hann window of given length

Usage

```
barthann(N, normalized = TRUE, alpha = 0.38)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.
`alpha`	Shape factor (DEFAULT = 0.38).

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Bartlet-Hann window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Bartlet-Hann window of size 100
x = barthann(100, FALSE)
# Plot the window
cplot(x
    , main = "Bartlet-Hann Window"
    , legend = attr(x, "type")
    )

# Generate another window with different smoothing factor
y = barthann(100, normalized = FALSE, alpha = 0.5)
# Compare the two windows
cplot(cbind(x, y)
    , main = "Bartlet-Hann Window"
    , legend = paste("Bartlet-Hann (alpha = ", c(0.38, 0.5), ")", sep = "")
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

bartlet — *Bartlet window*

Description

Computes Bartlet window of given length

Usage

```
bartlet(N, normalized = TRUE)
```

Arguments

N	Window length.
normalized	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Bartlet window.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate a Normalised Bartlet window of size 100
x = bartlet(100)
# Plot the window
cplot(x
    , main = "Bartlet Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = bartlet(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Bartlet Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

`blackman` *Blackman window*

Description

Computes Blackman window of given length

Usage

```
blackman(N, normalized = TRUE, alpha = 0.16)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.
`alpha`	Shape factor (DEFAULT = 0.16). Determines the smoothing of the window's sidelobes.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Blackman window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Blackman window of size 100
x = blackman(100, FALSE)
# Plot the window
cplot(x
    , main = "Blackman Window"
    , legend = attr(x, "type")
    )

# Generate another window with lower smoothing factor
y = blackman(100, normalized = FALSE, alpha = 0.4)
# Compare the two windows
cplot(cbind(x, y)
    , main = "Blackman Window"
    , legend = paste("Blackman (alpha = ", c(0.16, 0.4), ")", sep = "")
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

bolband *Bollinger Bands*

Description

Compute Bollinger Bands (Technical Analysis)

Usage

```
BolBand(Close, High, Low
    , fact = 2
    , win.size = 5
    , plot = FALSE
    , ...
    )
```

Arguments

Close	Vector. Close price.
High	Vector. High price.
Low	Vector.Low price.
fact	fact
win.size	win.size
plot	Logical. If TRUE plot isreturned.
...	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

bolbandb *Bollinger Bands Bandwidth*

Description

Compute Bollinger Bands Bandwidth (Technical analysis)

Usage

```
BolBandB(Close, High, Low
    , fact=2
    , win.size=5
    , plot=FALSE
    , ...
    )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`fact`	fact
`win.size`	win.size
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`bolfib` *Bollinger Bands - Fibonacci ratio*

Description

Compute Bollinger Bands - Fibonacci ratio (Technical Analysis)

Usage

```
Bol.Fib(Close, High, Low
    , win.size = 5
    , fibo = c(1.618, 2.618, 4.236)
    , plot = FALSE
    , ...
    )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`win.size`	win.size
`fibo`	fibo
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

boot — *General bootstrapping function*

Description

General bootstrapping function

Usage

```
boot(X
    , nboots = 100
    , func = NULL
    , init = NULL
    , message = "Bootstrapping..."
    , ...
    )
```

Arguments

X	X
nboots	nboots
func	func
init	init
message	message
...	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

bop — *Balance of Power*

Description

Compute Balance of Power (Technical Analysis)

Usage

```
Bop(Close, Open, High, Low, smoothed = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`Open`	Vector. Open price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`smoothed`	smoothed
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`box3d` *3D box*

Description

Plotting tools

Usage

```
box3d(x, y, z
     , pmat = getProjectionMatrix()
     , half = FALSE
     , ...
     )
```

Arguments

`x`	X axis
`y`	Y axis
`z`	Z axis
`pmat`	pamt
`half`	half
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

bpdlind — *BPDL indicator*

Description

Compute BPDL indicator (Technical Analysis)

Usage

```
BPDLind(Close, lag = 1, smoothed = TRUE, slag = 5)
```

Arguments

`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`smoothed`	smoothed
`slag`	slag

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

breadth — *Breadth trusth indicator*

Description

Compute Breadth trusth indicator (Technical Analysis)

Usage

```
Breadth(X, lag = 5, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`bromot` *Browniam motion*

Description

Simulate a standard Brownian motion

Usage

```
BroMot(nsim
     , T
     , S0 = 0
     , mi = 0
     , sigma = 1
     , geom = TRUE
     , same.rnd = TRUE
     , plot = FALSE
     , ...
     )
```

Arguments

`nsim`	Integer. Number of simulations
`T`	Time frame of the proces; if missing = nsim
`S0`	Starting point
`mi`	Drift value
`sigma`	Volatility value
`geom`	Logical. Type of BM to simulate, if TRUE simulate Geometric BM else Standard.
`same.rnd`	Logical. Parameter used when multiple series are simulated, id TRUE the same random path is used for all the series.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Value

A matrix of *simulation X n.series* dimension with simulated BM values.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
## Simulate STANDARD Brownian motion
# 100 simulations positvie drift
nsim = 1000
mi = 1.5
BroMot(nsim
, S0=0
```

```
, mi=mi
, sigma=1
, geom=FALSE
, same.rnd=TRUE
, plot=TRUE)

# 1000 simulations, negative drift
nsim = 1000
mi = -2
BroMot(nsim
, S0=1
, mi=mi
, sigma=1
, geom=FALSE
, same.rnd=TRUE
, plot=TRUE)

## Simulate GEOMETRIC Brownian motion
# 500 simulations, 5 series with different variance
nsim = 500
S0 = rep(1, 5)
mi = rep(0, 5)
sigma = seq(1,5)
BroMot(nsim
, S0=S0
, mi=mi
, sigma=sigma
, geom=TRUE
, same.rnd=TRUE
, plot=TRUE)
```

`bromot2d` *2-dimensional Browniam motion*

Description

Simulate n Brownian motion and plot the against each other

Usage

```
BroMot2D(nsim
, T
, S0
, mi
, sigma
, geom = TRUE
, same.rnd = FALSE
, laydisp = NULL
, plot = TRUE
, ...
)
```

Arguments

`nsim`	Integer. Number of simulations
`T`	Time frame of the proces; if missing = nsim
`S0`	Starting point
`mi`	Drift value
`sigma`	Volatility value
`geom`	Logical. Type of BM to simulate, if TRUE simulate Geometric BM else Standard.
`same.rnd`	Logical. Parameter used when multiple series are simulated, id TRUE the same random path is used for all the series.
`laydisp`	Vector. Set the plot window to show the results; specify row and column of the graphic window (par(mfrow=laydisp))
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Value

A matrix of *simulation X n.series* dimension with simulated BM values.

Note

TO BE COMPLETED!

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`BroMot`

Examples

```
S0 = c(1, 2, 3)
mi = c(0, 0, 0)
sigma = c(1, 3, 5)
BroMot2D(nsim=500
, S0=S0
, mi=mi
, sigma=sigma
, geom=TRUE
, same.rnd=FALSE
, laydisp=c(2,2)
)
```

bsgreeks *Black & Scholes greeks*

Description

Calculate analytically Black & Scholes greeks

Usage

```
BS.greeks(X = NULL, ...)
```

Arguments

X	An object of class "BS.price"
...	Further arguments to or from other methods - parameters accepted by the function BS.price

Value

A matrix containing the values for calculated greeks:

Delta

Vega

Theta

Rho

Lambda

Gamma

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

BS.price, BS.moments

Examples

```
# Set BS paramaters
under = 105
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03

# calculate BS price for a call option
# assuming normal distribution of prices
bs1 = BS.price(under
, strike
, rfr
```

```
, sigma
, maty
, yield
, calc.type = "standard"
, opt.type = "call")

# assuming gamma-reciprocal distribution of prices
bs2 = BS.price(under
, strike
, rfr
, sigma
, maty
, yield
, calc.type = "gammarec"
, opt.type = "call")

# calculate greeks for object bs1 of class "BS.price"
BS.greeks(bs1)
class(bs1)

# ... or alternatively passing the same BS paramaters
#  used for price calculation the results are the same
BS.greeks(under=under
, strike=strike
, rfr=rfr
, sigma=sigma
, maty=maty
, yield=yield
, opt.type = "call"
, calc.type = "standard" )

# Same examples as above for different calculation type
BS.greeks(bs2)
class(bs2)
BS.greeks(under=under
, strike=strike
, rfr=rfr
, sigma=sigma
, maty=maty
, yield=yield
, opt.type = "call"
, calc.type = "gammarec")
```

`bsImpvol` *Black & Scholes Implied volatility*

Description

Calculate Black & Scholes Implied volatility

Usage

```
BS.ImpVol(P, under, strike, rfr, sigma, maty,
yield,
```

```
calc.type =c("standard", "lognorm", "gammarec"),
opt.type = c("call", "put"),
interval = c(-20, 20))
```

Arguments

P	Observed Price; single numeric
under	Underlying asset price.
strike	Strike/Exercise price.
rfr	Risk free rate (continuos)
sigma	Assets standard deviation - annualised volatility.
maty	Period of maturity.
yield	Dividend yield (continuos)
calc.type	Calculation type.
opt.type	Type of option (Default="call").
interval	calculation interval applied to the function uniroot (uniroot)

Value

Matrix of Px1 dimensions with Implied volatility values. One row for each value of P.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

BS.greeks, uniroot

Examples

```
# Set BS paramaters
under<- 100
strike <- 95
rfr<- 0.08
sigma <- 0.2
maty<-0.5
yield<- 0.03
calc.type<-"lognorm"
opt.type<-"call"

# calculate implied volatility for single oberved price
P = 11
imp = BS.ImpVol(P, under, strike, rfr, sigma, maty, yield)
imp
# calculate implied volatility for multiple oberved prices
P = seq(9, 11, by=0.1)
imp = BS.ImpVol(P, under, strike, rfr, sigma, maty, yield)
imp
```

`bsmomt` *Black & Scholes moments*

Description

Calculate first four moments for Black & Scholes

Usage

```
BS.moments(BS = NULL, under, rfr, sigma, yield, maty)
```

Arguments

`BS`	An object of class "BS.price"
`under`	Underlying asset price.
`rfr`	Risk free rate (continuos)
`sigma`	Assets standard deviation - annualised volatility.
`yield`	Dividend yield (continuos)
`maty`	Period of maturity.

Value

A matrix containing the four moments (one for each row):

`Mom_1`

`Mom_2`

`Mean`

`Var`

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Set BS paramaters
under = 105
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03
# calculate BS price
bs = BS.price(under, strike, rfr, sigma, maty, yield)
# calculate moments for object bs of class "BS.price"
BS.moments(bs)
# ... or alternatively passing the same BS paramaters used for price
# calculation, the results are the same
BS.moments(NULL, under, rfr, sigma, yield, maty)
```

bsprice	*Black & Scholes price generic*

Description

Generic method for Black & Scholes price

Usage

```
BS.price(under, ...)
## Default S3 method:
BS.price(under
    , strike
    , rfr
    , sigma
    , maty
    , yield
    , calc.type =c("standard","lognorm","gammarec")
    , opt.type = c("call", "put")
    , ...)
## S3 method for class 'BS.price'
print(x, mod, ...)
```

Arguments

`under`	Underlying asset price.
`strike`	Strike/Exercise price.
`rfr`	Risk free rate (continuos)
`sigma`	Assets standard deviation - annualised volatility.
`maty`	Period of maturity.
`yield`	Dividend yield (continuos)
`calc.type`	Calculation type.
`opt.type`	Type of option (Default="call").
`x`	An object of class "BS.price".
`mod`	Control object for print method.
`...`	Further arguments to or from other methods.

Details

The parameter "calc.type" allows to change the Black & Scholes calculation according to different distributional assumptions.

- `standard`: Log asset price normally distributed
- `lognorm`: Log asset price log-normally distributed
- `gammarec`: Log asset price Gamma-Reciprocal distributed

Value

An object of class "BS.price" containing:

`BS Price`

`Factor d1`

`Factor d2`

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`BS.greeks`, `BS.moments`

Examples

```
# Set BS paramaters
under = 100
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03

# calculate BS price for a call option assuming normal distribution of prices
bs1 = BS.price(under
                    , strike
                    , rfr
                    , sigma
                    ,maty
                    , yield
                    , calc.type = "standard"
                    , opt.type = "call")
bs1
# same example assuming gamma-reciprocal distribution of prices
bs2 = BS.price(under
                    , strike
                    , rfr
                    , sigma
                    , maty
                    , yield
                    , calc.type = "gammarec"
                    , opt.type = "call")
bs2
```

`buypre` *Buying pressure indicator*

Description

Compute Buying pressure indicator (Technical Analysis)

Usage

```
buypre(Close, Low, lag = 5, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`capm` *Capm - default method*

Description

Default method for CAPM

Usage

```
Capm(PTF, ...)

## Default S3 method:
Capm(PTF, PTF_M, rf = NULL, rfr = NULL, ...)
```

Arguments

`PTF`	Matrix of returns, one series for each asset in the portfolio.
`PTF_M`	Vector of returns for the market portfolio
`rf`	Vector. Risk free asset returns
`rfr`	Numeric. Risk free rate
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
    # load example dataset
    data(ex_ptf)
    # Generate a random return risk free asset
    rf = rnorm(NROW(ex_ptf), mean = 0.05, sd = 0.01)
    # Calculate CAPM
    Capm(PTF = ex_ptf[,-1], PTF_M = ex_ptf[,1], rf)

## Not run:
    ## Example with real time series
    fromDt = as.Date("2010-01-01")
    ACME = get.fs("APKT",SName = "Acme Packet", from = fromDt)
    ABTL = get.fs("ABTL", SName = "Autobytel", from = fromDt)
    CNAF = get.fs("CNAF", from = fromDt)
    BIIB = get.fs("BIIB", SName = "Biogen", from = fromDt)
    SONY = get.fs("SNE", SName = "Sony", from = fromDt)
    ENI = get.fs("E", SName = "Eni", from = fromDt)
    ptf = combine.fs(ACME, ABTL, CNAF, BIIB, SONY, ENI);
    head(ptf)

    # Load a Benchmark Portfolio Index
    NASDAQ = get.fs("^IXIC", SName = "NASDAQ", from = fromDt);

    R_ptf = Ret(ptf, na.rm = TRUE);
    # Return of the Benchmark portfolio (NASDAQ index)
    R_NASDAQ = Ret(NASDAQ, na.rm = TRUE)

    # Generate a random return risk free asset
    rf = rnorm(NROW(R_ptf), mean = 0.05, sd = 0.01)
    Capm(R_ptf, R_NASDAQ, rf)

## End(Not run)
```

---

cbarplot *Customised Bar Plot*

Description

Workhorse function for automatic bar plotting

Usage

```
cbarplot(X
    , main = NULL
    , xtitle = ""
    , ytitle = ""
    , xlabels = NULL
    , ylabels = NULL
    , yrange = NULL
    , show.xlabels = TRUE
    , show.ylabels = TRUE
    , show.xticks = FALSE
```

```
, show.yticks = FALSE
, grid = TRUE
, grid.method = "sampling"
, show.legend = TRUE
, legend = NULL
, legend.col = theme.params[["col"]]
, beside = FALSE
, density = NULL
, border = "transparent"
, multicolor = FALSE
, theme.params = getCurrentTheme()
, overrides = list(...)
, ...
)
```

Arguments

`X`	Matrix of data to plot. One bar per row, bars are grouped by the columsn of X.
`main`	Main title for the plot
`xtitle`	Title for the x-axis
`ytitle`	Title for the left y-axis
`xlabels`	Labels for x-axis tick marks
`ylabels`	Labels for left y-axis tick marks
`yrange`	y-axis range
`show.xlabels`	Logical. If TRUE, x-axis labels are plotted
`show.ylabels`	Logical. If TRUE, y-axis labels are plotted
`show.xticks`	Logical. If TRUE, x-axis ticks are plotted
`show.yticks`	Logical. If TRUE, y-axis ticks are plotted
`grid`	Logical. If TRUE, a grid is plotted.
`grid.method`	One of "sampling", "equispaced". See draw.grid for details.
`show.legend`	Logical. If TRUE, legend is added to the plot.
`legend`	Vector of text for the legend
`legend.col`	Colors for the elements in the legend.
`beside`	Logical. If FALSE, the columns of X are stacked, if TRUE the columns are portrayed as juxtaposed bars. Used when NCOL(X) > 1.
`density`	A vector giving the density of shading lines for the color filling of the bars. See barplot for details.
`border`	The color to be used for the border of the bars. See barplot for details.
`multicolor`	Logical. If TRUE, a separate color is used for each data point, as provided by the 'col' parameter of the theme.
`theme.params`	RAdamant graphics theme.
`overrides`	List of attributes for the theme override.
`...`	Alternative way to quickly override the theme.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`barplot`, `setThemeAttr`, `draw.grid`, `draw.legend`, `draw.x.axis`, `draw.x.title`, `draw.y.title`, `draw.y.axis`.

Examples

```
cbarplot(rnorm(10), main = "Random Bars")
```

`cci` *Commodity channel index*

Description

Compute Commodity channel index (Technical Analysis)

Usage

```
cci(High, Low, Close, lag = 5, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`cciv2` *Commodity channel index v02*

Description

Compute Commodity channel index v02 (Technical Analysis)

Usage

```
cci.v2(High, Low, Close, lag = 5, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`chaikin` *Chaikin oscillator*

Description

Compute Chaikin oscillator (Technical Analysis)

Usage

```
chaikin(Close
    , High = NULL
    , Low = NULL
    , Vol = NULL
    , fast.lag = 3
    , slow.lag = 10
    , plot = TRUE
    , ...
    )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Vol`	Vector. Asset traded Volume.
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`chaosacc` *Chaos Accelerator oscillator*

Description

Compute Chaos Accelerator oscillator (Technical Analysis)

Usage

```
chaosAcc(X)
```

Arguments

`X`	X

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

chist *Custom Histogram Plot*

Description

Custom histogram plot

Usage

```
chist(x
    , nclass = min(max(round(NROW(x)/10), 10), NROW(x))
    , density = c("kernel", "normal")
    , kernel = c("gaussian", "epanechnikov", "rectangular"
                , "triangular", "biweight", "cosine", "optcosine")
    , theme.params = getCurrentTheme()
    , main = "Histogram and Kernel Density Estimation"
    , xtitle = NULL
    , ytitle = NULL
    , legend = NULL
    , show.legend = TRUE
    , normalised = FALSE
    , ...
    )
```

Arguments

`x` The input data on which the histogram is computed.

`nclass` one of:

- a vector giving the breakpoints between histogram cells.
- a single number giving the number of cells for the histogram.
- a character string naming an algorithm to compute the number of cells.
- a function to compute the number of cells.

In the last three cases the number is a suggestion only.

`density` The model used to compute the probability density estimation:

- "kernel": Kernel density estimation is computed. The kernel function used is controlled by the 'kernel' parameter.
- "normal": A Normal distribution is fitted to the data.

`kernel` the basis function used for kernel density estimation. Used only when density = "kernel".

`theme.params` RAdamant graphics theme.

`main` The plot title

`xtitle` Title for x-axis.

`ytitle` Title for y-axis

`legend` The legend text.

`show.legend` Logical. If TRUE, the legend is added to the plot.

`normalised` Logical. If TRUE, the histogram and the density function are scaled so that the maximum point is 1.

`...` Additional parameters passed to cplot.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

hist, cplot.

Examples

```
# Generate random data from the Normal distribution
x = rnorm(1000);

# Compute histogram plot, and fit Normal density
chist(x, nclass = 20, density = "normal");

# Compute histogram plot, and fit Epanechnikov Kernel density
chist(x, nclass = 20, density = "kernel", kernel = "epanechnikov");
```

`chvol` *Chaikin volatility indicator*

Description

Compute Chaikin volatility indicator (Technical Analysis)

Usage

```
Ch.vol(High, Low, Close, lag = 5, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

cleanup — *Clean memory*

Description

Cleanup environment and (optionally) performs Garbage Collection

Usage

```
cleanup(keep = c(), env = parent.frame(), gc = FALSE)
```

Arguments

keep	CHARACTER. Vector of variables to keep in memory.
env	Environment from which objects are removed. Defaults to the environment from which this function is called.
gc	Logical. If TRUE, garbage collection is performed to release memory. (Default = TRUE)

Value

Void

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

clust — *Time series clusters*

Description

Create a simple cluster partition of a time series

Usage

```
TSClust(x, ...)

## Default S3 method:
TSClust(x, y=NULL, n_clust=5,
bk.type=c("quantile","volatility","uniform","custom"),
pc_vol=0.1, win.size=10,  custom_breaks=NULL,
lab.dig=0, ...)

## S3 method for class 'TSClust'
summary(object, funs = summary, ...)
```

```
## S3 method for class 'TSClust'
plot(x, smooth=FALSE, ...)
```

Arguments

`x, object`	Univariate time series or an object of class "TSClust"
`y`	y
`n_clust`	number of cluster
`bk.type`	Breaks type
`custom_breaks`	custom_breaks
`lab.dig`	lab.dig
`funs`	function to run inside summary.TSClust
`smooth`	smooth
`pc_vol`	pc_vol
`win.size`	win.size
`...`	further arguments accepted by "funs"

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`clv` *Close Location value oscillator*

Description

Compute Close Location value oscillator (Technical Analysis)

Usage

```
clv(Close, High = NULL, Low = NULL, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

cmf — *Chaikin Money Flow*

Description

Compute Chaikin Money Flow (Technical Analysis)

Usage

```
cmf(Close, Low, High, Volume, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`Low`	Vector. Low price.
`High`	Vector. High price.
`Volume`	Volume
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

cmof — *Chande Momentum Oscillator*

Description

Compute Chande Momentum Oscillator (Technical Analysis)

Usage

```
cmof(X, lag = 5, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

coefmreg — *Extract Model Coefficients for (Multi)-Regression object*

Description

Generic method for extracting model coefficoents from object of classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'reg'
coef(object, ...)

## S3 method for class 'mreg'
coef(object, ...)
```

Arguments

`object`	Instance of class 'reg'/'mreg'.
`...`	Further arguments to or from other methods.

Value

One of the following:

- class 'mreg': A matrix containing all model coefficients, one column for each model.
- class 'reg': A matrix containing the model specific coefficients.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`mreg`.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y1 = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);
```

```
Y2 = -2 + 1.2*X1 -X2 + rnorm(N, sd = sigma);

# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2), X = cbind(X1, X2), plot = FALSE);
# Extract all coefficients
coef(mod)
# Extract coefficients from the first model
coef(mod[[1]])
```

`cofit` *Cornish Fisher Transformation*

Description

Estimate quantiles based on Cornish Fisher formula, which only uses skewness and kurtosis.

Usage

```
cofit(X, p, k = NULL, s = NULL)
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	Vector of probability threshold (interval [0, 1])
`k`	Kurtosis (Default: NULL -> becomes kurt(X))
`s`	Skewness (Default: NULL -> becomes skew(X))

Value

A matrix length(p) by NCOL(X) of estimated quantiles.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Estimate 5% quantile from Normal random data.
cofit(rnorm(1000), p = 0.05)
# Compare to theoretical quantile
qnorm(0.05)

# Estimate 5% quantile from Student's T random data.
cofit(rt(1000, 16), p = 0.05)
# Compare to theoretical quantile
qt(0.05, df = 16)
```

`colinprs` *Co-Linearity analysis*

Description

This function performs a Co-Linearity analysis between the columns of X.
Correlation factors between columns are computed, and pairs of columns with a correlation factor higher than a specified threshold are returned.

Usage

```
colin.pairs(X, trsh = 0.8)
```

Arguments

`X`	Matrix of data series (one column per variable).
`trsh`	Threshold over which two columns are considered too correlated (Default: 0.8).

Value

A list of with the following elements:

`CoLinMat`	Lower Triangular correlation matrix (Correlations between the columns of X).
`CoLinPairs`	Data frame of columns [VAR1, VAR2, Rho] containing the pairs of columns with a correlation factor higher than the given threshold, sorted in descending order.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load sample time series data
data(ex_ptf);

# Compute correlation matrix and column pairs
# with correlation higher than 0.8
colin.pairs(ex_ptf);
```

`colinred` *Multi Co-Linearity reduction*

Description

Performs multicollinearity reduction. Cross Co-Linearity analysis between the columns of Y and X is computed, then for each column Yi, a reduced set of the columns of X is computed by removing those columns that are too correlated (one for each co-linear pair).
In the removal process, those columns of X that are most correlated to Yi are kept.

Usage

```
colin.reduce(Y, X, max.iter = 100, trsh = 0.85)
```

Arguments

Y	Matrix of data series - Dependent variables (one column per variable).
X	Matrix of data series - Independent variables (one column per variable).
max.iter	Max number of iterations allowed.
trsh	Threshold over which two columns are considered too correlated (Default: 0.8).

Value

A list of Ny elements (Ny = number of columns of Y):

i-th element	Matrix containing a subset of the columns of X. This is obtained by removing collinear entries. This element of the list is named after the corresponding i-th column of Y (or a default is given if Yi has no name).

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

colin.pairs, cross.colin.

Examples

```
# Load sample time series data
data(ex_ptf);

# Select dependent variable
Y = ex_ptf[, 1, drop = FALSE];
# Select indipendent variables
X = ex_ptf[, -1, drop = FALSE];
# Print column names
colnames(X)

# Collinearity Reduction
X.red = colin.reduce(Y, X, trsh = 0.8);
# Print remaining column names
colnames(X.red[[1]])
```

combine *Combine Multiple objects*

Description

This is a generic function, the default implementation combines Financial Series objects.

Usage

```
combine(...)
## Default S3 method:
combine(...)
## S3 method for class 'fs'
combine(..., which = "Close")
```

Arguments

`...`	All input objects to be combined.
`which`	Which column/columns to extract from each input object

Value

Result depends on the implementation.
The default method is a call to combine.fs which returns a matrix containing the selected columns from each input object.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load a set of assets
StartDate = as.Date("2010-01-01");
ACME = get.fs("APKT", SName = "Acme Packet", from = StartDate);
ABTL = get.fs("ABTL", SName = "Autobytel", from = StartDate);
CNAF = get.fs("CNAF", from = StartDate);
BIIB = get.fs("BIIB", SName = "Biogen", from = StartDate);
SONY = get.fs("SNE", SName = "Sony", from = StartDate);
ENI = get.fs("E", SName = "Eni", from = StartDate);

# Combine all series together in matrix format
Portfolio = combine(ACME, ABTL, CNAF, BIIB, SONY, ENI);
Portfolio[1:10, ]
# Combine Close and Volume data from each series
Portfolio2 = combine(ACME
                            , ABTL
                            , CNAF
                            , BIIB
                            , SONY
                            , ENI
                            , which = c("Close", "Volume")
                            );
```

```
Portfolio2[1:10, ]
```

`cosine` *Cosine window*

Description

Computes Cosine window of given length

Usage

```
cosine(N, normalized = TRUE)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Cosine window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Normalised Cosine window of size 100
x = cosine(100)
# Plot the window
cplot(x
    , main = "Cosine Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = cosine(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Cosine Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

covecar — *Extract Model Coefficients from Vector AutoRegressive object*

Description

Generic method for extracting model coefficients matrix from object of class 'VecAr'.

Usage

```
## S3 method for class 'VecAr'
coef(object, ...)
```

Arguments

`object`	Instance of class 'VecAr'.
`...`	Further arguments to or from other methods.

Value

A matrix containing all model coefficients, one column for each variable in the model.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `coef.mreg`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Extract coefficients
coef(mod)
```

covesvar — *Compute residual and coefficients covariance matrix from Vector AutoRegressive object*

Description

Generic method for computing residual and coefficients covariance matrix from object of class 'VecAr'.

Usage

```
## S3 method for class 'VecAr'
estVar(object, ...)
## S3 method for class 'VecAr'
vcov(object, ...)
```

Arguments

`object`	Instance of class 'VecAr'.
`...`	Further arguments to or from other methods.

Value

A matrix with calculated residual / coefficients covariance

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

VecAr, residuals.VecAr, coef.VecAr

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Extract residual covariance matrix
estVar.VecAr(mod)
# Extract coefficients covariance matrix
vcov.VecAr(mod)
```

---

cplot — *2-Dimensional Plotting*

Description

Workhorse function for automatic plotting

Usage

```
cplot(X
    , base = NULL
    , xrange = NULL
    , yrange = NULL
    , theme.params = getCurrentTheme()
    , xtitle = ""
```

```
, xlabels = NULL
, ytitle = ""
, ylabels = NULL
, ytitle2 = ""
, ylabels2 = NULL
, show.xlabels = TRUE
, show.ylabels = TRUE
, main = NULL
, legend = NULL
, legend.col = theme.params[["col"]]
, show.legend = TRUE
, shaded = FALSE
, grid = TRUE
, overrides = list(...)
, new.device = FALSE
, append = FALSE
, multicolor = FALSE
, ...
)
```

Arguments

`X`	Matrix of data to plot. One line per column
`base`	x-coordinates of the plot. All columns of X will share the same base
`xrange`	x axis range
`yrange`	y axis range
`theme.params`	RAdamant graphics theme
`xtitle`	Title for the x-axis
`xlabels`	Labels for x-axis tick marks
`ytitle`	Title for the left y-axis
`ylabels`	Labels for left y-axis tick marks
`ytitle2`	Title for the right y-axis
`ylabels2`	Labels for right y-axis tick marks
`show.xlabels`	Logical. If TRUE, x-axis labels are plotted
`show.ylabels`	Logical. If TRUE, y-axis labels are plotted
`main`	Main title for the plot
`legend`	Vector of text for the legend
`legend.col`	Colors for the elements in the legend
`show.legend`	Logical. If TRUE, legend is added to the plot
`shaded`	Logical vector. If TRUE, a shaded area is added to the corresponding column.
`grid`	Logical. If TRUE, a grid is plotted.
`overrides`	overrides list
`new.device`	Logical. If TRUE, a new window device is opened.
`append`	Logical. If TRUE, append to existing plot
`multicolor`	Logical. If TRUE, a separate color is used for each data point, as provided by the 'col' parameter of the theme
`...`	Additional parameters passed to the function create.empty.plot. Also used to quickly override the theme.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

plot, draw.grid, draw.legend, draw.projections, draw.x.axis, draw.x.title, draw.y.title, draw.y.axis

Examples

```
# Generate four random time series
X = matrix(cumsum(rnorm(1000)), ncol = 4)
colnames(X) = c("A", "B", "C", "D");

# Simple plot
cplot(X)

# Change Title and xlabels
Xlab = paste("t[", 0:249, "]", sep = "");
cplot(X
    , main = "Four Random Time Series"
    , xlabels = parse(text = Xlab)
    )

# Add shaded area to the first time series
cplot(X
    , main = "Four Random Time Series"
    , xlabels = parse(text = Xlab)
    , shaded = TRUE
    )

# Add 45 degree shaded area to the second time series
cplot(X
    , main = "Four Random Time Series"
    , xlabels = parse(text = Xlab)
    , shaded = c(FALSE, TRUE)
    # Theme overrides
    , shade.angle = 45
    )

# Plot
cplot(X[, 1]
    , main = "Gradient Shaded Area Plot"
    , xlabels = parse(text = Xlab)
    , shaded = TRUE
    # Use different Theme
    , theme.params = getTheme("Vanilla")
    #### Theme overrides ####
    # filling density of the shaded area
    , shade.density = 100
    # Alpha transparency will be interpolated from 0 to 1 (Not Run, VERY SLOW)
```

```
#, shade.alpha = c(0, 1)
# Multiple colors for the shaded area
, shade.col = jet.colors(30)
# Multiple stripes are used to generate color gradient
, shade.stripes = 50
# Remove rotation for x-axis
, xlab.srt = 0
)
```

---

cplot3d *3-Dimensional plotting*

Description

Workhorse function for 3D automatic plotting

Usage

```
cplot3d(x, y, z
, fill = c("simple", "colormap", "gradient")
, main = ""
, xtitle = ""
, ytitle = ""
, ztitle = ""
, xlim = range(x) + 0.1*diff(range(x))*c(-1, 1)
, ylim = range(y) + 0.1*diff(range(y))*c(-1, 1)
, zlim = range(z, na.rm = TRUE) + 0.1*diff(range(z, na.rm = TRUE))*c(-1, 1)
, pre = NULL
, post = NULL
, theme.params = getCurrentTheme()
, overrides = list(...)
, new.device = FALSE
, append = FALSE
, axis = TRUE
, xlabels = NULL
, ylabels = NULL
, zlabels = NULL
, show.xlabels = TRUE
, show.ylabels = TRUE
, show.zlabels = TRUE
, show.xticks = TRUE
, show.yticks = TRUE
, show.zticks = TRUE
, ...)
```

Arguments

x	x coordinates for the plot
y	y coordinates for the plot
z	z coordinates for the plot
fill	fill

`main`	main
`xtitle`	xtitle
`ytitle`	ytitle
`ztitle`	ztitle
`xlim`	xlim
`ylim`	ylim
`zlim`	zlim
`xlabels`	xlables
`ylabels`	ylabels
`zlabels`	zlabels
`pre`	pre
`post`	post
`theme.params`	theme.params
`overrides`	overrides
`new.device`	new.device
`append`	append
`axis`	axis
`show.xlabels`	show.xlabels
`show.ylabels`	show.ylabels
`show.zlabels`	show.zlabels
`show.xticks`	show.xticks
`show.yticks`	show.yticks
`show.zticks`	show.zticks
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`cramv` *Cramers V*

Description

Calculate Cramers V

Usage

```
cramv(x, y)
```

Arguments

`x`	x
`y`	y

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

crbtree *CRR Binomial Tree*

Description

Option evaluation with Cox, Rossand and Rubinstein Binomial Tree

Usage

```
CRR.BinTree(Nsteps
, under
, strike
, rfr
, sigma
, maty
, yield
, life
, ret.steps = FALSE)
```

Arguments

`Nsteps`	Nsteps
`under`	Underlying asset price.
`strike`	Strike/Exercise price.
`rfr`	Risk free rate (continuos).
`sigma`	Assets standard deviation - annualised volatility.
`maty`	Period of maturity.
`yield`	Dividend yield (continuos).
`life`	Option life.
`ret.steps`	Logical. If TRUE the calculated steps (step matrix) are returned.

Value

List of results containing the following elements:

`Price_eval`	: Estimated option value at each step.
`Moments`	: Moments of the distribution of the share returns (both Black & Scholes and CRR values are displayed).
`Values`	: Option estimated values (both Black & Scholes and CRR values are displayed).
`Price_Path`	: Step matrix containing the expected share price at each step.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

BS.price, StepMat, JR.BinTree

Examples

```
# set option parameters
under = 105
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03
life = 0.5

# estimate option price using Jarrow and Rudd Binomial Tree
crr = CRR.BinTree(Nsteps=10
                        , under
                        , strike
                        , rfr
                        , sigma
                        , maty
                        , yield
                        , life
                        , ret.steps=TRUE)
crr$Values
# ... confront results with B&S method
BS.price(under, strike, rfr, sigma, maty, yield)
# get step matrix
crr = CRR.BinTree(Nsteps=10
                        , under
                        , strike
                        , rfr
                        , sigma
                        , maty
                        , yield
                        , life
                        , ret.steps=TRUE)
crr$Price_Path
```

---

croscf *Cross Correlation Function*

Description

Compute the cross correlation function for each pairs of variables (Yi Xj)

Usage

```
cross.ccf(Y, X, lag.max = 10, ci = 0.95, plot = TRUE, ...)
```

Arguments

`Y`	Matrix of data series (one column per variable)
`X`	Matrix of data series (one column per variable)
`lag.max`	Max lag to be computed by the cross correlation function (Default: 10)
`ci`	Confidence Interval (Default: 0.95)
`plot`	Logical. If TRUE, results are plotted.
`...`	Additional parameters accepted by the function plot.cross.ccf.

Value

An object of class "cross.acf". This is a list of Ny*Nx elements, where each entry is the cross correlation of the pair (Yi, Xj).

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate two random integrated series
N = 100
X = matrix(rnorm(N), nrow = N/2, ncol=2);
# Create two series as a linear combination of X plus noise
Y = X
# Perform Cross Correlation Analysis
cross.ccf(Y, X)
```

`crosplot` *Y Vs X Cross Plot*

Description

Plot the input dependent variable Y versus each input independent variable X

Usage

```
cross.plot(Y
    , X
    , theme.params = getCurrentTheme()
    , xlabels = NULL
    , two.axis = TRUE
    , shaded.first = FALSE
    , overrides = list(...)
    , ...
    )
```

Arguments

`Y`	Dependent variable.
`X`	Matrix containing all independent variables (one column per variable).
`theme.params`	Theme parameters (Default: getCurrentTheme()).
`xlabels`	Vector of labels associated to the rows of X (i.e. Time labels)(Default: NULL)
`two.axis`	Logical. If TRUE, series are plotted on two axis (two scales).
`shaded.first`	Logical. If TRUE, the variable Y is shaded.
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...))
`...`	Alternative way to quickly override the theme.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load sample time series data
data(ex_ptf)
# Define the dependent variable
Y = ex_ptf[, 1, drop = FALSE];
# Define the independent variables
X = ex_ptf[, -1];
# Define x-axis labels
time.labels = paste("t[", 1:length(Y), "]", sep = "")
# Cross plot
cross.plot(Y, X
        , xlabels = parse(text = time.labels)
        , overrides = list(xlab.srt = 0)
        )
```

---

crscolin *Cross Co-Linearity Analysis*

Description

Perform a cross Co-Linearity analysis between the columns of Y and X:
Correlation factors between each column Yi and all columns of X are calculated for different time lags.
Pairs of columns of X with a correlation factor higher than a specified threshold are also returned.

Usage

```
cross.colin(Y, X, max.lag = 8, trsh = 0.8)
```

Arguments

`Y`	Matrix of data series - Dependent variables (one column per variable)
`X`	Matrix of data series - Independent variables (one column per variable)
`max.lag`	Max lag for which cross correlation is computed
`trsh`	Threshold over which two columns are considered too correlated (Default: 0.8)

Value

A list of Ny + 2 elements (Ny = number of columns of Y):

`First Ny elements`	Lagged correlation matrix (Nx by max.lag+1) between Yi and X. Named as the column names of Y (or default is given if null).
`CoLinMat`	Lower Triangular correlation matrix (Correlations between the columns of X)
`CoLinPairs`	Data frame of columns [VAR1, VAR2, Rho] containing the pairs of columns with a correlation factor higher than the given threshold, sorted by Rho in descending order.

Author(s)

RAdamant Development Team `team@r-adamant.org`

See Also

`colin.pairs`

Examples

```
# Load sample time series data
data(ex_ptf);

# Select dependent variable
Y = ex_ptf[, 1, drop = FALSE];
# Select indipendent variables
X = ex_ptf[, -1, drop = FALSE];

# Cross Co-Linearity analysis
cross.colin(Y, X, max.lag = 4, trsh = 0.8);
```

---

cumfun *Cumulative functions*

Description

Cumulative max / min / Mean / Standard Deviation / Variance / sum on each column of the input matrix.

Usage

```
cumMax(X, lag = 0, padding = NA, na.rm = FALSE)
```

Arguments

X	Input matrix/sequence
lag	Vector of integer lags. If lag >= 0 data are shifted to the right, else to the left. (Default: 0)
padding	Value used to initialise the output matrix (Default: NA)
na.rm	Logical. If TRUE, N-lag entries are removed from the output. Also NA in the input are replaced by -Inf (Default: FALSE)

Details

Sequences are treated as one-column matrices

Value

A matrix of cumulative maximums of X. Number of rows depends on the na.rm parameter. Number of columns is NCOL(X)

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

lew

---

dataset *Example datasets for portfolio and time series analysis*

Description

`ex_ts`: Univariate timeseris of 126 observations; `ex_ptf`: Matrix of returns: 60 rows and 8 colums. The first column is taken as a "market fund" and the other 7 columns are 8 possible indexes. `ex_fs`: An object of class "fs" containing financial series: 252 rows and 6 colums.

Usage

```
data(ex_ts)
data(ex_ptf)
data(ex_fs)
data(ex_credit)
```

Source

Artificially created.

`decimals` *Count Decimals*

Description

Count the number of digits of the decimal part (mantissa) of a number

Usage

```
decimals(x, max.digits = 10, ...)
```

Arguments

`x`	The number for which the count of decimals is required.
`max.digits`	Controls the resolution. See details.
`...`	Not used, for future releases.

Details

The number x is first converted into a string, where the decimal part is truncated after max.digits. The number of significant digits of the decimal part are hence calculated. The truncation allows to remove the artifacts introduced by the finite resolution of the numbers representation.

Value

The number of digits of the mantissa

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Print the mantissa of the number 0.01 with 80 digits.
sprintf("%.80f", 0.01)

# Real number of digits is 2
decimals(0.01, max.digits = 10)

# Number of digits of the mantissa of the computer representation of 0.01
decimals(0.01, max.digits = 100)
```

`decscal` *Decimal scale*

Description

Compute decimal scale of a vector

Usage

```
Decscal(x, scale = 0.1)
```

Arguments

`x`	x
`scale`	scale

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`dema` *Double EMA*

Description

Compute multiple Double EMA on the input data, one for each column of X[, i] and window size win.size[j]

Usage

```
dema(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	X
`win.size`	win.size
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
DEMA is a weighted combination of EMA: 2*EMA(X) - EMA(EMA(X)).
Smoothing factor: lambda = 2/(win.size+1).

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
dema(x, 10)

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
dema(x, 30, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
dema(ex_fs, 30, plot=TRUE)

## End(Not run)
```

---

demark *DeMark indicator*

Description

Compute DeMark indicator (Technical Analysis)

Usage

```
demark(High, Low, Close, lag = 5, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

dgev — *Generalised Extreme Value (GEV)*

Description

Generalised Extreme Value (GEV) - Density function

Usage

```
dgev(X, mu = 0, xi = 0.1, sigma = 1)
```

Arguments

`X`	X
`mu`	mu
`xi`	xi
`sigma`	sigma

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

dgpd — *Generalised Pareto Distribution (GPD)*

Description

Generalised Pareto Distribution (GPD) - Density function

Usage

```
dgpd(X, xi = 0.1, sigma = 1, trsh = 0)
```

Arguments

`X`	X
`xi`	xi
`sigma`	sigma
`trsh`	trsh

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`dma` *Derivative Moving Averages*

Description

Compute multiple Derivative Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
dma(X, fast.win = 5, slow.win = 28, plot = FALSE, ...)
```

Arguments

`X`	X
`fast.win`	fast.win
`slow.win`	slow.win
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
Formula: 100 * (movMax(SMA(X, fast.win), slow.win) - movMin(SMA(X, fast.win), slow.win)) / X.

Value

A object of class 'ma' with attributes type = "DMA" and 'win.size' as from the corresponding input parameters [fast.win,slow.win]:
- matrix of size NROW(X) by NCOL(X) where each column is the moving average of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`sma`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average
dma(x, fast.win=10, slow.win=35)

## Not run:
# refine results of moving average
setCurrentTheme(2)
dma(x, fast.win=10, slow.win=35, plot = TRUE)

## End(Not run)
```

`dpo` *Detrended price oscillator*

Description

Compute Detrended price oscillator (Technical Analysis)

Usage

```
dpo(Close, lag = 5, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

drawdown *Drawdown*

Description

Drawdown risk analysis

Usage

```
drawdown(x, ...)
## Default S3 method:
drawdown(x
                                    , FUN=max
                                    , relative=FALSE
                                    , plots=c("regular", "smooth", "no.plot")
                                    , ...)
## S3 method for class 'drawdown'
summary(object, show.extr=TRUE, ...)
ExtremeDD(DD, FUN, lag = 1, rolling = FALSE, plot = TRUE, ...)
```

Arguments

`x`	Univariate input time series.
`FUN`	Extreme function applied for the max / min drawdown calculation (Default: max)
`relative`	Logical. If TRUE relative drawdown will be calculated.
`plots`	Character. Type of plot to be returned (De)
`DD, object`	An object of class "drawdown"
`show.extr`	Logical. if TRUE extreme drawdown will be calculated.
`lag`	Integer. Number of lag periods used for rolling calculation.
`rolling`	Logical. If TRUE extreme will be calculated on a moving window.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments accepted by the function `cplot` or `sma`.

Details

The function "ExtremeDD" is called inside "summary.drawdown".

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# load example time series
data(ex_ts)
x = ex_ts

# calculate drawdown - no plot
```

```
dd = drawdown(x, plots="no.plot")
# calculate drawdown - regular plot
dd = drawdown(x, plots="regular")
# calculate drawdown - smoothed plot with different color
dd = drawdown(x, plots="smooth", col="green")

# summary information and maximum drawdown
summary(dd)
# ... summary information and rolling maximum drawdown
summary(dd, rolling=TRUE, lag=10)
```

`dropn` *Drop N Terms from a Linear Regression Model*

Description

This is a conceptual extension of the function drop1 although the format of the output returned is different.
Iteratively removes N terms from the model.

Usage

```
dropn(mod, N = 1, ...)
```

Arguments

`mod`	A fitted model object
`N`	The number of terms to drop from the model.
`...`	Further arguments passed to drop1.

Value

The model obtained after the removal of N terms.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`drop1`.

Examples

```
# Generate some random data
N = 20;
x1 = rnorm(N);
x2 = rnorm(N);
x3 = rnorm(N);
x4 = rnorm(N);

# Define a model based on x1 and x3
```

```
y = x1 - 3*x3 + 0.5*rnorm(N);
# Estimate the full model
mod = lm(y ~ x1 + x2 + x3 + x4);
summary(mod)

# Remove the two worst terms
modred = dropn(mod, N = 2);
summary(modred)
```

`edwdist` *Edgeworth distribution*

Description

Simulate empirical Edgeworth distribution

Usage

```
EdgeWorthDist(init, Nsteps, p=0.5)
```

Arguments

`init`	init
`Nsteps`	Nsteps
`p`	p

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`edwprice` *Edgeworth option price*

Description

Option evaluation with Edgeworth adapted Binomial Tree

Usage

```
Edgeworth.price(init, under, strike, rfr, sigma, maty, yield)
```

Arguments

`init`	init
`under`	under
`strike`	strike
`rfr`	rfr
`sigma`	sigma
`maty`	maty
`yield`	yiels

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`ema` *Exponential Moving Average*

Description

Compute multiple Exponential Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
ema(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function Mmovav.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
Smoothing factor: lambda = 2/(win.size+1).

Value

A object of class 'ma' with attributes type = "EMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
ema(x, 10)
# compute moving average with multiple lags
ema(x, c(10,20))

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
ema(x, 30, plot = TRUE)
# multiple lags
ema(x, seq(5,50,10), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
ema(ex_fs, 30, plot=TRUE)
# multiple lags
ema(ex_fs, seq(5,50,10), plot=TRUE)

## End(Not run)
```

---

emat — *Trend corrected Exponential Moving Averages*

Description

Compute multiple Trend corrected Exponential Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
emat(X, win.size = NROW(X), alpha = 0.1, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`alpha`	weight for the trend correction (Default: 0.1)
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
EMAT is a dynamic model regulated by the smoothing factors lambda = 2/(win.size+1) and alpha.

Value

A object of class 'ma' with attributes type = "EMAT", 'lambda' and 'alpha':
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
emat(x, 10, alpha=0.5)
# compute moving average with multiple lags
emat(x, c(10,20), alpha=0.3)

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
emat(x, 15, plot = TRUE)
# multiple lags
emat(x, seq(5,30,5), plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
emat(ex_fs, 30, plot=TRUE)
# multiple lags
emat(ex_fs, seq(5,50,10), plot=TRUE)

## End(Not run)
```

`eom` *Ease of Movement oscillator*

Description

Compute Ease of Movement oscillator (Technical Analysis)

Usage

```
eom(Close, High = NULL, Low = NULL, Vol = NULL, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Vol`	Vector. Asset traded Volume.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`epma` *end Point Moving Averages*

Description

Computes multiple End-Points Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
epma(X, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable)
`win.size`	Vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function Movav

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
EPMA Weights are given by a win.size-long line with angular coefficient = -3 and intercept = 2*win.size-1

Value

A object of class 'Movav' with attributes type = "EPMA" and 'win.size' as from the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

Movav

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
epma(x, 10)
# compute moving average with multiple lags
epma(x, c(10,15,20))

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
epma(x, 30, plot = TRUE)
# multiple lags
epma(x, c(10,30,50), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
epma(ex_fs, 30, plot=TRUE)
# multiple lags
epma(ex_fs, c(10,30,50), plot=TRUE)

## End(Not run)
```

erf *Elder Ray force*

Description

Compute Elder Ray force (Technical Analysis)

Usage

```
erf(Close, High = NULL, Low = NULL, lag = 13, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

erfi *Elder Ray force index*

Description

Compute Elder Ray force index (Technical Analysis)

Usage

```
erfi(X, Volume, lag = 13, plot = FALSE, ...)
```

Arguments

`X`	X
`Volume`	Vector. Asset traded Volume.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`es` *Expected Shortfall*

Description

General ES, computed on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
ES(X, ...)

## Default S3 method:
ES(X
    , p = 0.05
    , probf = c("Normal", "T-Student", "Cornish-Fisher", "GPD-POT")
    , df = max(4, (kurt(X)+3))
    , trsh = -hVaR(X)
    , ...
    )
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	Vector of probabilities (Default: 0.05)
`probf`	Probability distribution (see details). Case insensitive, partial matching is supported.
`df`	Degrees of freedom for the Student T distribution (Default: max(4, (kurt(X)+3)))
`trsh`	vector of NCOL(X) thresholds used to identify the tail data for the GPD-POT method
`...`	Additional parameters passed to the functions 'cofit' and 'gpd.ES'.

Details

Accepted probability distributions:

- "Normal": Normal distribution.
- "T-Student": Student'T distribution.
- "Cornish-Fisher": Cornish-Fischer formula for quantiles estimation.
- "GPD-POT": Peak Over Threshold method, based on Generalised Pareto Distribution (EVT).

Value

A matrix length(p) by NCOL(X) of computed ES values, based on the input distribution.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

gpd.ES, mqt, cofit.

Examples

```
# Load sample asset data
data(ex_ptf);
# Compute ES on multiple confidence levels (Normal)
ES(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "Normal");

# T-Student
ES(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "T");

# Extreme Value Theory (GPD)
ES(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "GPD");
```

---

factor *Factorise variable*

Description

Factorise numerical variables according to defined number of bins

Usage

```
Factorise(X, nseg,
seg.type = c("freq_equal", "width_equal"),
na.replace = NULL)
extrBreak(var, Factors)
## S3 method for class 'Factorise'
print(x, ...)
```

Arguments

X	Numeric input matrix.
nseg	INTEGER / VECTOR. Number of segments to factorise numerical variables.
seg.type	CHARACTER. Type of segments to create. (Default: "equal frequencies")
na.replace	CHARACTER / NUMERIC. Value to replace missing. If NULL missing values are not considered in the computation.
var	Character. Name(s) of the variable(s) for which to extract the breaks.
Factors, x	an object of class "Factorise"
...	Further arguments to or from other methods.

Details

The function `extrBreak` allows to extract the breaks of one or more variables from an object of class Factorise.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# load example data set "credit"
data(ex_ptf)
## Create matrix of factorised variables
# one segment
fact = Factorise(ex_ptf, nseg = c(2,4), seg.type="f")
fact
# two segments
fact = Factorise(ex_ptf, nseg = c(2,4), seg.type="f")
fact

# load example data set
data(ex_credit)
# consider only the numerical variable
num = ex_credit[ ,c(3,6,14)]
# four segments
fact = Factorise(num, nseg = c(2,3,4,5), seg.type="f")
fact

# extract the breaks for one variable
extrBreak("duration", Factors=fact)
# extract the breaks for two varaibles
extrBreak(c("duration","age"), Factors=fact)
# try to extract the breaks for a variable that doesn't exist in the data...
extrBreak("sex", Factors=fact)
```

`fft` *Customised Fast Fourier Transform*

Description

Computes FFT on each column of X. For Financial series objects (class 'fs'), Close data is extracted.

Usage

```
FFT(x, ...)

## Default S3 method:
FFT(x
    , Fs = 1
    , half = FALSE
    , window = NULL
    , plot = TRUE
    , optimised = TRUE
```

```
    , ...
    )
```

Arguments

`x`	Matrix of data series (one column per variable).
`Fs`	Sampling frequency (Default: 1).
`half`	Logical. If TRUE, half spectrum indices are computed.
`window`	Function or character name of the window used to smooth the data (Default: NULL. Results in rectangular window).
`plot`	Logical. If TRUE, frequency spectrum is plotted.
`optimised`	Logical. If TRUE, the number of FFT evaluation points is the next integer (power of 2) that allows the fast computation
`...`	Additional parameters passed to the plot (in the default implementation)

Value

An object of the class 'FFT'. It is a complex matrix (same number of columns as x) of frequency data. The following attributes are attached to the object:

`Fs`	The input Fs parameter
`window`	The window function used to smooth the input data
`freq`	The frequencies where the FFT was evaluated
`fpoints`	The array indices where the frequency points relative to 'freq' are stored
`half`	The input half parameter.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load sample financial series data
data(ex_fs)

# Frequency Analysis - Full spectrum
FFT(ex_fs)

# Frequency Analysis - Half spectrum (right side) and use blackman windowing, remove area
FFT(ex_fs
    , half = TRUE
    , window = blackman
    , shaded = FALSE)

# Show periodicity instead of frequency, and use hamming window
FFT(ex_fs
    , half = TRUE
    , window = hamming
    , show.periodicity = TRUE)

# Use kaiser window, zoom in to show only 10% of the half frequency spectrum, use semilog
```

```
FFT(ex_fs
    , half = TRUE
    , window = kaiser
    , show.periodicity = TRUE
    , zoom = 10
    , semilog = TRUE)

# Multiple FFT on matrix input.
# Use Bartlet-Hann window, zoom in to show only 20% of the full frequency spectrum, use s
FFT(ex_fs[,]
    , window = barthann
    , zoom = 20
    , semilog = TRUE
    , shaded = FALSE)
```

---

finplot *Plot financial time series*

Description

Generic plotting for financial data. Produces a two panels plot

Usage

```
fin.plot(X
    , top.vars = c("Close", "High", "Low")
    , bottom.vars = "Volume"
    , style = c("default", "candlestick")
    , snames = attr(X, "SName")
    , xlabels = rownames(X)
    , main = ""
    , main2 = ""
    , ytitle = ""
    , ytitle2 = ""
    , theme.top = getCurrentTheme()
    , overrides = list(...)
    , theme.bottom = getCurrentTheme()
    , overrides2 = NULL
    , ...
    )
```

Arguments

`X`	Input matrix of data to be plotted.
`top.vars`	Indices or names of the columns for the top plot.
`bottom.vars`	Indices or names of the columns for the bottom plot.
`style`	Not used. For future releases.
`snames`	Names of the series being plotted.
`xlabels`	Labels for the x-axis.
`main`	Main title for the top plot.

main2	Main title for the bottom plot.
ytitle	Title for the y-axis (top plot).
ytitle2	Title for the y-axis (bottom plot).
theme.top	Theme parameters list for the top plot (Default: getCurrentTheme()).
overrides	List of parameters to override theme for the top plot. Only parameters that match those defined by the theme are overridden (DEFAULTlist(...)).
theme.bottom	Theme parameters list for the bottom plot.
overrides2	List of parameters to override theme for the bottom plot. (Default: NULL).
...	Additional parameters passed to the cplot function. Also used to quickly specify theme overrides.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

cplot.

Examples

```
# Load sample financial series data
data(ex_fs)
# Plot the data
plot(ex_fs)
# Change the style and color of the bottom chart
plot(ex_fs, overrides2 = list(type = "l", col = "grey"))
```

---

firsthit — *First hit of a Brownian motion*

Description

Calcualte probability and expected time to Hit an absorbing barrier for a Browniam motion

Usage

```
ProbHit(B, S0, mi, sigma)
FirstHit(B, S0, mi, sigma, geom=FALSE, nsim=500, plot=FALSE)
```

Arguments

B	Numeric. Barrier value.
S0	Initial level of the process.
mi	Drift value.
sigma	Volatility value.
geom	Logical. Type of BM to simulate, if TRUE simulate Geometric BM else Standard.
nsim	Integer. Number of simulations; needed to produce the plot
plot	Logical. If TRUE plot with simulated BM and the barrier is returned.

Value

`ProbHit` returns the probability of hitting the barrier. `FirstHit` returns the expected time period before the first hit.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`PDFHit`

Examples

```
# Calculate the probability to hit the barrier 5
# for increasing values of the drift.
ProbHit(B=1, S0=5, mi=0.05, sigma=1)
ProbHit(B=1, S0=5, mi=0.1, sigma=1)
ProbHit(B=1, S0=5, mi=0.3, sigma=1)
ProbHit(B=1, S0=5, mi=0.5, sigma=1)


# Calculate expected time before hitting the barrier 3.
# process starting from 0
S0 = 0
# positive drift
mi = 1
FirstHit(B=3, S0=S0, mi=mi, sigma=0.5, geom=FALSE, nsim=500, plot=TRUE)

# expected time before hitting a positive barrier (B=1)
# if the process has a negative drift ...
FirstHit(B=1, S0=S0, mi=-1, sigma=0.5, geom=FALSE)
# ... of course you will wait forever...
```

`fitvecar` *Extract Model Fitted Values from Vector AutoRegressive object*

Description

Generic method for extracting model fitted values from object of class 'VecAr'.

Usage

```
## S3 method for class 'VecAr'
fitted(object, ...)
```

Arguments

`object`	Instance of class 'VecAr'.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

VecAr, predict.mreg.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Extract fitted values
fitted(mod)
```

---

flogbuf — *Flush the log buffer to file*

Description

Flush the content of the log buffer to file and console.

Usage

```
flushLogBuffer(console = FALSE
                        , logfile = getLogFile(env = env)
                        , env = getOption("RAdamant")
                        )
```

Arguments

console	Logical. If TRUE, content is sent to console.
logfile	The path to the log file.
env	The environment where the info is stored (Default: getOption("RAdamant")).

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Save content of the log buffer to file and print content to console as well
flushLogBuffer(console = TRUE);
```

fmeas *Four Measures indexes*

Description

Calculate the Four Measures indexes

Usage

```
FourMeasures(PTF, ...)
## Default S3 method:
FourMeasures(PTF, PTF_M, rf = NULL, rfr = 0, ...)
## S3 method for class 'Capm'
FourMeasures(PTF, rfr = 0, ...)
```

Arguments

PTF	Input portfolio or an object of class "Capm"
PTF_M	Market/benchmark portfolio
rfr	risk free rate
rf	risk free asset
...	Further arguments to or from other methods

Value

Return a matrix containing the values for the following indexes: Sharpe, Treynor, Jensen and Appraisal

Author(s)

RAdamant Development Team `team@r-adamant.org`

See Also

Sharpe, Treynor, Jensen, Appraisal

fmlmreg *Extract formula from regression object*

Description

Extract formula from regression ("reg" / "mreg") object

Usage

```
## S3 method for class 'reg'
formula(x, ...)

## S3 method for class 'mreg'
formula(x, ...)
```

Arguments

`x`	An object of class "reg" / "mreg"
`...`	Further arguments passed to or from other methods.

Value

A formula if input x is an object of class "reg".
A list of formulas if x is an object of class "mreg".

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`mreg`

`forcidx` *Force index*

Description

Compute Force index (Technical Analysis)

Usage

```
forcidx(X, Volume, lag = 5, sth = TRUE,
sth.lag = 13, mov = sma, plot = FALSE, ...)
```

Arguments

`X`	X
`Volume`	Volume
`lag`	Integer. Number of lag periods.
`sth`	sth
`sth.lag`	sth.lag
`mov`	mov
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from othermethods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`frama` *Fractal Moving Average*

Description

Fractal Moving Average, computed on each column of the input data X and for each pair (fast.win[i], slow.win[i]).

Usage

```
frama(X, win.size = 10, tau = 4.6,
keep.lambda = FALSE, keep.ER = FALSE, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of window sizes (lags) (Default: 10).
`tau`	controls how the smoothing factor lambda is calculated (lambda = exp(tau*log(ER))) (Default: 4.6).
`keep.lambda`	Logical. If TRUE, adaptive smoothing factor lambda is returned as an attribute (Default: FALSE).
`keep.ER`	Logical. If TRUE, adaptive Efficiency Ratio ER is returned as an attribute (Default: FALSE).
`plot`	Logical. Return plot.
`...`	Additional parameters for future development.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'Movav' with attributes type = "FRAMA", 'lambda' and 'ER' as required and 'win.size' and 'tau' given by the corresponding input parameters:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`ema`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
frama(x, 20, tau=4.6)
# compute moving average with multiple lags
frama(x, c(40,50,60), tau=5.0)

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
frama(x, 20, tau=4.6, plot = TRUE)
# multiple lags
frama(x, c(10,15,30,50), tau = 4.0, plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
frama(ex_fs, 20, tau=4.6, plot = TRUE)
# multiple lags
frama(ex_fs, c(10,15,30,50), tau = 4.0, plot=TRUE)

## End(Not run)
```

---

fsevecar *VAR Forecast Standard Error*

Description

Compute forecast standard error for VAR model

Usage

```
FSE.VecAr(X, steps, ...)
```

Arguments

`X`	X
`steps`	steps
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`fulp` *Full price*

Description

Compute Full price (Technical Analysis)

Usage

```
fullP(Close, Open, High, Low, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`Open`	Vector. Open price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`funcomx` *Function comment*

Description

Given an input file, this functions created an index based commented version of the file.

Usage

```
func.comment.idx(control.df =
    data.frame(FNAME = c(), FCODE = c(),
    AREA = c(), SECTION = c(), CLASS = c()),
infile = NULL, incode = NULL, outfile = NULL, max.dgt = 3)
```

Arguments

`control.df`	List of function names. See Details
`infile`	Input file (Full path: Mandatory).
`incode`	Input code array (Alternative to infile: Mandatory). Each entry is considered to be a line of code.
`outfile`	Output commented file (Full path: Optional). If provided, an output file is generated.
`max.dgt`	Controls the number of digits to be used on each section of the comment.

Details

This data frame is a list of function names:

- FNAME = Name of the function
- FCODE = code identifier for the function. (a-Z)(0-9).
- AREA = Macro area (Description) classification for the function.
- SECTION = Section (Description) classification for the function (Sub-AREA)
- CLASS = The class of the returned object.

Value

String array where every entry is a line of code. Each original line of the input code is preceded by a special comment.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
#### EXAMPLE #####
tst = data.frame(FNAME = c("sd", "lm")
                , FCODE = c("SD", "LM")
                , AREA = c("s5", "s2")
                , SECTION = c("s1", "s1")
                , CLASS = c("c1", "c2")
                );

incode = rbind(paste("sd =", as.character(deparse(args(sd)))[1])
                , as.matrix(deparse(body(sd)))
                , ""
                , ""
                , paste("lm =", as.character(deparse(args(lm)))[1])
                , as.matrix(deparse(body(lm)))
               )
func.comment.idx(tst, incode = incode, max.dgt=3)
```

`funlcnt` *Modularity Analysis*

Description

Given a package name or a list of functions, for each function X in the package or the list it counts the lines of code, the number of subcalls made to any other function Y of the list/package and the number of other functions that make calls to the function X.
Results are plotted if requested.

Usage

```
func.line.cnt(package = NULL, plot = TRUE, ...)

## S3 method for class 'modularity'
plot(x
    , qtz.type = "linear"
    , qtz.nbins = 30
    , qtz.cutoff = 30
    , theme.params = getCurrentTheme()
    , overrides = list(...)
    , border = "transparent"
    , savepng = FALSE
    , savepath = getwd()
    , save.width = 480
    , save.height = 480
    , save.resolution = 72
    , ...
    )
```

Arguments

`package`	CHARACTER. Single name of the package to load or array list of function names.
`x`	An object of class "modularity".
`plot`	Logical. If TRUE, results are plotted on bar charts.
`qtz.type`	CHARACTER. qtz.type = "Linear" | "Log" | "None". Partial match on the value is attempted.
`qtz.nbins`	Integer. Number of bins to be computed. Used only when qtz.type is "Linear" or "Log" (Default: 30).
`qtz.cutoff`	Used only when qtz.type = "Log" (Default: 30). More granular binning below the cutoff point.
`theme.params`	A valid RAdamant Theme. See setThemeAttr for details. (Default: getCurrentTheme())
`overrides`	List of parameters used to override the theme. Only parameters that match those defined by the theme are overridden (Default: list(...))
`border`	Color used for the border line of the barplot.
`savepng`	Logical. If true, charts are saved to png file.
`savepath`	The path where png files are saved (Default: getwd()).
`save.width`	The image width of the png file. See png for details.
`save.height`	The image height of the png file. See png for details.
`save.resolution`	The image resolution of the png file. See png for details.
`...`	Alternative way to quickly override theme parameters.

Details

The parameter "qtz.type" controls the type of quantization used to set the bin size for the bar chart of the Code Length Distribution.
Values:

- If "Linear", qtz.nbins equispaced intervarls are computed.
- If "Log", qtz.nbins log-spaced intervals are computed based on qtz.cutoff.
- In any other case the bin size is set to 1.

The parameter "qtz.cutoff" controls how bins are computed when qtz.type = "Log": qtz.nbins equispaced intervals are computed on a log(x/qtz.cutoff) scale.
This creates more intervals/bins in the range 0 < x < qtz.cutoff.

Value

An object of the class "modularity". This is a data frame containing the stats for each function in the input list/package, with the following columns:

`fcn.name`	Name of the function.
`fcn.lines`	Number of lines of code.
`fcn.subcalls`	Number of distinct calls made to other functions.
`fcn.called`	Number of distinct functions using this function.

The following attribute is attached to the object:

`package`	The input package argument.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
## Not run:
# Modularity Analysis for the RAdamant package
rad = func.line.cnt(package = "RAdamant")
# Log quantization
plot(rad, qtz.type = "Log", qtz.cutoff = 10)

## End(Not run)
```

---

fwmovav *Front Weighted Moving Averages*

Description

`fw1`: Computes multiple Front Weighted 32 Day Moving Averages on the input data, one for each column X[, i].
`fw2`: Computes multiple Front Weighted 18 Day Moving Averages on the input data, one for each column X[, i].
`fw3`: Computes multiple Front Weighted 2 Day Moving Averages on the input data, one for each column X[, i].

Usage

```
fw1(X, plot = FALSE, ...)
fw2(X, plot = FALSE, ...)
fw3(X, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function movav.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'ma' with attributes type = "FW1/2/3" and 'weights' given by the FW1/2/3 filter weights:
- matrix of size NROW(X) by NCOL(X) where each column is the moving average of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`garch` *Garch*

Description

Estimate Generalised Autoregressive Conditional Eteroschedasticity models (Garch)

Usage

```
Garch(x, ...)
## Default S3 method:
Garch(x
                                , Y=NULL
                                , order=c(alpha=1,beta=1)
                                , n.init = NULL
                                , type=c("garch","mgarch","tgarch","egarch")
                                , prob=c("norm","ged","t")
                                , ...)
```

Arguments

`x`	Vevotr/Matrix. Univariate time series of returns.
`Y`	Exogenous regressors for the Mean Equation
`order`	Vector of integers. Arch and Garch parameters order. (Default: 1,1)
`type`	Type of Garch to be estimated: "garch", "mgarch", "tgarch", "egarch". (Default: "garch").
`prob`	Innovations probability density: "norm", "ged", "t". (Default: "norm")
`n.init`	Number of initial observation for calculating initial variance. If NULL the entire sample is used.
`...`	Further arguments accepted by the function `optim`.

Details

Available methods for object of class "Garch": print, logLik, vcov, predict, coef.

Value

An object of class "Garch" containing a list of the following elements:

Type	Type of Garch model estimated.
Order	Arch and Garch order.
Mean_Equation	Results for the mean equation.
Results	Results for the variance equation.
LogLik	Log-Likelihood value.
Vcov	Asymptotic covariance matrix (calculated from numerical Hessian)
Volatility_Persistence	Persistence of volatility
AIC	Akaike information criterion
Fitted	Matrix containing: Original return series, Fitted value from mean equation, Resiudual series, Innovations, Estimated variance.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

optim, newsimp, predict.Garch

Examples

```
# load example time series
data(ex_ts)
x = ex_ts

# Estimate Garch(1,1) model - normal distribution for the innovations
gg1 = Garch(x, type="garch", prob="norm")
gg1

# Estimate Garch(1,1) model - normal distribution for the innovations
gg1 = Garch(x, type="garch", prob="ged")
gg1

# Estimate TGarch(1,1) model - normal distribution for the innovations
gg2 = Garch(x, type="tgarch")
gg2

# Estimate EGarch(1,1) model - GED distribution for the innovations
gg3 = Garch(x, type="egarch", prob="g")
gg3
```

`garchlik` *GARCH likelihood functions*

Description

Calculate likelihood for Garch, TGarch, EGarch and MGarch models

Usage

```
like.mgarch(theta, x, Y, order, k, prob=c("norm","ged","t"))
like.garch(theta, ee, x, Y, order, k, prob = c("norm","ged", "t"))
like.tgarch(theta, ee, x, Y, order, k, prob = c("norm","ged", "t"))
like.egarch(theta, ee, x, Y, order, k, prob = c("norm","ged", "t"))
```

Arguments

`theta`	Vector of paramaters.
`ee`	Vector of innovations.
`x`	Original series of returns.
`k`	Number of mean equation regressors.
`Y`	Matrix of exogenous variables used for the mean equation.
`order`	Model parameter order
`prob`	Innovations probability density: "norm", "ged", "t". (Default: "norm")

Details

Those functions are called inside the main Garch function in order to obtain numerical optimisation of the input parameters.
The input parameter of the functions are calculated directly inside the Garch function (see `Garch`)

Value

Likelihood value

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gartest — *Garch residual tests*

Description

Compute ARCH-LM and Ljung-Box test for residual correlation

Usage

```
Archlm(x, lags, std=FALSE, plot.acf=FALSE)
LjungBox(x, lags, plot.acf = FALSE)
```

Arguments

`x`	Series of residual or an object of class "Garch".
`lags`	Number of lags to calculate the autocorrelation function.
`plot.acf`	Logical. If TRUE plot of autocorrelation function is returned.
`std`	Logical. If TRUE input residual will be standardised.

Author(s)

RAdamant Development Team `team@r-adamant.org`

See Also

`Garch`

Examples

```
# load example time series
data(ex_ts)
x = ex_ts

gg1 = Garch(x, order = c(1,1), type="garch", prob="norm")
# perform Ljung-Box test with 10 lags
LjungBox(gg1, 10)
# perform ARCH-LM test with 10 lags and show ACF plot
Archlm(gg1, 1, std=TRUE, plot.acf=TRUE)
```

gauss — *Gauss window*

Description

Computes Gauss window of given length

Usage

```
gauss(N, normalized = TRUE, sigma = 0.5)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.
`sigma`	Standard Deviation - Expansion factor. sigma <= 0.5.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Gauss window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Normalised Gauss window of size 100
x = gauss(100)
# Plot the window
cplot(x
    , main = "Gauss Window"
    , legend = attr(x, "type")
    )

# Generate a non-normalised window
y = gauss(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Gauss Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )

# Generate another window with smaller expansion factor
z = gauss(100, normalized = FALSE, sigma = 0.1)
# Compare the two expansion factors
cplot(cbind(y, z)
    , main = "Gauss Window"
    , legend = paste("Gauss (sigma = ", c(0.5, 0.1), ")")
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

`gdema` *Generalised Double EMA*

Description

Compute multiple Generalised Double EMA on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
gdema(X, win.size = NROW(X), alpha = 0.7, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`alpha`	weight in the interval [0, 1]. (Default: 0.7)
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
GDEMA is a weighted combination of EMA and DEMA: alpha*DEMA(X) + (1-alpha) * EMA(X).
Smoothing factor: lambda = 2/(win.size+1).

Value

A object of class 'ma' with attributes type = "GDEMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
gdema(x, 10)

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
gdema(x, 30, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
gdema(ex_fs, 15, plot=TRUE)
```

```
## End(Not run)
```

---

getacfci *Normal confidence intervals for correlation*

Description

Compute the Normal confidence intervals for correlation and partial autocorrelation data.

Usage

```
get.acf.ci(X, ci = 0.95)
```

Arguments

X	Instance of class 'acf' as returned by functions acf, pacf, ccf
ci	Confidence interval required (Default: 0.95)

Value

A vector containing the two symmetrical confidence intervals.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate some random integrated data
x = cumsum(rnorm(30));
# The confidence intervals in mcf are calculated using get.acf.ci
res = mcf(x)
# Extract the PACF part and compute the confidence intervals
get.acf.ci(res$PACF[[1]])
# Same as
get.acf.ci(pacf(x, plot = FALSE))
```

---

getfs *Download Financial Series data from Yahoo!*

Description

Download Yahoo! time series data and returns a Financial Series (fs) object.

Usage

```
get.fs(symbol = NULL
    , SName = NULL
    , from = as.Date("1950-01-01")
    , to = Sys.Date()
    , strip.spaces = TRUE
    , strip.char = "."
    )
```

Arguments

`symbol`	The input stock symbol.
`SName`	Name that will be assigned to the time series. If NULL (default) the name is retrieved from Yahoo!
`from`	Date object. The start date of the time series (Default: as.Date("1950-01-01")).
`to`	Date object. The end date of the time series (Default: Sys.Date()).
`strip.spaces`	Logical. If TRUE, spaces from SName are replaced with the value of strip.char (Default: TRUE).
`strip.char`	The character used to replaces spaces in SName (Default: ".").

Value

A financial Time Series object. This is a matrix of Yahoo! daily data with columns (Open, High, Low, Close, Volume, Adj.Close).
The following attributes are attached to the object:

`SName`	The Name/Description of the financial series.
`Symbol`	the input stock symbol.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Get Dow Jones quotes from Jan 2010
DowJones = get.fs("^DJI", from = as.Date("2010-01-01"))
DowJones
```

`getlmwgh` *Extract Linear Model Weights Percentages*

Description

Extract weights percentages of the coefficients of a linear model.

Usage

```
get.lm.weights(mod, pct = FALSE)
```

Arguments

`mod`	The model from which the regression weights percentages are calculated.
`pct`	Logical. If TRUE, weighs are returned in percentage terms

Value

A vector containing the weights percentages of the regression terms.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generte normalized data (unitary standard deviation)
x1 = Zscore(1:10);
x2 = Zscore(exp(x1));
# Create linear model (weights: 1/3 to x1 and 2/3 to x2)
y = x1 + 2*x2;

# Estimate the model
mod = lm(y ~ x1 + x2);
# Compute weigths
get.lm.weights(mod);
get.lm.weights(mod, pct = TRUE);
```

---

getpred *Extract Model Predictors*

Description

Extract the column names of the regression terms of a linear model

Usage

```
get.predictors(mod)
```

Arguments

`mod`	The model from which the regression terms are extracted.

Value

A vector containing the column names of the regression terms.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Formula
get.predictors(y ~ x1 + x2);

# Linear Model
x1 = 1:10;
x2 = log(x1);
y = x1 + x2
get.predictors(lm(y ~ x1 + x2))
```

`gevar` *GEV - VaR calculation*

Description

GEV - VaR calculation

Usage

```
gev.VaR(Xbmax, mu = NULL, xi = NULL, sigma = NULL, prob = 0.01, ...)
```

Arguments

`Xbmax`	Xbmax
`mu`	mu
`xi`	xi
`sigma`	sigma
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gevarci — *GEV - VaR calculation and Confidence Intervals*

Description

GEV - VaR calculation and Confidence Intervals

Usage

```
gev.VaR.ci(Xbmax, VaR = sum(gev.VaR.constraint(parms = c(0, xi, sigma),
type = "both", Xbmax = Xbmax, prob = prob))/2, xi = 0.1,
sigma = 1, alpha = 0.01, df = 3, prob = alpha[1], ...)
```

Arguments

`Xbmax`	Xbmax
`VaR`	VaR
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gevarcnt — *GEV - VaR Joint Confidence Intervals by Profile Likelihood*

Description

GEV - VaR Joint Confidence Intervals by Profile Likelihood

Usage

```
gev.VaR.contour(Xbmax,
VaR = sum(gev.VaR.constraint(parms = c(0, xi, sigma),
type = "both", Xbmax = Xbmax, prob = prob))/2, xi = 0.1,
sigma = 1, alpha = 0.01, df = 3, prob = alpha[1], ...)
```

Arguments

`Xbmax`	Xbmax
`VaR`	VaR
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

gevarcst *GEV - Domain range for the VaR parameter*

Description

GEV - Domain range for the VaR parameter

Usage

```
gev.VaR.constraint(parms, type = c("left", "right", "both"),
Xbmax, prob = 0.01, ...)
```

Arguments

`parms`	parms
`type`	type
`Xbmax`	Xbmax
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gevarg *GEV - VaR range grid for contour calculation*

Description

GEV - VaR range grid for contour calculation

Usage

```
gev.VaR.range(Xbmax,
VaR = sum(gev.VaR.constraint(parms = c(0, xi, sigma),
type = "both", Xbmax = Xbmax, prob = prob))/2, xi = 0.1,
sigma = 1, alpha = 0.01, df = 3, prob = alpha[1], ...)
```

Arguments

Xbmax	Xbmax
VaR	VaR
xi	xi
sigma	sigma
alpha	alpha
df	df
prob	prob
...	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

gevark *GEV - VaR Log Likelihood*

Description

GEV - VaR Log Likelihood

Usage

```
gev.VaR.like(parms, Xbmax, prob = 0.01, ...)
```

Arguments

`parms`	parms
`Xbmax`	Xbmax
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gevci` *GEV - Distribution fitting and Confidence Intervals*

Description

GEV - Distribution fitting and Confidence Intervals

Usage

```
gev.ci(Xbmax, mu = 0, xi = 0.1, sigma = 1, alpha = 0.01, df = 3, ...)
```

Arguments

`Xbmax`	Xbmax
`mu`	mu
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gevcont — *GEV - Joint Confidence Intervals by Profile Likelihood*

Description

GEV - Joint Confidence Intervals by Profile Likelihood

Usage

```
gev.contour(Xbmax
                            , mu = 0
                            , xi = 0.1
                            , sigma = 1
                            , alpha = 0.01
                            , df = 3
                            , ...)
```

Arguments

`Xbmax`	Xbmax
`mu`	mu
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gevlike — *GEV - Log Likelihood*

Description

GEV - Log Likelihood

Usage

```
gev.like(parms, Xbmax, ...)
```

Arguments

`parms`	parms
`Xbmax`	Xbmax
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`gevmcst` *GEV - Domain range for the mu parameter*

Description

GEV - Domain range for the mu parameter

Usage

```
gev.mu.constraint(parms
                                  , type = c("left", "right", "both")
                                  , Xbmax
                                  , ...)
```

Arguments

`parms`	parms
`type`	type
`Xbmax`	Xbmax
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`gevml` *GEV - Maximum Likelihood Parameters Estimation*

Description

GEV - Maximum Likelihood Parameters Estimation

Usage

```
gev.ml(Xbmax, init = c(0, 0.1, 1), ...)
```

Arguments

`Xbmax`	Xbmax
`init`	init
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gevrng` *GEV - Parameters range grid for contour calculation*

Description

GEV - Parameters range grid for contour calculation

Usage

```
gev.range(Xbmax
                        , mu = 0
                        , xi = 0.1
                        , sigma = 1
                        , alpha = 0.01
                        , df = 3
                        , ...)
```

Arguments

`Xbmax`	Xbmax
`mu`	mu
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gevsicst` *GEV - Domain range for the sigma parameter*

Description

GEV - Domain range for the sigma parameter

Usage

```
gev.sigma.constraint(parms
                                    , type = c("left", "right", "both")
                                    , Xbmax
                                    , parm.type = c("mu", "VaR", "ES")
                                    , prob = 0.01
                                    , ...)
```

Arguments

`parms`	parms
`type`	type
`Xbmax`	Xbmax
`parm.type`	parm.type
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gevxicst` *GEV - Domain range for the xi parameter*

Description

GEV - Domain range for the xi parameter

Usage

```
gev.xi.constraint(parms, type = c("left", "right", "both"),
Xbmax, parm.type = c("mu", "VaR","ES"), prob = 0.01, ...)
```

Arguments

`parms`	parms
`type`	type
`Xbmax`	Xbmax
`parm.type`	parm.type
`prob`	prob
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gini` *Gini index*

Description

Calculate Gini index based on the results of a classification model.

Usage

```
Gini(x, ...)
## Default S3 method:
Gini(x, ...)
## S3 method for class 'scorecard'
Gini(x, glob = TRUE, ...)
```

Arguments

`x`	An object of class "scorecard" or a matrix containing "Number of Goods" and "Number of bads"
`glob`	Logical. If TRUE the function returns the Gini index for the model otherwise, it returns a separate index for each variable
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# load example data set
data(ex_credit)

## Generate Score Card
data = ex_credit[ ,-1]
target = ex_credit[ ,1]
# Two examples of scorecard
sc2 = Score.card(X=data, Y=target, nseg = c(2,4))
sc3 = Score.card(X=data, Y=target, nseg = c(2:5))

# calculate global Gini
Gini(sc2, glob=TRUE)
Gini(sc3, glob=TRUE)
# calculate Gini for each variable
Gini(sc2, glob=FALSE)
Gini(sc3, glob=FALSE)
```

---

glogbuf — *Retrieve the content of the Log Buffer*

Description

Retrieve the content of the Log Buffer.

Usage

```
getLogBuffer(env = getOption("RAdamant"))
```

Arguments

`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Value

Returns the content of the log buffer.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Retrieve content of the log buffer.
getLogBuffer();
```

---

gmma   *Guppy's Multiple EMA*

Description

Compute Guppy's Multiple EMA on the input data, one for each column of X[, i].

Usage

```
gmma(X, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

GMMA is two sets (short and long window sizes) of six EMA:
- Short Windows: 3, 5, 8, 10, 12, 15
- Long Windows: 30, 35, 40, 45, 50, 60.

Value

A object of class 'ma' with attributes type = "GMMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*12 with twelve moving averagesfor each column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`ema`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute guppy moving averages
gmma(x)

## Not run:
```

```
# refine results of moving average
setCurrentTheme(1)
# single lag
gmma(x, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
gmma(ex_fs, plot=TRUE)

## End(Not run)
```

`gpdboot` *GPD - parameters bootstrapping*

Description

GPD - parameters bootstrapping

Usage

```
gpdboot(Xtail, trsh = 0, xi = NULL, sigma = NULL, nboots = 100, ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`nboots`	nboots
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdci *GPD - Distribution fitting and Confidence Intervals*

Description

GPD - Distribution fitting and Confidence Intervals

Usage

```
gpd.ci(Xtail, trsh = 0, xi = 0.1, sigma = 1, alpha = 0.01, df = 2, ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdcnt *GPD - Joint Confidence Intervals by Profile Likelihood*

Description

GPD - Joint Confidence Intervals by Profile Likelihood

Usage

```
gpd.contour(Xtail
                        , trsh = 0
                        , xi = 0.1
                        , sigma = 1
                        , alpha = 0.01
                        , df = 2
                        , ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

gpdes — *GPD - Expected Shortfall (ES) calculation*

Description

GPD - Expected Shortfall (ES) calculation

Usage

```
gpd.ES(Xtail, trsh = 0, xi = NULL, sigma = NULL, N, prob = 0.01, ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`N`	N
`prob`	prob
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdesci *GPD - ES calculation and Confidence Intervals*

Description

GPD - ES calculation and Confidence Intervals

Usage

```
gpd.ES.ci(Xtail, trsh = 0, ES = trsh + 10^-5, xi = 0.1,
alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

Xtail	Xtail
trsh	trsh
ES	ES
xi	xi
alpha	alpha
df	df
N	N
prob	prob
...	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

gpdescnt *GPD - ES Joint Confidence Intervals by Profile Likelihood*

Description

GPD - ES Joint Confidence Intervals by Profile Likelihood

Usage

```
gpd.ES.contour(Xtail, trsh = 0, ES = trsh + 10^-5,
xi = 0.1, alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`ES`	ES
`xi`	xi
`alpha`	alpha
`df`	df
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`gpdescst` *GPD - Domain range for the ES parameter*

Description

GPD - Domain range for the ES parameter

Usage

```
gpd.ES.constraint(parms
                                    , type = c("left", "right", "both")
                                    , trsh = 0
                                    , ...)
```

Arguments

`parms`	parms
`type`	type
`trsh`	trsh
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

gpdesfce *GPD - Log Likelihood 3D surface as a function of Expected Shortfall*

Description

GPD - Log Likelihood 3D surface as a function of Expected Shortfall

Usage

```
gpd.ES.surface(ES = NULL, xi = NULL, Xtail,
trsh = 0, N, prob = 0.01, grid.size = 100, alpha = 0.01, ...)
```

Arguments

`ES`	ES
`xi`	xi
`Xtail`	Xtail
`trsh`	trsh
`N`	N
`prob`	prob
`grid.size`	grid.size
`alpha`	alpha
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdesk *GPD - ES Log Likelihood*

Description

GPD - ES Log Likelihood

Usage

```
gpd.ES.like(parms, Xtail, trsh = 0, N, prob = 0.01, ...)
```

Arguments

`parms`	parms
`Xtail`	Xtail
`trsh`	trsh
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdesml` *GPD - Maximum Likelihood ES Estimation*

Description

GPD - Maximum Likelihood ES Estimation

Usage

```
gpd.ES.ml(Xtail, trsh = 0, N, init = c(1, 0.1), ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`N`	N
`init`	init
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdesrng — *GPD - ES range grid for contour calculation*

Description

GPD - ES range grid for contour calculation

Usage

```
gpd.ES.range(Xtail, trsh = 0, ES = trsh + 10^-5,
xi = 0.1, alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`ES`	ES
`xi`	xi
`alpha`	alpha
`df`	df
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdlk — *GPD - Log Likelihood*

Description

GPD - Log Likelihood

Usage

```
gpd.like(parms, Xtail, trsh = 0, ...)
```

Arguments

`parms`	parms
`Xtail`	Xtail
`trsh`	trsh
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

gpdml *GPD - Maximum Likelihood Parameters Estimation*

Description

GPD - Maximum Likelihood Parameters Estimation

Usage

```
gpd.ml(Xtail, trsh = 0, init = c(0.1, 1), ...)
```

Arguments

Xtail	Xtail
trsh	trsh
init	init
...	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

gpdrng *GPD - Parameters range grid for contour calculation*

Description

GPD - Parameters range grid for contour calculation

Usage

```
gpd.range(Xtail
                    , trsh = 0
                    , xi = 0.1
                    , sigma = 1
                    , alpha = 0.01
                    , df = 2
                    , ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`alpha`	alpha
`df`	df
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdsfc` *GPD - Log Likelihood 3D surface*

Description

GPD - Log Likelihood 3D surface

Usage

```
gpd.surface(xi = NULL, sigma = NULL, Xtail,
trsh = 0, grid.size = 100, alpha = 0.01, ...)
```

Arguments

`xi`	xi
`sigma`	sigma
`Xtail`	Xtail
`trsh`	trsh
`grid.size`	grid.size
`alpha`	alpha
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdsgcnt` *GPD - Domain range for the sigma parameter*

Description

GPD - Domain range for the sigma parameter

Usage

```
gpd.sigma.constraint(parms, type = c("left", "right", "both"),
Xtail, trsh = 0, ...)
```

Arguments

`parms`	parms
`type`	type
`Xtail`	Xtail
`trsh`	trsh
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdvar` *GPD - VaR calculation*

Description

GPD - VaR calculation

Usage

```
gpd.VaR(Xtail, trsh = 0, xi = NULL, sigma = NULL, N, prob = 0.01, ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`xi`	xi
`sigma`	sigma
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdvarci` *GPD - VaR calculation and Confidence Intervals*

Description

GPD - VaR calculation and Confidence Intervals

Usage

```
gpd.VaR.ci(Xtail, trsh = 0, VaR = trsh + 10^-5,
xi = 0.1, alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`VaR`	VaR
`xi`	xi
`alpha`	alpha
`df`	df
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdvarcn — *GPD - VaR Joint Confidence Intervals by Profile Likelihood*

Description

GPD - VaR Joint Confidence Intervals by Profile Likelihood

Usage

```
gpd.VaR.contour(Xtail, trsh = 0, VaR = trsh + 10^-5,
xi = 0.1, alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`VaR`	VaR
`xi`	xi
`alpha`	alpha
`df`	df
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

gpdvarct — *GPD - Domain range for the VaR parameter*

Description

GPD - Domain range for the VaR parameter

Usage

```
gpd.VaR.constraint(parms
                                , type = c("left", "right", "both")
                                , trsh = 0
                                , ...)
```

Arguments

`parms`	parms
`type`	type
`trsh`	trsh
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdvarg` *GPD - VaR range grid for contour calculation*

Description

GPD - VaR range grid for contour calculation

Usage

```
gpd.VaR.range(Xtail, trsh = 0, VaR = trsh + 10^-5,
xi = 0.1, alpha = 0.01, df = 2, N, prob = alpha[1], ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`VaR`	VaR
`xi`	xi
`alpha`	alpha
`df`	df
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdvarlk` *GPD - VaR Log Likelihood*

Description

GPD - VaR Log Likelihood

Usage

```
gpd.VaR.like(parms, Xtail, trsh = 0, N, prob = 0.01, ...)
```

Arguments

`parms`	parms
`Xtail`	Xtail
`trsh`	trsh
`N`	N
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`gpdvarml` *GPD - Maximum Likelihood VaR Estimation*

Description

GPD - Maximum Likelihood VaR Estimation

Usage

```
gpd.VaR.ml(Xtail, trsh = 0, N, init = c(1, 0.1), ...)
```

Arguments

`Xtail`	Xtail
`trsh`	trsh
`N`	N
`init`	init
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`gpdvarsf` *GPD - Log Likelihood 3D surface as a function of VaR*

Description

GPD - Log Likelihood 3D surface as a function of VaR

Usage

```
gpd.VaR.surface(VaR = NULL, xi = NULL, Xtail,
trsh = 0, N, prob = 0.01, grid.size = 100, alpha = 0.01, ...)
```

Arguments

`VaR`	VaR
`xi`	xi
`Xtail`	Xtail
`trsh`	trsh
`N`	N
`prob`	prob
`grid.size`	grid.size
`alpha`	alpha
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

gpdxicst — *GPD - Domain range for the xi parameter*

Description

GPD - Domain range for the xi parameter

Usage

```
gpd.xi.constraint(parms, type = c("left", "right", "both"),
Xtail, trsh = 0, N, parm.type = c("sigma", "VaR", "ES"),
prob = 0.01, ...)
```

Arguments

`parms`	parms
`type`	type
`Xtail`	Xtail
`trsh`	trsh
`N`	N
`parm.type`	parm.type
`prob`	prob
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

grad — *Compute numerical gradient of a function*

Description

Plotting tools

Usage

```
grad(func = NULL
                , x
                , scalar = TRUE
                , eps = sqrt(.Machine$double.neg.eps)
                , ...)
```

Arguments

`func`	func
`x`	x
`scalar`	scalar
`eps`	eps
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`grangcas` *Granger Causality test*

Description

Perform Granger causality test for parameters of VAR model

Usage

```
## S3 method for class 'VecAr'
GrangCas(X, cause = colnames(coef(X)), digits = 3, ...)
```

Arguments

`X`	An object of class "VecAr"
`cause`	Vector of character. Name of the variables to be used as "cause". By default all the variables are tested.
`digits`	number of digits to be printed.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

grautil — *RAdamant Plot Utility Functions*

Description

Utility functions for internal plotting functions.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

hamming — *Hamming window*

Description

Computes Hamming window of given length

Usage

```
hamming(N, normalized = TRUE)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Hamming window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Normalised Hamming window of size 100
x = hamming(100)
# Plot the window
cplot(x
    , main = "Hamming Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = hamming(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Hamming Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
```

```
    , xlab.srt = 0
    )
```

hann *Hann window*

Description

Computes Hann window of given length

Usage

```
hann(N, normalized = TRUE)
```

Arguments

N	Window length.
normalized	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Hann window.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate a Normalised Hann window of size 100
x = hann(100)
# Plot the window
cplot(x
    , main = "Hann Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = hann(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Hann Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

heas — *Heikin - Ashi techniques*

Description

Compute Heikin - Ashi techniques (Technical Analysis)

Usage

```
he_as(Close, Open, High, Low, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`Open`	Vector. Open price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

hes — *Historical Expected Shortfall*

Description

Compute historical ES on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
hES(X, p = 0.05, centered = FALSE)
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	vector of probabilities (Default: 0.05)
`centered`	Logical. If TRUE, input data are standardised prior to compute ES.

Value

A matrix length(p) by NCOL(X) of computed historical VaR

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Load sample Financial series data
data(ex_fs);
# Compute Historical ES (5% confidence level) on 1-day Returns
hES(Ret(ex_fs));

# Generate some random data
X = cbind(rnorm(1000), rnorm(1000, sd = 2))
# Compute multiple Historical ES (1%, 2.5%, 5% confidence levels)
hES(X, p = c(1, 2.5, 5)/100);
```

---

hhv *Highest high*

Description

Compute Highest high (Technical Analysis)

Usage

```
hhv(X, lag, na.rm = TRUE)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`na.rm`	na.rm

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`hill` *Hill function*

Description

Approximation of the shape parameter (xi) of the Generalised Pareto distribution.

Usage

```
Hill(X, trsh = hVaR(X))
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`trsh`	Vector of NCOL(X) thresholds used to identify the tail data for the estimation.

Value

A matrix 1 by NCOL(X) of computed shape parameters

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`hma` *Hull Moving Averages*

Description

Compute multiple Hull Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
hma(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable)
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Further arguments to or from other methods

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
HMA is a combination of WMA: WMA(2*WMA(X, win.size/2) - wma(X, win.size), sqrt(win.size)).

Value

A object of class 'ma' with attributes type = "HMA" and 'win.size' as from the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

wma

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
hma(x, 10)
# compute moving average with multiple lags
hma(x, c(10,20))

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
hma(x, 30, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
hma(ex_fs, 30, plot=TRUE)


## End(Not run)
```

---

hroi *Historical Returns on Investment*

Description

Computes historical returns on investment and two-sided VaR. Analysis of the performance of the returns as a function of the holding period. For Financial series objects (class 'fs'), Close data is processed.

Usage

```
hroi(X
    , lag = 1
    , mode = c("auto", "range", "selected")
    , autolag.start = 1
    , range.step = 1
    , log = TRUE
    , VaR.type = "norm"
    , p = 0.05
    , ...
    )
```

Arguments

`X`	Input matrix of data to be plotted.
`lag`	The maximum lag used to compute returns (Default: 1).
`mode`	Controls how the lags are computed. See details.
`autolag.start`	Starting lag value for the case where mode = "auto" (Default: 1). See details.
`range.step`	Lag increment used for the case where mode = "range" (Default: 1). See details.
`log`	Logical. If TRUE, log returns are computed. DEFAULT = TRUE.
`VaR.type`	The distribution used for VaR calculation. See `VaR` for details.
`p`	The confidence interval used for VaR calculation. (Default: 0.05)
`...`	Additional parameters passed to the VaR function.

Details

For each input time series, returns are calculated for multiple lags, hence average and two-sided Value at Risk (Profit & Loss with p% confidence interval) are computed on the returns.
The number and the way lags are computed is controlled by the mode parameter:

- auto: All lags between autolag.start and max(lag) (DEFAULT option)
- range: All lags between min(lag) and max(lag) with increment given by range.step
- selected: Only selected lags are calculated.

Value

An instance of the class 'roi'. This is a list of length given by the number of columns of the input X. Each entry is a matrix with columns [Return (Avg.), VaR (Profit), VaR (Loss)] where the rows are calculated for each lag. The following attributes are attached to the object:

`log`	The input log parameter.
`lag`	The lags for which returns are computed.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Ret`, `VaR`, `plot.roi`.

Examples

```
# Load sample financial series data
data(ex_fs)

# Historical returns for all lags between 1 and 10 days
hroi(ex_fs, lag = 10)

# Historical returns for lags between 2 and 10 with increment 2
hroi(ex_fs, lag = c(2, 10), mode = "range", range.step = 2)

# Historical returns for selected lags
hroi(ex_fs, lag = c(2, 5, 10), mode = "selected")

# Analyse the performance of the returns up to 200 days and plot results
plot(hroi(ex_fs, lag = 200, log = FALSE), xlab.srt = 0)
```

---

hvar *Historical Value at Risk*

Description

Compute historical VaR on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
hVaR(X, p = 0.05, centered = FALSE)
```

Arguments

X	Input matrix/sequence. Sequences are treated as one column matrices.
p	vector of probabilities (Default: 0.05)
centered	Logical. If TRUE, input data are standardised prior to compute VaR.

Value

A matrix length(p) by NCOL(X) of computed historical VaR

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Load sample Financial series data
data(ex_fs);
# Compute Historical VaR (5% confidence level) on 1-day Returns
hVaR(Ret(ex_fs));

# Generate some random data
X = cbind(rnorm(1000), rnorm(1000, sd = 2))
# Compute multiple Historical VaR (1%, 2.5%, 5% confidence levels)
hVaR(X, p = c(1, 2.5, 5)/100);
```

ichkh *Ichimoku Kinko Hyo*

Description

Compute Ichimoku Kinko Hyo (Technical Analysis)

Usage

```
Ichkh(Close, High, Low, plot = FALSE, ...)
```

Arguments

`Close`	close
`High`	high
`Low`	low
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

impulse *Unitary impulse*

Description

Generates an impulse sequence of specified length

Usage

```
impulse(N, value = 1)
```

Arguments

`N`	Length of the impulse
`value`	value of the impulse (Default: 1)

Value

Impulse sequence of specified length

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

in2woe *Data to Weight of Evidence*

Description

Transform input data according to weight of evidence

Usage

```
input2woe(data, nseg, woe, ...)
```

Arguments

`data`	MATRIX or DATA.FRAME. Input data.
`nseg`	Integer of Vector. Number of segment to split the numerical variables.
`woe`	A matrix of results created by the function `WeightEvid`
`...`	Further parameter for the function `Factorise`

Details

Input data can contain both numerical and categorical variables. Numerical variables will be factorised according with the specified number of segments; categorical variables will be processed as they are (no aggregation for the existing classes).
The factorisation of the numerical variables is performed by the function `Factorise`.
Each value in the input data will be replaced with the corresponding Weight of Evidence.

Value

A matrix with the same number of rows of the input data and number of columns given by:
Number of categorical variables + Number of numerical variables * Number of segments.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# load example data set "credit"
data(ex_credit)
# calculate weight of evidence
input = ex_credit[ ,-1]
target = ex_credit[ ,1]
woe = WeightEvid(data=input, target=target, nseg = 2:3, missing=FALSE)
# quick look of the results got from WeightEvid
head(woe)
# recode input data according to weight of evidence calculation
new = input2woe(data = input, nseg=2:3, woe=woe)
# quick look of the new data
head(new)
```

`inertia` *Inertia oscillator*

Description

Compute Inertia oscillator (Technical Analysis)

Usage

```
Inertia(X, lag, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`invlogit` *Inverse Logit transformation*

Description

Inverse Logit transformation

Usage

```
inv.logit(y)
```

Arguments

`y`	y

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`irsvecar` *VAR Impulse response*

Description

Compute Impulse response function and Wold decomposition for VAR model

Usage

```
IRS.VecAr(X, imp, resp = NULL, steps = 5, cum = TRUE, ortho = FALSE, ...)
PHI.VecAr(X, steps, ortho = FALSE, ...)
```

Arguments

`X`	An object of class "VecAr".
`imp`	Vector of characters. Impulse variable(s).
`resp`	Vector of characters. Response variable(s).
`steps`	Integer. Number of forward steps.
`cum`	Logical. If TRUE cumulated impulse will be returned.
`ortho`	Logical. If TRUE orthogonal impulse will be returned.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
data(ex_ptf)
colnames(ex_ptf)
X = ex_ptf[ ,1:4]
# estimate VAR(2) model
var = VecAr(X, ar.lags=1:2, type="const", exog = NULL)

PHI.VecAr(var, steps=10, ortho=TRUE)

# Impulse response function - single inpulse
imp = "Asset_1"
resp = c("Fund", "Asset_1", "Asset_2", "Asset_3")
im = IRS.VecAr(var, imp=imp, resp=resp, steps=10, ortho=TRUE)
im
# view plots
cplot(im[[1]], lwd=2)
```

isfs — *Check for inheritance from Financial Series class*

Description

Check for inheritance from Financial Series class

Usage

```
is.fs(X)
```

Arguments

X	The object to be checked.

Author(s)

RAdamant Development Team <team@r-adamant.org>

jbtest — *Jaques-Brera normality test*

Description

Compute Jaques-Brera normality test for each column of X

Usage

```
JB.test(X, plot.hist=FALSE)
```

Arguments

X	Matrix of data series (one column per variable)
plot.hist	Logical. Return histogram.

Value

Matrix of Jaques-Brera scores and P-Value

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

kurt, skew

`jensen` *Jensen index*

Description

`Jensen`: Calculate Jensen index for a portfolio. `Jensen.Capm`: Get Jensen index from an object of class "Capm".

Usage

```
Jensen(PTF, ...)
## Default S3 method:
Jensen(PTF, PTF_M, rf = NULL, rfr = 0, ...)
## S3 method for class 'Capm'
Jensen(PTF, rfr = 0, ...)
```

Arguments

`PTF`	Input portfolio or an object of class "Capm"
`PTF_M`	Market/benchmark portfolio
`rfr`	risk free rate
`rf`	risk free asset
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Sharpe`, `Treynor`, `Appraisal`

`jrbtree` *JR Binomial Tree*

Description

Option evaluation with Jarrow and Rudd Binomial Tree

Usage

```
JR.BinTree(Nsteps, p=0.5, under, strike, rfr, sigma,
maty, yield, life, ret.steps = FALSE)
```

Arguments

`Nsteps`	Nsteps
`p`	Probability for each step; by default the stpes are supposed to equiprobable (p = 0.5)
`under`	Underlying asset price.
`strike`	Strike/Exercise price.
`rfr`	Risk free rate (continuos).
`sigma`	Assets standard deviation - annualised volatility.
`maty`	Period of maturity.
`yield`	Dividend yield (continuos).
`life`	Option life.
`ret.steps`	Logical. If TRUE the calculated steps (step matrix) are returned.

Value

List of results containing the following elements:

`Price_eval`	: Estimated option value at each step.
`Moments`	: Moments of the distribution of the share returns (both Black & Scholes and JR values are displayed).
`Values`	: Option estimated values (both Black & Scholes and JR values are displayed).
`Price_Path`	: Step matrix containing the expected share price at each step.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`BS.price`, `StepMat`, `CRR.BinTree`

Examples

```
# set option parameters
under = 105
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03
life = 0.5

# estimate option price using Jarrow and Rudd Binomial Tree (10 steps)
jrt = JR.BinTree(Nsteps=10
                    , p=0.5
                    , under
                    , strike
                    , rfr
                    , sigma
                    , maty
                    , yield
```

```
                        , life)
jrt$Values
# ... confront results with B&S method
BS.price(under, strike, rfr, sigma, maty, yield)
# get step matrix
jrt = JR.BinTree(Nsteps=10
                        , p=0.5
                        , under
                        , strike
                        , rfr
                        , sigma
                        , maty
                        , yield
                        , life
                        , ret.steps=TRUE)
jrt$Price_Path
```

`kaiser` *Kaiser window*

Description

Computes Kaiser window of given length (Discrete Prolate Spheroidal Sequence approximation).

Usage

```
kaiser(N, normalized = TRUE, alpha = 3)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.
`alpha`	Shape factor (Default: 3).

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Kaiser window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Kaiser window of size 100
x = kaiser(100, FALSE)
# Plot the window
cplot(x
    , main = "Kaiser Window"
    , legend = attr(x, "type")
    )

# Generate another window with different smoothing factor
```

```
y = kaiser(100, normalized = FALSE, alpha = 6)
# Compare the two windows
cplot(cbind(x, y)
    , main = "Kaiser Window"
    , legend = paste("Kaiser (alpha = ", c(3, 6), ")", sep = "")
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

---

kama *Kauffman Adaptive Moving Average*

Description

Kauffman Adaptive Moving Average, computed on each column of the input data X and for each pair (fast.win[i], slow.win[i]).

Usage

```
kama(X, fast.win = 2, slow.win = 30, lag = 5,
keep.lambda = FALSE, keep.ER = FALSE, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`fast.win`	vector of fast window sizes (fast lags) (Default: 2)
`slow.win`	vector of slow window sizes (slow lags) (Default: 30)
`lag`	vector of lags used to compute Kauffman efficiency ratio (Default: 5). Recycled to be of equal length as fast and slow lags if necessary
`keep.lambda`	Logical. If TRUE, adaptive smoothing factor lambda is returned as an attribute (Default: FALSE)
`keep.ER`	Logical. If TRUE, adaptive Efficiency Ratio ER is returned as an attribute (Default: FALSE)
`plot`	Logical. Return plot.
`...`	Further arguments to or from other methods.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

An object of class 'Movav' with attributes type = "KAMA", 'lambda' and 'ER' as required and 'fast.win', 'slow.win' and 'lag' given by the corresponding input parameters:
- matrix of size NROW(X) by NCOL(X)*length(fast.win) where each column is the moving average of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ama

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
kama(x, fast.win=5, slow.win=20, lag=10:20)

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
# compute moving average with single lag
kama(x, fast.win=5, slow.win=20, lag=10:20, plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
kama(ex_fs, fast.win=5, slow.win=20, lag=5, plot=TRUE)

## End(Not run)
```

---

kelt *Keltner channel*

Description

Compute Keltner channel (Technical Analysis)

Usage

```
kelt(Close, High, Low, mult = 2, plot = FALSE, ...)
```

Arguments

Close	Vector. Close price.
High	Vector. High price.
Low	Vector. Low price.
mult	mult
plot	Logical. If TRUE plot is returned.
...	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

kri — *Kairi Relative Index*

Description

Compute Kairi Relative Index (Technical Analysis)

Usage

```
kri(X, lag1 = 10, lag2 = 20, plot = FALSE, ...)
```

Arguments

`X`	X
`lag1`	lag1
`lag2`	lag2
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

kurtskew — *Kurtosis and Skewness*

Description

`kurt`: Compute the excess kurtosis for each column of X `skew`: Compute the skewness for each column of X

Usage

```
kurt(X, pval = FALSE)
skew(X, pval = FALSE)
```

Arguments

`X`	Matrix of numeric data series (one column per variable).
`pval`	Logical. Return P-Value.

Value

Matrix of Excess Kurtosis / Skewness and P-Value

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`JB.test`

`kvo` *Klinger oscillator*

Description

Compute Klinger oscillator (Technical Analysis)

Usage

```
kvo(Close, High = NULL, Low = NULL,
Vol = NULL, cumulative = FALSE, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Vol`	Vector. Asset traded Volume.
`cumulative`	cumulative
`plot`	LOGICAL. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`lagret` *Time Series Operators*

Description

`Ret`: Compute N-points Returns on each column of the input matrix.
`Lag`: Compute lag on each column of the input matrix.
`Diff`: Compute lagged difference on each column of the input matrix.
`MDiff`: Compute Multiple lagged differences on each column of the input matrix. \ cr `MLag`: Compute Multiple lags on each column of the input matrix

Usage

```
Ret(X
    , lag = 1
    , log = FALSE
    , mode = "selected"
    , na.rm = FALSE
    , plot = FALSE
    , ...)

Lag(X
    , lag = 1
    , na.rm = FALSE
    , padding = NA)

Diff(X
    , lag = 1
    , padding = NA
    , na.rm = FALSE)

MDiff(X
      , lag = 1
      , padding = NA
      , mode = c("auto", "range", "selected")
      , na.rm = FALSE)

MLag(X
    , lag = 1
    , na.rm = FALSE
    , padding = NA
    , mode = c("auto", "range", "selected")
    , autolag.start = 1)
```

Arguments

`X`	Input data (i.e. matrix/vector of prices)
`lag`	INTEGER or VECTOR. number of lags (it can be both positive and negative)
`log`	BOOLEAN: compute log-returns

`na.rm`	BOOLEAN: remove NAs
`plot`	BOOLEAN: return plot
`padding`	value to replace removed observations
`mode`	mode of using the vector of lags
`autolag.start`	autolag.start
`...`	Further arguments to or from other methods

Details

Sequences are treated as one-column matrices.
The parameter "mode" allows to control the calculation when the parameter is passed as a vector:

- auto: only the first element is used;
- range: if the lag arguments is composed of two numbers, the computation is performed for all the integers contained in the interval, ex: lag = c(4,10) allow to calculate all the lags between 4 and 10;
- selected: the computation is done only for the lag specified in the argument.

Value

A matrix (n.obs X n.lag) containing lagged /differenced time series or returns

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`plot.ret`

Examples

```
# load an example dataset containing financial daily prices
data(ex_fs)
x = ex_fs[ ,1:4]

# compute multiple multiple lags for single time series
# different uses of the parameter "mode"
res = MLag(x[,1], lag = c(4,8), mode="range")
res[1:10, ]
res = MLag(x[,1], lag = c(4,8), mode="selected")
res[1:10, ]
res = MLag(x[,1], lag = 4, mode="auto")
res[1:10, ]

## SINGLE LAG
# calculate return for single time series
res = Ret(x[ ,1], lag=4, log=TRUE, na.rm=TRUE)
res[1:10, ,drop=FALSE]

# calculate return for multiple time series
res = Ret(x, lag=10, log=TRUE, na.rm=TRUE)
```

```
res[1:10, ,drop=FALSE]

## MULTIPLE LAGS
# calculate return for single time series
res = Ret(x[ ,1], lag=c(2,4,6,8), mode = "selected", log=TRUE, na.rm=TRUE)
res[1:10, ,drop=FALSE]

# calculate return for multiple time series
res = Ret(x[, 1:2], lag=c(2,4,6,8), mode = "selected", log=FALSE, na.rm=FALSE)
res[1:10, ,drop=FALSE]


## PLOT RESULTS
# calculation and plot for single series
Ret(x[,1]
    , lag = 5
    , mode = "selected"
    , plot=TRUE
    , style="bar"
    , main="Returns - 5 Lags")

# calculation and plot for multiple series
par(mfrow=c(2,2))
Ret(x
    , lag = 5
    , mode = "selected"
    , plot=TRUE
    , style="bar"
    , main="Returns - 5 Lags")

## Not run:
# get APPLE financial series
symbol.lookup("Apple")
APPLE = get.fs("AAPL", from=as.Date("2008-06-01"), to=as.Date("2011-04-01"));
RAPPLE = Ret(APPLE
                , mode = "selected"
                , plot = TRUE
                , style = "bar"
                ,  ylab.fmt = .3
                , na.rm = TRUE)
RAPPLE;

## End(Not run)
```

`lanczos` *Lanczos window*

Description

Computes Lanczos window of given length

Usage

```
lanczos(N, normalized = TRUE)
```

Arguments

`N`	Window length.
`normalized`	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Lanczos window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate a Normalised Lanczos window of size 100
x = lanczos(100)
# Plot the window
cplot(x
    , main = "Lanczos Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = lanczos(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Lanczos Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

`lew` *Moving window*

Description

Apply a given function to an extending window of the lagged data series of the input matrix, each column separately.

Usage

```
lew(X, lag = 0, padding = NA, na.rm = FALSE,
func = NULL, is.cumulative = TRUE, ...)
```

Arguments

`X`	Input matrix/sequence
`lag`	vector of integer lags. If lag >= 0 data are shifted to the right, else to the left. (Default: 0)
`padding`	value used to initialise the output matrix (Default: NA)

`na.rm`	Logical. If TRUE, N-lag entries are removed from the output (Default: FALSE)
`func`	function applied to the extending data window (Default: NULL)
`is.cumulative`	Logical. If TRUE it the function provided must be cumulative by itself (like cummax, cummin, etc..) (Default: TRUE)
`...`	Additional parameters accepted by the function 'func'

Details

Sequences are treated as one-column matrices

Value

A matrix where func has been applied on increasing data windows for each column of X. Number of rows depends on the na.rm parameter. Number of columns is NCOL(X)

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`cumSum`, `cumMin`, `cumMax`, `cumSd`, `cumVar`

`liftgain` *Classification model accuracy plots*

Description

Plot cumulative Gain, Lift chart and ROC curve for a classification model

Usage

```
Gain(x, ...)
Lift(x, ...)
ROCplot(x, ...)
## S3 method for class 'scorecard'
Gain(x, pc = 0.1, ...)
## S3 method for class 'scorecard'
Lift(x, pc = 0.1, ...)
## S3 method for class 'scorecard'
ROCplot(x, ...)
```

Arguments

`x`	An object of class "scorecard"
`pc`	Numeric. A value indicating the perentile used to create data points.
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Score.card`

Examples

```
# load example data set
data(ex_credit)

## Generate Score Card
data = ex_credit[ ,-1]
target = ex_credit[ ,1]

# Two examples of socrecards
sc2 = Score.card(X=data, Y=target, nseg = c(2,4))
# Three segments for numerical variables
sc3 = Score.card(X=data, Y=target, nseg = c(2,3,4))

# Lift chart
Lift(sc2)
Lift(sc3)
# Cumualtive Gain
Gain(sc2)
Gain(sc3)
# ROC plot
ROCplot(sc2)
ROCplot(sc3)
```

`llv` *Lowest low*

Description

Compute Lowest low (Technical Analysis)

Usage

```
llv(X, lag, na.rm = TRUE)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`na.rm`	na.rm

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

logger *Main logging function*

Description

Send the input message to console and log file.

Usage

```
Logger(message = ""
      , from = deparse(sys.call(sys.parent()))
      , level = 1
      , line = NA
      , env = getOption("RAdamant")
      , console = getConsoleLogging(env = env)
      , logfile = getLogFile(env = env)
      )
```

Arguments

message	Message to be logged.
from	The level in the call stack from which the log message was generated.
level	The debug level (importance) of the input message (level >= 1).
line	The code line number that the message refers to.
env	The environment where the logging options are stored.
console	Logical. If TRUE, the message is sent to console.
logfile	The filename where the log information is saved.

Note

This is an internal logging function. It is supposed to be called from other functions.

Author(s)

RAdamant Development Team <team@r-adamant.org>

logit *Logit transformation*

Description

Logit transformation

Usage

```
logit(x, adjust = 5e-05)
```

Arguments

x	x
adjust	adjust

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

lrbtree *LR Binomial Tree*

Description

Option evaluation with Leinsen and Reimer Binomial Tree

Usage

```
LR.BinTree(Nsteps, under, strike, rfr,
sigma, maty, yield, life, ret.steps = FALSE)
```

Arguments

Nsteps	Nsteps
under	Underlying asset price.
strike	Strike/Exercise price.
rfr	Risk free rate (continuos).
sigma	Assets standard deviation - annualised volatility.
maty	Period of maturity.
yield	Dividend yield (continuos).
life	Option life.
ret.steps	Logical. If TRUE the calculated steps (step matrix) are returned.

Value

List of results containing the following elements:

- `Price_eval` : Estimated option value at each step.
- `Moments` : Moments of the distribution of the share returns (both Black & Scholes and CRR values are displayed).
- `Values` : Option estimated values (both Black & Scholes and LR values are displayed).
- `Price_Path` : Step matrix containing the expected share price at each step.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`BS.price`, `StepMat`, `JR.BinTree`, `CRR.BinTree`

Examples

```
# set option parameters
under = 105
strike = 95
rfr = 0.08
sigma = 0.2
maty = 0.5
yield = 0.03
life = 0.5

# estimate option price using Leinsen and Reimer Binomial Tree
lr = LR.BinTree(Nsteps=10
                    , under
                    , strike
                    , rfr
                    , sigma
                    , maty
                    , yield
                    , life
                    , ret.steps=TRUE)
lr$Values
# ... confront results with B&S method
BS.price(under, strike, rfr, sigma, maty, yield)
# get step matrix
lr = LR.BinTree(Nsteps=10
                    , under
                    , strike
                    , rfr
                    , sigma
                    , maty
                    , yield
                    , life
                    , ret.steps=TRUE)
lr$Price_Path
```

macd — *Moving Average Convergence / Divergence*

Description

Compute Moving Average Convergence / Divergence (Technical Analysis)

Usage

```
macd(X, fast.lag = 12, slow.lag = 26, signal.lag = 14, plot = TRUE, ...)
```

Arguments

`X`	X
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`signal.lag`	signal.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

mass — *Mass indicator*

Description

Compute Mass indicator (Technical Analysis)

Usage

```
mass(High, Low, Close , lag = 9, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

masscum — *Mass indicator cumulative*

Description

Compute Mass indicator cumulative (Technical Analysis)

Usage

```
mass.cum(High, Low, Close = NULL, lag = 9, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

mcf — *Auto-Correlation and Partial Auto-Correlation*

Description

Compute auto-correlation and partial auto-correlation function on a matrix

Usage

```
mcf(X, lag.max = 10, ci = 0.95, plot=TRUE, ...)
```

Arguments

X	Matrix of data series (one column per variable)
lag.max	Max lag to be computed by the cross correlation function (Default: 10)
ci	Confidence Interval (Default: 0.95)
plot	Logical. If TRUE, results are plotted.
...	Additional parameters accepted by the function plot.cross.ccf.

Value

An object of class "mcf". This is a list with two entries:

ACF	List of Auto-Correlation Functions (one for each column of X).
PACF	List of Partial Auto-Correlation Functions (one for each column of X).

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`cross.ccf`

Examples

```
# Dow Jones
DJ = get.fs("^DJI", SName = "DowJones"
    , from=as.Date("2008-06-01")
    , to=as.Date("2009-04-01")
    );
# Compute Returns
RDJ = Ret(DJ, na.rm = TRUE)

# Plot Autocorrelation Function and Partial ACF
mcf(RDJ, lag.max = 30)
# Using another theme
mcf(RDJ, lag.max = 30, theme = getTheme("vanilla"))
```

---

mcgind *McGinley Dynamic Indicator*

Description

Compute McGinley Dynamic Indicator (Technical Analysis)

Usage

```
mcgind(X, lag = 12, plot = FALSE, ...)
```

Arguments

`x`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`mclog` *Manage Console Logging*

Description

Set and retrieve the console logging status. Control whether logging info is printed to console.

Usage

```
setConsoleLogging(consoleLogging = TRUE
                  , env = getOption("RAdamant"))

getConsoleLogging(env = getOption("RAdamant"))
```

Arguments

`consoleLogging`	Logical. If TRUE, log information are also sent to console.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Value

Returns the current ConsoleLogging status.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Retrieve current debug level
getConsoleLogging();

# Enable logging
setDebugTraceLevel(1);
setDebugLevel(1);
# Enable Console Logging
setConsoleLogging(TRUE);
cplot(1:10)
```

---

mcosc *McClellan Oscillator*

Description

Compute McClellan Oscillator (Technical Analysis)

Usage

```
mcosc(X, fast.lag = 19, slow.lag = 39, hist.lag = 9, plot = TRUE, ...)
```

Arguments

`X`	X
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`hist.lag`	hist.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

mcplot *Muliple correlation plot*

Description

Multiple correlation plot

Usage

```
mcplot(X
     , hist.nclass = 10
     , theme.params = getCurrentTheme()
     , coLin = TRUE
     , main = ifelse(coLin, "Co-Linearity Analysis"
                          , "Multi-Correlation Analysis")
     , new.device = FALSE
     , ...
     )
```

Arguments

`X`	Matrix of data series (one column per variable).
`hist.nclass`	Number of bins used for computing histogram plot (Default: 10).
`theme.params`	RAdamant graphics theme.
`coLin`	Logical. If TRUE, Co-Linearity analysis is performed, otherwise Correlation analysis is assumed. See details.
`main`	The plot title
`new.device`	Logical. If TRUE, a new devide is opened.
`...`	Further arguments passed to chist.

Details

The parameter 'coLin' controls how correlation coefficients are displayed:

- coLin = TRUE: the higher the correlation (in absolute terms) the more the corresponding columns are collinear.
  The correlation coefficient is displayed with variable colors ranging from green (abs(rho) = 0) to red (abs(rho) = 1).
- coLin = FALSE: Colors are switched ranging from red (abs(rho) = 0) to green (abs(rho) = 1).

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`chist`

Examples

```
# Load sample time series data
data(ex_ptf);

# Compute Multi Collinearity Analysis
# (High correlation (abs(rho)) in red)
mcplot(ex_ptf[, c(2:5)]);

# Compute Multi Correlation Analysis
# (High correlation (abs(rho)) in green)
mcplot(ex_ptf[, c(2:5)]
    # Increase number of histogram bins
    , hist.nclass = 30
    # Specify correlation type analysis
    , coLin = FALSE
    # Use Normal distribution fitting for the histograms
    , density = "normal"
    );
```

`mcsi` *McClellan Summation Index*

Description

Compute McClellan Summation Index (Technical Analysis)

Usage

```
mcsi(matr, nr, nc, lag1, lag2, plot = FALSE, ...)
```

Arguments

`matr`	matr
`nr`	nr
`nc`	nc
`lag1`	lag1
`lag2`	lag2
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`mdbtlev` *Manage Debug Trace Level*

Description

Set and retrieve the level of function nesting for which logging is performed. Controls how much information is sent to the log about the execution of each function executed inside the call stack.

Usage

```
setDebugTraceLevel(level = 1, env = getOption("RAdamant"))
getDebugTraceLevel(env = getOption("RAdamant"))
```

Arguments

`level`	The level of nesting (level >= 1). See details.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Details

The amount of information sent to log depends on the debug trace level:

- level = 1: Only top level function calls are logged.
- level = 2: Top and second level function calls (function within a function) are logged.
- level = N: All functions in the call stack up to level N are logged.

Value

The current value of debug trace level.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Retrieve current debug level
getDebugTraceLevel();

# Enable logging to console
setConsoleLogging(TRUE);

# Set minimal level of trace debugging
setDebugTraceLevel(1);
cplot(1:10);

# Set high level of trace debugging
# (up to the 5-th level of inner function call)
setDebugTraceLevel(5);
cplot(1:10);
```

mdbuglev *Manage Debug Level*

Description

Set and retrieve the level of debugging. Control how much information is sent to the log about the execution of each function executed.

Usage

```
setDebugLevel(level = 1, env = getOption("RAdamant"))
getDebugLevel(env = getOption("RAdamant"))
```

Arguments

`level`	The level of debug required (level >= 0). See details.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Details

The amount of information sent to log depends on the debug level:

- level = 0: No information is sent to the log.
- level = 1: Information about main body and conditional executions.
- level = 2: Include information about first level inner loop.
- level = 3: Include information about second level inner loop (loop within loop).
- level = N: Include information about N-th level inner loop.

Value

The current level of debugging.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Retrieve current debug level
getDebugLevel();

# Set minimal level of debugging and traceback
setDebugLevel(1);
setDebugTraceLevel(1);
# Enable Console logging
setConsoleLogging(TRUE);

# Compute FFT on some random two-colums matrix.
x = FFT(matrix(cumsum(rnorm(256)), 128, 2), plot = FALSE)
# Prints nothing because FFT.default has no logging message
plot(x, shaded = FALSE)
```

```
# Increase Traceback level
setDebugTraceLevel(2);
# Now prints logging info for plot.FFT
plot(x, shaded = FALSE)

# Increase Debug level
setDebugLevel(2);
# Now prints additional logging info for plot.FFT
# (from code executed inside a loop)
plot(x, shaded = FALSE)
```

`means` *Geometric and Harmonic means*

Description

`gmean`: Compute the geometric mean for each column of X
`hmean`: Compute the harmonic mean for each column of X

Usage

```
gmean(X, ...)
hmean(X, ...)
```

Arguments

`X`	Matrix of data series (one column per variable)
`...`	Additional parameters accepted by the function sum (i.e. na.rm)

Value

Matrix of harmonic / geometric means

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`mfind` *Money flow indicator*

Description

Compute Money flow indicator (Technical Analysis)

Usage

```
Mflow.ind(Close, High, Low, Volume, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Volume`	Vector. Asset traded Volume.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`mflow` *Money flow*

Description

Compute Money flow (Technical Analysis)

Usage

```
Mflow(Close, High, Low, Volume, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Volume`	Vector. Asset traded Volume.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

mfratio — *Money flow ratio*

Description

Compute Money flow ratio (Technical Analysis)

Usage

```
Mflow.ratio(Close, High, Low, Volume, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Volume`	Vector. Asset traded Volume.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

minmaxs — *Mini/Max Scale*

Description

Compute minimum / maximum scale of a vector

Usage

```
Minmaxscal(x, tmin = 0, tmax = 1)
```

Arguments

`x`	x
`tmin`	tmin
`tmax`	tmax

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

mlbsize *Manage Log Buffer Size*

Description

Set and retrieve the size of the current log buffer.

Usage

```
setLogBufferSize(size = 10000, env = getOption("RAdamant"), ...)
getLogBufferSize(env = getOption("RAdamant"))
```

Arguments

`size`	The capacity (number of records) of the log buffer.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).
`...`	Additional parameters passed to flushLogBuffer.

Value

Returns the size of the current log buffer.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Retrieve current buffer size
getLogBufferSize();

# Set the size of the log buffer to 10 records
# This will also force a flush to file of the current content.
setLogBufferSize(10);
```

mlogfile — *Manage Logging Filename*

Description

Set and retrieve the full filename and location of the current log file.

Usage

```
setLogFile(logfile = NULL, env = getOption("RAdamant"))
getLogFile(env = getOption("RAdamant"))
```

Arguments

`logfile`	String. The full path to the log file.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Value

The full filename and location of the current log file.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Retrieve current log file
getLogFile();

# Set log file
setLogFile("path-to-logfile");
```

mlogwarn — *Manage log warnings*

Description

Set and retrieve the LogWarning status. Not all functions support this feature.

Usage

```
setLogWarning(showWarning = TRUE, env = getOption("RAdamant"))
getLogWarning(env = getOption("RAdamant"))
```

Arguments

`showWarning`	Logical. If TRUE, a warning is generated if the log buffer is full and no logfile is available.
`env`	The environment where the info is stored (Default: getOption("RAdamant")).

Value

The current value of LogWarning (TRUE/FALSE).

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Retrieve current status
getLogWarning();

# Set the size of the log buffer to 10 records
setLogBufferSize(10);
# Set an invalid entry for the log file
setLogFile(logfile = NULL);

# Enable logging
setDebugLevel(1)
# Enable Log Warning
setLogWarning(TRUE);
cplot(1:10) # Prints a warning

# Disable Log Warning
setLogWarning(FALSE);
cplot(1:10) # No warning

# Restore RAdamant package options
# .First.lib()
```

---

mma *Modified EMA*

Description

Compute multiple Modified EMA on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
mma(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
MMA is a EMA with smoothing factor: lambda = 1/win.size.

Value

A object of class 'ma' with attributes type = "MMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
mma(x, 15)
# compute moving average with multiple lags
mma(x, c(5, 10, 30, 50))

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
mma(x, 30, plot = TRUE)
# multiple lags
mma(x, c(5, 10, 30, 50), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
mma(ex_fs, c(5, 10, 30, 50), plot=TRUE)

## End(Not run)
```

`mndma` *Modified N-Day Moving Averages*

Description

Computes multiple Modified N-Day Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
mndma(X, win.size = 50, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable)
`win.size`	Vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function sma

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'Movav' with attributes type = "MNDMA" and 'win.size' as from the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`sma`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
mndma(x, 50)
# compute moving average with multiple lags
mndma(x, c(40,50,60))

## Not run:
# refine results of moving average
```

```
setCurrentTheme(2)
# single lag
mndma(x, 50, plot = TRUE)
# multiple lags
mndma(x, c(30,40,50), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
mndma(ex_fs, 25, plot=TRUE)
# multiple lags
mndma(ex_fs, seq(5,25,5), plot=TRUE)

## End(Not run)
```

`mom` *Momentum oscillator*

Description

Compute Momentum oscillator (Technical Analysis)

Usage

```
mom(X, lag = 5, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`moments` *Main Moments*

Description

Calculate sample moments on each columns of X and sample moments of a probabilty density function.

Usage

```
moments(X)
SampMom(P, X, moms = 1:2)
```

Arguments

`X`	Matrix of data series (one column per variable)
`P`	Vector of probabilities
`moms`	Moments to calculate; default first and second and moment

Value

Matrix of moments

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`JB.test`, `skew`, `kurt`

`movapply` *Moving Apply function*

Description

Applies a given function to a sliding window of the input data

Usage

```
movApply(X
    , win.size = 1
    , padding = NA
    , rm.transient = FALSE
    , func = NULL
    , ...
    )
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	Vector of data window sizes that will be passed to the given function "func" (Default: 1).
`padding`	Padding value to fill transient of result (output data rows from 1 to win.size-1). Default: NA.
`rm.transient`	transient: Logical. If TRUE, transient is removed, otherwise funct is applied to the transient (Default: FALSE).
`func`	Function to be run.
`...`	Additional parameters accepted by the function func.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A matrix of size NROW(X) by NCOL(X)*length(win.size).
The function specified by 'func' is applied to each sliding window SWi (given by win.size[i]) and each column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

movav *Generic Multiple) Moving Average*

Description

Generic Multiple Moving Average (MA filter). Compute multiple FIR filtering on each column of the input data

Usage

```
Movav(X, ...)
## Default S3 method:
Movav(X, win.size = NULL,
func = NULL, padding = 0,
rm.transient = TRUE, normalize.weights = FALSE,
type = "MA", desc = "Moving Average",
plot= FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of lengths of the FIR filters to be applied on the data X. (Default: NULL).
`func`	function accepting an integer N and returning an N-long set of filter coefficients.
`padding`	value to replace leading lagged values.
`rm.transient`	remove initial lagged window.
`normalize.weights`	Normalise weights for weighted moving averages.
`type`	Charachter attribute attached to the result (Default: "MA").
`desc`	desc
`plot`	Logical. Return plot.
`...`	Further arguments to or from other methods

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'Movav' with attributes 'type' and 'win.size' as given by the corresponding input parameters:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`movfunc` *Moving Base Functions*

Description

Applies the function "Max", "Min", "Standard Deviation" or "Variance" to a sliding window of the input data

Usage

```
movMax(X, win.size = 1, ...)
movMin(X, win.size = 1, ...)
movSd(X, win.size = 1, ...)
movVar(X, win.size = 1, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	Vector of data window sizes that will be used for the calculations (Default: 1).
`...`	Additional parameters accepted by the function movApply

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A matrix of size NROW(X) by NCOL(X)*length(win.size). max is applied to each sliding window SWi (given by win.size[i]) and

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`movApply`

`mqt` *Multiple Quantiles from Students T distribution*

Description

Compute quantiles from Students T distribution for multiple values of degrees of freedom

Usage

```
mqt(p, df, ...)
```

Arguments

`p`	Vector of probabilities (Default: 0.05)
`df`	Vector of degrees of freedom
`...`	Further arguments to and from other methods

Value

A matrix length(p) by length(df) of computed quantiles

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Multiple quantiles
mqt(p = seq(0.01, 0.05, by = 0.01), df = c(2, 3, 4))
```

`mreg` *Multiple Regression*

Description

Perform a linear regression for each column Yi of Y, using the columns of X as predictors.
Linear Models or Generalised Linear Models can be used for the regression.
Stepwise regression is also possible, and a constraint to limit the number of selected columns can be specified.

Usage

```
mreg(Y
    , X
    , xlabels = NULL
    , backtest = 0
    , stress.idx = c()
    , type = "simple" # simple | stepwise
    , model = "lm" # lm | glm
    , ci = 0.95
```

```
, max.vars = NCOL(X)
, intercept = TRUE
, family = gaussian
, weights = NULL
, scope = NULL
, trace = FALSE
, plot = TRUE
, theme.params = getCurrentTheme()
, overrides = NULL
, ...
)
```

Arguments

`Y`	Matrix of data series - Dependent variables (one column per variable).
`X`	Matrix of data series - Independent variables (one column per variable).
`xlabels`	Labels for the x-axis.
`backtest`	Vector of NCOL(Y) integers. Each entry sets the number of data points to be used for backtesting the respective i-th model for Yi. If greater than 0, an additional regression is run on a reduced sample given by the first 1:backtest data points (development sample) , hence the performance of the model is evaluated against the excluded data points (validation sample). Parameter is recycled to the number of columns of Y.
`stress.idx`	Vector of indices identifying the data points that represent a 'stress' regime from the base case scenario. If provided, an extended linear model is computed, where a different regression coefficient for each predictor is estimated to model the regime change.
`type`	Vector of NCOL(Y) entries, each from one of the following: • "simple": All columns of X are used in the regression of Yi. • "stepwise": Stepwise regression is performed to compute the best model with no more than max.vars predictors. Parameter is recycled to the number of columns of Y.
`model`	Vector of NCOL(Y) entries, each from one of the following: • "lm": Linear Model (lm) is used for the regression of Yi. • "glm": Generalised Linear Model (glm) is used for the regression. Parameter is recycled to the number of columns of Y.
`ci`	Confidence Intervals on the model estimation.
`max.vars`	Vector of NCOL(Y) integers. Each entry allows to put a constraint on the max number of predictors to enter the i-th model, when type = "stepwise". Parameter is recycled to the number of columns of Y.
`intercept`	Logical vector with NCOL(Y) entries. If TRUE, intercept term is included in the regression. Parameter is recycled to the number of columns of Y.
`family`	Vector of NCOL(Y) family names or list with NCOL(Y) entries (a family function per entry). Each entry sets the family used by the glm model. Parameter is recycled to the number of columns of Y.
`weights`	Weights to be used for weighted lm/glm. Useful when Yi is a probability measure, to convert the probabilities in absolut count terms, so that binomial/logit family can be used.

`plot`	Logical. If TRUE results are plotted for each model.
`scope`	Defines the range of models examined in the stepwise search. See `step` for details. By default all columns of X are in scope.
`trace`	Controls the debug trace level for thw stepwise regression.
`theme.params`	Plotting thems.
`overrides`	Overrides parameters.
`...`	Additional parameters passed to the lm/glm function.

Value

An object of class 'mreg'. This is a list of NCOL(Y) elements of class 'reg'.
Each 'reg' object is a list with the following components:

`lm`	the regression model, as returned by lm/glm.
`summary`	a summary of the model.
`formula`	the model formula.
`weights`	the weights used on the regression.
`coeff.weights`	the percentage weights of the regression coefficients.
`target`	the dependent variable Yi.
`response`	the predicted response (and confidence intervals) on the scale of Yi. Matrix of columns [fit, lwr, upr].
`residuals`	the residuals on the scale of Yi.
`linear.target`	the dependent variable on the link scale (i.e. logit(Yi)).
`linear.predictors`	the predicted response (and confidence intervals) on the link scale. Matrix of columns [fit, lwr, upr].
`linear.residuals`	the residuals on the link scale.
`ci`	the confidence interval level.
`model.type`	the type of model used. One of 'lm' or 'glm'.
`family`	the family used for the glm model
`regression.type`	the type of regression computed. One of 'simple' or 'stepwise'.
`fcast`	when backtest > 0, this is the forecasted response (and confidence intervals) on the scale of Yi, computed using the validation sample. This is NULL if backtest = 0.
`fcast.residuals`	the forecast residuals (on the scale of Yi) when backtest > 0, NULL otherwise.
`stress.idx`	the input argument used to identify a stress regime.
`backtest`	the input argument used to backtest the data.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

glm, lm, step, plot.mreg, get.lm.weights, dropn.

Examples

```
# Generate some random data
N = 50;
sigma = 0.1;
X1 = cumsum(rnorm(N));
X2 = rnorm(N);
X3 = cumsum(rnorm(N));
X4 = rnorm(N);

# Define a linear model
Y1 = 1.5 + X1 + 2*X3 + rnorm(N, sd = sigma);
# Define a logit model
Y2 = inv.logit(-2.2 + 0.3*X2 - 0.2*X4 + rnorm(N, sd = sigma));

# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2)
          , X = cbind(X1, X2, X3, X4)
          # Stepwise regression
          , type = "stepwise"
          # lm on Y1 and glm on Y2
          , mode = c("lm", "glm")
          # Set the family.
          # It is recycled but family is only used for glm
          , family = "binomial"
          # Constrain the maximum number of variables
          # that can enter the regression
          , max.vars = c(3, 2)
          # Use another theme
          , theme.params = getTheme(2)
          );
```

---

msort *Sort matrix*

Description

Sort each column of the input matrix X independently

Usage

```
SORT(X, decreasing = FALSE, ...)
```

Arguments

X	Input matrix.
decreasing	Logical. Decreasing order.
...	Further arguments to or from other methods.

Value

A matrix with the same dimensions as the original input X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
data(ex_fs)
x = ex_fs[1:20, 1:3]
SORT(x, decreasing = FALSE)
```

---

mtacf *Cool.Acf methods*

Description

Plot and Print methods for class 'cool.acf'

Usage

```
## S3 method for class 'cool.acf'
print(x, ...)

## S3 method for class 'cool.acf'
plot(x
     , theme.params = getCurrentTheme()
     , xtitle = "Lag"
     , ytitle = expression(rho)
     , overrides = list(...)
     , ...
     )
```

Arguments

`x`	Instance of class 'cool.acf'
`theme.params`	Theme parameters (Default: getCurrentTheme())
`xtitle`	Title for the x-axis (Default: "Lag")
`ytitle`	Title for the y-axis (Default: expression(rho))
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...))
`...`	Alternative way to quickly override theme parameters

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Run Multi correlation analysis
X = mcf(rnorm(30), plot = FALSE);
# Extract cool.acf component
Y = X$ACF[[1]]
class(Y)
# Plot Autocorrelation function
plot(Y)
```

`mtccf` *Cross.ccf functions*

Description

Methods for class 'cross.ccf'

Usage

```
## S3 method for class 'cross.ccf'
print(x, ...)

## S3 method for class 'cross.ccf'
plot(x
    , theme.params = getCurrentTheme()
    , xtitle = "Lag"
    , ytitle = expression(rho)
    , overrides = list(...)
    , ...
    )
```

Arguments

`x`	Instance of class 'cross.ccf'
`theme.params`	Theme parameters (Default: getCurrentTheme())
`xtitle`	Title for the x-axis (Default: "Lag")
`ytitle`	Title for the y-axis (Default: expression(rho))
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...))
`...`	Alternative way to quickly override theme parameters.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Generate two random integrated series
N = 100
X = matrix(rnorm(N), nrow = N/2, ncol=2);
# Create two series as a linear combination of X plus noise
Y = X
# Perform Cross Correlation Analysis
Z = cross.ccf(Y, X, plot = FALSE)
plot(Z)
```

`mtmcf` *Multi-Correlation Function methods*

Description

Plot and Print method for class 'mcf'

Usage

```
## S3 method for class 'mcf'
print(x, ...)

## S3 method for class 'mcf'
plot(x
    , theme.params = getCurrentTheme()
    , xtitle = "Lag"
    , ytitle = expression(rho)
    , overrides = list(...)
    , ...)
```

Arguments

`x`	Instance of class 'mcf'
`theme.params`	Theme parameters (Default: getCurrentTheme())
`xtitle`	Title for the x-axis (Default: "Lag")
`ytitle`	Title for the y-axis (Default: expression(rho))
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...))
`...`	Alternative way to quickly override theme parameters

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`cplot`

Examples

```
## Not run:
# Dow Jones
DJ = get.fs("^DJI"
            , SName = "DowJones"
            , from = as.Date("2008-06-01")
            , to = as.Date("2009-04-01")
            );
# Compute Returns
RDJ = Ret(DJ, na.rm = TRUE)

# Compute Multi Correlation Function
res = mcf(RDJ, lag.max = 30, plot = FALSE)
# Plot Autocorrelation Function and Partial ACF
plot(res)
# Using another theme
plot(res, theme = getTheme("vanilla"))

## End(Not run)
```

---

mtoscil — *Plot function for Oscillators*

Description

Plot and Print method for Oscillators (Technical Analysis)

Usage

```
## S3 method for class 'oscil'
print(x, digits = 5, ...)

## S3 method for class 'oscil'
plot(x, Y = NULL, main = "",
show.trsh = NULL, xlabels = rownames(Y),
theme.params =getTheme(1), overrides = NULL, ...)
```

Arguments

`x`	x
`Y`	Y
`main`	main
`show.trsh`	show treshold
`xlabels`	xlabels
`theme.params`	them.params
`overrides`	overrides
`digits`	digits
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

mtreg — *Summary methods for (Multi)-Regression object*

Description

Summary method for classes 'reg' and 'mreg'.

Usage

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

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

Arguments

object	Instance of class 'reg'/'mreg'.
...	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);

# Run Multi-Regression
mod = mreg(Y, X = cbind(X1, X2), plot = FALSE);
# Print Summary
summary(mod)
```

`mtunivar` *Methods for univariate analysis*

Description

Print, Plot and Summary methods for class 'univar'

Usage

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

## S3 method for class 'univar'
print(x, ...)

## S3 method for class 'univar'
plot(x
, theme.params = getCurrentTheme()
, overrides = list(...)
, ...
)
```

Arguments

`x, object`	Instance of class 'univar'
`theme.params`	params: Theme parameters (Default: getCurrentTheme())
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...))
`...`	Alternative way to quickly override theme parameters.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`univar`, `cplot`

Examples

```
# Load sample time series data
data(ex_ptf)
# Define the dependent variable
Y = ex_ptf[, 1, drop = FALSE];
# Define the independent variables
X = ex_ptf[, -1];
# Define x-axis labels
time.labels = paste("t[", 1:length(Y), "]", sep = "")
# Univar analysis
res = univar(Y, X, plot = FALSE);
plot(res
    , xlabels = parse(text = time.labels)
```

```
# Remove x-labels rotation
, xlab.srt = 0
# Set more space between x-labels and the x-axis line
# (10% of diff(par("usr")[3:4]))
, xlab.offset = 0.1
# Set more space between x-title and the x-axis line
# (20% of diff(par("usr")[3:4]))
, xtitle.offset = 0.2
# Only 4 tickmarks on the y-axis
, y.ticks = 4
)
```

`namutil` *Get column and row names*

Description

Retrieve column / row names from a matrix.

Usage

```
get.col.names(X, default = "X")
get.row.names(X, default = "")
```

Arguments

`X`	Input matrix.
`default`	LOGICAL vector. Each entry determines the sort direction of the respective column of X. Recycled if necessary. (Default: FALSE).

Details

Sequences are treated as one column matrices.
Default names are given if input has missing names.

Value

A character sequence containing the column names of X, or a default set of names if X has no column names

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

newsimp *News impact curve*

Description

Compute News impact curve for Garch models

Usage

```
newsimp(x, ...)
## S3 method for class 'Garch'
newsimp(x, plot = TRUE, ...)
## Default S3 method:
newsimp(x
, theta
, order
, type=c("garch","mgarch", "egarch","tgarch")
, plot=FALSE
, ...)
```

Arguments

x	A vector of innvations (x axis of the plot) or an object of class "Garch".
theta	Vector of Garch model parameters.
order	Vector of integers. Arch and Garch parameters order. (Default: 1,1)
type	Type of Garch to be estimated: "garch", "mgarch", "tgarch", "egarch". (Default: "garch").
plot	Logical. If TRUE plot of the NIC is returned.
...	Further arguments to or from other methods

Value

The function returns the NIC curve plus a matrix containing: Sigma values (y axis) and Innovations (x axis).
The plot is made by the cplot function, for more information about the graphical parameters take a look here `cplot`.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`Garch`, `cplot`

Examples

```
# load example time series
data(ex_ts)
x = ex_ts

# Symmetric NIC - GARCH example
gg1 = Garch(x, order = c(2,1), type="garch", prob="g")
newsimp(gg1)

# Asymmetric NIC - EGARCH and TGARCH example
gg2 = Garch(x, type="egarch", prob="g")
newsimp(x=gg2)
gg3 = Garch(x, type="tgarch")
newsimp(x=gg3)
```

---

normfit *Fit Normal Distribution*

Description

Fit a Normal distribution on the input data.

Usage

```
norm.fit(x, n = 200, ...)
```

Arguments

x	the data on which the Normal distribution is fitted.
n	the number of data points on with the estimated distribution is evaluated
...	Further arguments to or from other methods.

Value

A list with thefollowing elements:

mi	The estimated mean.
sigma	The estimated standard deviation.
x	The quantiles where the Normal distribution is evaluated.
y	The value of the Normal distribution at the points given by x.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate some random data from a Normal distribution.
x = rnorm(100);

# Fit distribution
res = norm.fit(x, n = 30);
res
```

normlike — *Normal Distribution - Log Likelihood function*

Description

Normal Distribution - Log Likelihood function

Usage

```
norm.like(parms, X, ...)
```

Arguments

`parms`	parms
`X`	X
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

objgarch — *Garch objects*

Description

Extract objects from Garch model (class "Garch")

Usage

```
## S3 method for class 'Garch'
coef(object, names=TRUE, ...)
## S3 method for class 'Garch'
logLik(object, ...)
## S3 method for class 'Garch'
vcov(object, ...)
```

Arguments

`object`	An object of class "Garch"
`names`	Return names
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`obv` *On Balance Volume oscillator*

Description

Compute On Balance Volume oscillator (Technical Analysis)

Usage

```
Obv(Close, Volume)
```

Arguments

`Close`	Vector. Close price.
`Volume`	Vector. Asset traded Volume.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`oscil` *Oscillator default method*

Description

Compute Oscillator (Technical Analysis)

Usage

```
oscil(X, ...)
## Default S3 method:
oscil(X, Y, pc = FALSE, type = "oscil", ...)
```

Arguments

`X`	X
`Y`	Y
`pc`	pc
`type`	type
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`pchan` *Price channel*

Description

Compute Price channel (Technical Analysis)

Usage

```
Pchan(CLose
     , High
     , Low
     , lag = 20
     , na.rm = TRUE
     , plot = FALSE
     , ...
     )
```

Arguments

`CLose`	CLose
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`na.rm`	na.rm
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

pdfhit *Density of Hitting probability*

Description

Probability density function for first hitting barriers

Usage

```
PDFHit(t
, B = 0
, S0 = 0
, mi
, sigma
, cumul = FALSE
, plot = FALSE
, ...
)
```

Arguments

t	Vector. Time period.
B	Numeric. Barrier value.
S0	Initial level of the process.
mi	Drift value.
sigma	Volatility value.
cumul	Logical. If TRUE cumulative probability distribution is computed.
plot	Logical. If TRUE plot is returned.
...	Further arguments to or from othermethods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

FirstHit, ProbHit

Examples

```
## Show density function for different values of "sigma"
sigma = c(0.02, 0.06, 0.1, 0.15, 0.2, 0.25, 0.3)
# simulate PDFHit for each value of sigma
pdf = matrix(NA, 100, length(sigma))
colnames(pdf) = paste("Sigma=",sigma)
for(s in 1:length(sigma))
    pdf[,s] = PDFHit(t=1:100
                    , B=0
                    , S0=1
                    , mi=0
```

```
                        , sigma = sigma[s]
                        , cumul=FALSE
                        , plot=FALSE
                        )
# plot different functions
cplot(pdf, main="Density of Hitting probability")
```

---

perf   *Performance indicator*

Description

Compute Performance indicator (Technical Analysis)

Usage

```
Perf(X, ini.per = 1, cut = TRUE, plot = FALSE, ...)
```

Arguments

`X`	X
`ini.per`	ini.per
`cut`	cut
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

pfe   *Polarized fractal efficiency*

Description

Compute Polarized fractal efficiency (Technical Analysis)

Usage

```
pfe(X, lag = 9, corr_fact = 200, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`corr_fact`	corr_fact
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`pgarch` *Print Garch*

Description

Print function for Garch model

Usage

```
## S3 method for class 'Garch'
print(x, digits = 5, ...)
```

Arguments

`x`	x
`digits`	digits
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`pgev` *Generalised Extreme Value (GEV) - Probability function*

Description

Generalised Extreme Value (GEV) - Probability function

Usage

```
pgev(X, mu = 0, xi = 0.1, sigma = 1)
```

Arguments

`X`	X
`mu`	mu
`xi`	xi
`sigma`	sigma

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`pgpd` *Generalised Pareto Distribution (GPD) - Probability function*

Description

Generalised Pareto Distribution (GPD) - Probability function

Usage

```
pgpd(Q, xi = 0.1, sigma = 1, trsh = 0)
```

Arguments

`Q`	Q
`xi`	xi
`sigma`	sigma
`trsh`	trsh

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

plikeci — *Likelihood confidence intervals calculation*

Description

General function for profile likelihood confidence intervals calculation

Usage

```
plike.ci(ML.init = c()
       , flike = NULL
       , alpha = 0.01
       , df = NULL
       , frange = list()
       , par.names = NULL
       , ...)
```

Arguments

`ML.init`	ML.init
`flike`	flike
`alpha`	alpha
`df`	df
`frange`	frange
`par.names`	par.names
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

plikecnt — *Likelihood joint confidence intervals contour*

Description

General function for profile likelihood joint confidence intervals contour

Usage

```
plike.contour(ML.init = c(), flike = NULL,
alpha = 0.01, df = NULL, frange = list(),
par.names = NULL, grid.size = 100, ...)
```

Arguments

`ML.init`	ML.init
`flike`	flike
`alpha`	alpha
`df`	df
`frange`	frange
`par.names`	par.names
`grid.size`	grid.size
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`plikerng` *Range grid for contour calculation*

Description

General range grid for contour calculation

Usage

```
plike.range(ML.init = c()
, flike = NULL
, alpha = 0.01
, df = NULL
, frange = list()
, par.names = NULL
, grid.size = 100
, max.iter = 100
, tol = 10^-5
, ...
)
```

Arguments

`ML.init`	ML.init
`flike`	flike
`alpha`	alpha
`df`	df
`frange`	frange
`par.names`	par.names

`grid.size`	grid.size
`max.iter`	max.iter
`tol`	tol
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`plotfft` *Customised Fast Fourier Transform - Plotting*

Description

Plot function for class 'FFT'. Plots Modulus and Phase for each column of the FFT object x

Usage

```
## S3 method for class 'FFT'
plot(x
    , theme.params = getCurrentTheme()
    , overrides = list(...)
    , shaded = TRUE
    , show.periodicity = FALSE
    , show.legend = FALSE
    , zoom = 100
    , semilog = FALSE
    , new.device = FALSE
    , ...
    )
```

Arguments

`x`	Instance of class 'FFT'.
`theme.params`	RAdamant graphics theme (Default: getCurrentTheme()).
`overrides`	List of parameters to override the theme. Only parameters that match those defined by the theme are overridden (Default: list(...)).
`shaded`	Logical. If TRUE, the modulus of x is shaded.
`show.periodicity`	Logical. If TRUE, Periods (1/frequencies) are showed instead of frequencies on the x-axis (Default: FALSE).
`show.legend`	Logical. If TRUE, legend is added to the plot (Default: FALSE).
`zoom`	Zoom
`semilog`	Logical. If TRUE, the modulus of the FFT is shown on a dB scale.

`new.device`	Logical. If TRUE, a new plotting device is opened.
`...`	Additional parameters passed to the cplot function. Also used to quickly specify theme overrides.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

cplot.

Examples

```
# Load sample financial series data
data(ex_fs)

# Frequency Analysis
Xf = FFT(ex_fs, plot = FALSE)

# Plot full spectrum
plot(Xf)

# Plot half spectrum
plot(Xf
    # Plot half spectrum (right side)
    , half = TRUE
    # Use Blackman window
    , window = blackman
    # Remove area shading
    , shaded = FALSE)

# Show periodicity instead of frequency, and use hamming window
plot(Xf
    , half = TRUE
    , window = hamming
    , show.periodicity = TRUE)

# Plot with other options
plot(Xf
    # Half spectrum
    , half = TRUE
    # Kaiser Window
    , window = kaiser
    , # Show cycles instead of frequencies
    , show.periodicity = TRUE
    # Zoom in to show only 10% of the half spectrum
    , zoom = 10
    # Use semilog axis (decibel)
    , semilog = TRUE
    )
```

`plotfs` *Plot fs data*

Description

Plot method for Financial Series (fs) object.

Usage

```
## S3 method for class 'fs'
plot(x, ...)
```

Arguments

`x`	Instance of class 'fs'
`...`	Additional parameters passed to fin.plot function.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`fin.plot`.

Examples

```
# Load sample financial series data
data(ex_fs)
# Plot the data
plot(ex_fs)
# Change the style and color of the bottom chart
plot(ex_fs, overrides2 = list(type = "l", col = "grey"))
```

`plotkit` *Plotting Tools*

Description

Utility functions used for Plotting

Usage

```
draw.grid(X
     , base = NULL
     , theme.params = getCurrentTheme()
     , method = c("equispaced", "sampling")
     )

draw.legend(legend = ""
     , theme.params = getCurrentTheme()
     , overrides = list(...)
     , ...
     )

draw.x.axis(X
     , base = NULL
     , xlabels = NULL
     , theme.params = getCurrentTheme()
     , show.labels = TRUE
     , show.ticks = TRUE
     , ...
     )

draw.x.title(xtitle = "", theme.params = getCurrentTheme())

draw.y.axis(X
     , ylabels = NULL
     , theme.params = getCurrentTheme()
     , side = 1
     , show.labels = TRUE
     , show.ticks = TRUE
     , ...
     )

draw.y.title(ytitle = "", theme.params = getCurrentTheme(), side = 1)
```

Arguments

`X`	Matrix of data series being plotted (y-values). One column per series.
`base`	Corresponding x-values (common to all series) associated to the entries of X. If NULL, then base = 1:NROW(X).
`theme.params`	A valid RAdamant Theme. See setThemeAttr for details. (Default: getCurrentTheme())
`overrides`	List of parameters used to override the theme. Only parameters that match those defined by the theme are overridden (Default: list(...))
`legend`	Vector of legend texts
`xlabels`	Labels for the x-axis
`ylabels`	Labels for the y-axis
`xtitle`	Title for the x-axis

`ytitle`	Title for the y-axis
`show.labels`	Logical. If TRUE, labels are showed.
`show.ticks`	Logical. If TRUE, tickmarks are showed.
`side`	The side (1 = left, 2 = right) where the y-axis labels and title are plotted.
`method`	Controls how the x-coordinates of the grid vertical lines are computed. If method = "equispaced", N = getThemeAttr("x.ticks", exact = TRUE) points between min(base) and max(base) are computed. If method = "sampling", the N lines are drawn at the points given by base[seq(1, length(base), len = N)].
`...`	Further arguments to or from other methods.

Details

These are utility funtions used as building blocks for high level plotting with cplot. Most of the behaviour is controlled by the theme options.
For details on the available options, see `setThemeAttr`.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`cplot`

`plotmov` *Plot Moving Average*

Description

Plot method for object of class 'Movav' (Moving Average)

Usage

```
## S3 method for class 'Movav'
plot(x, fs = NULL, main = attr(x, "desc"), ...)
```

Arguments

`x`	instance of class 'Movav'
`fs`	Matrix containing the original data series (one column per variable). For financial time series (class = 'fs'), only 'Close' column is processed.
`main`	Main title of the plot
`...`	Additional parameters accepted by the functions cplot and fin.plot

Details

If the original data series is an instance of class 'fs', then the plot will have two panels:

- plot of fs and x on the top;
- histogram of the Volume data of the financial series X.

Value

VOID

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

cplot

Examples

```
# Compute Exponential Moving Average and plot results
x = ema(rnorm(100), 10)

# Plot Multiple Moving Averages together using "" plotting class
plot(x)

## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:1000,2, drop=FALSE])
# set RAdamant theme (1 - Finance or 2 - Vanilla)
setCurrentTheme(1)
plot.Movav(cbind(kama(x)
                ,frama(x)
                ,ema(x, 10)
                ,gdema(x, 10)
                ,zlma(x, 10)
                )
           , x
           )

# plot multiple moving average results from an object of class "fs"
data(ex_fs)
class(ex_fs)
x = ex_fs
# set RAdamant theme (1 - Finance or 2 - Vanilla)
setCurrentTheme(2)
plot.Movav(cbind(kama(x)
                ,frama(x)
                ,ema(x, 10)
                ,gdema(x, 10)
                ,zlma(x, 10)
                )
           , x
           )
```

plotmreg — *Plot (Multi)-Regression object*

Description

Plot method for classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'mreg'
plot(x, ...)

## S3 method for class 'reg'
plot(x
, mode = c("response", "link")
, title = ifelse(x$model.type == "lm"
, "LS Regression"
, "GLM Regression"
)
, theme.params = getCurrentTheme()
, overrides = list(...)
, ...
)
```

Arguments

`x`	Instance of class 'reg'/'mreg'.
`mode`	One of 'response' or 'link'. Controls on which scale results are plotted. See `mreg` for details.
`title`	The plot title
`theme.params`	RAdamant graphics theme.
`overrides`	List of parameters to override the theme.
`...`	Additional arguments passed to cplot.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`mreg`, `cplot`.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);

# Run Multi-Regression
mod = mreg(Y, X = cbind(X1, X2), plot = FALSE);
plot(mod, theme.params = getTheme(2), xlab.srt = 0)
```

---

plotpvar *Plot VAR Predictions*

Description

Plot method for classes 'predVecAr'.

Usage

```
## S3 method for class 'predVecAr'
plot(x
     , main = "VAR Forecast"
     , xlabels = NULL
     , legend = NULL
     , theme.params = getCurrentTheme()
     , shaded = FALSE
     , ...
     )
```

Arguments

x	Instance of class 'predVecAr'.
main	The plot title
xlabels	Labels for x-axis ticks.
legend	Legend text.
theme.params	RAdamant graphics theme.
shaded	Shaded plot.
...	Additional arguments passed to cplot.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

VecAr, predict.VecAr, cplot.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Run 5-step ahead standard prediction
pred = predict(mod, steps = 5, plot = FALSE);
# Plot prediction
plot(pred, shaded = TRUE, shade.density = 50, shade.angle = 30)
```

---

plotret — *Plot Returns*

Description

Plot method for class "ret"

Usage

```
## S3 method for class 'ret'
plot(x
     , style = c("line"
     , "bar")
     , xlabels = rownames(x)
     , theme.params = getCurrentTheme()
     , ...
     )
```

Arguments

`x`	an objekt of class "ret"
`style`	plot style, "line" plot or "bar" plot
`xlabels`	xlabels
`theme.params`	theme.params
`...`	Further arguments to or from other methods

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

Ret

Examples

```
# load an example dataset containing financial daily prices
data(ex_fs)
x = ex_fs[ ,1:4]

# calculation and plot for single series
Ret(x[,1]
    , lag = 5
    , plot = TRUE
    , mode = "selected"
    , style="bar"
    , main="Returns - 5 Lags"
    )

# calculation and plot for multiple series
par(mfrow = c(2,2))
Ret(x
    , lag = 5
    , mode = "selected"
    , plot = TRUE
    , style = "bar"
    , main = "Returns - 5 Lags"
    )
```

---

plotroi — *Plot Return on Investment objects*

Description

Plot method for class 'roi'.

Usage

```
## S3 method for class 'roi'
plot(x
     , main = "Historical Return on Investment"
     , xtitle = "Lag"
     , ...
     )
```

Arguments

x	Instance of class 'roi'.
main	Title for the plot.
xtitle	The title for the x-axis.
...	Additional parameters passed to the cplot function.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

cplot.

Examples

```
# Load sample financial series data
data(ex_fs)

# Analyse the performance of the returns (Close data)
# Plot results on returns up to 200 days
plot(hroi(ex_fs
    , lag = 200
    , log = FALSE)
    , xlab.srt = 0
    )

# Analyse the performance of the returns (All data)
# Plot results on returns up to 200 days
plot(hroi(ex_fs[,]
    , lag = 200
    , log = FALSE)
    , xlab.srt = 0
    )
```

---

plotsme — *Plot Sample Mean Excess class*

Description

Plotting function for Sample Mean Excess class

Usage

```
## S3 method for class 'sme'
plot(x
    , main = attr(x
    , "desc")
    , xtitle = get.col.names(attr(x, "data"))
    , ...
    )
```

Arguments

`x`	OBJECT of class "sme".
`main`	main
`xtitle`	xtitle
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`plotspec` *Spectrogram Plotting*

Description

Plot method for class 'specgram'.

Usage

```
## S3 method for class 'specgram'
plot(x
     , show.periodicity = FALSE
     , theme.params = getCurrentTheme()
     , xtitle = "Time"
     , ytitle = ifelse(show.periodicity
                       , "Periodicity"
                       , "Frequency"
                       )
     , plot3d = FALSE
     , overrides = list(...)
     , ...
     )
```

Arguments

`x`	Instance of class 'specgram'
`show.periodicity`	Logical. If TRUE, Periods (1/frequencies) are showed instead of frequencies on the x-axis (Default: FALSE)
`theme.params`	RAdamant graphics theme. (Default: getCurrentTheme())
`xtitle`	Title for the x-axis (Default: "Time")
`ytitle`	Title for the y-axis (Default: "Frequency" or "Periodicity" depending on the value of show.periodicity)
`plot3d`	Logical. If TRUE, 3D spectrogram is plotted.
`overrides`	List of parameters to override the theme. Only parameters that match those defined by the theme are overridden (Default: list(...))
`...`	Used to quickly specify theme overrides.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

specgram.

Examples

```
# Load sample financial series data
data(ex_fs)

# 3D spectrogram
spec = specgram(ex_fs, plot = FALSE)
# Plotting
plot(spec, plot3d = TRUE)
```

---

pmreg *Print (Multi)-Regression object*

Description

Print method for classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'reg'
print(x, ...)

## S3 method for class 'mreg'
print(x, ...)
```

Arguments

x	Instance of class 'reg'/'mreg'.
...	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

mreg.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);

# Run Multi-Regression
mod = mreg(Y, X = cbind(X1, X2), plot = FALSE);
# Print object
mod
```

---

ppo *Percentage Price oscillator*

Description

Compute Percentage Price oscillator (Technical Analysis)

Usage

```
ppo(X, fast.lag = 10, slow.lag = 30, plot = TRUE, ...)
```

Arguments

`X`	X
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

ppredvar — *Print Vector AutoRegressive predictions*

Description

Print method for class 'predVecAr'.

Usage

```
## S3 method for class 'predVecAr'
print(x, ...)
```

Arguments

`x`	Instance of class 'predVecAr'.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `predict.VecAr`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Run 5-step ahead prediction
predict(mod, steps=5)
```

prbsar — *Parabolic Stop and Reverse (PSAR)*

Description

Compute Parabolic Stop and Reverse (PSAR) (Technical Analysis)

Usage

```
prbsar(Close, High, Low
     , accel = c(0.02, 0.2)
     , plot = FALSE
     , ...
     )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`accel`	accel
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`prdvecar` *Vector AutoRegressive Prediction*

Description

Predict method for class 'VecAr'

Usage

```
## S3 method for class 'VecAr'
predict(object
    , exog = NULL
    , steps = 5
    , ci = 0.95
    , simulate = FALSE
    , sd.sim = 1
    , aggregate = TRUE
    , scenarios = 1
    , plot = TRUE
    , ...
    )
```

Arguments

`object`	Instance of class 'VecAr'
`exog`	A matrix or data frame containing the exogenous variables to be used for the prediction.
`steps`	The number of prediction steps
`ci`	The confidence level used to calculate the prediction error.
`simulate`	Logical. If TRUE, a random innovation term is added to each prediction equation (Default: FALSE).
`sd.sim`	The variance of the innovation term (Default: 1).

`aggregate`	Logical. If TRUE, the results from all prediction scenarios will be aggregated (Default: TRUE).
`scenarios`	The number of scenarios to simulate (Default: 1).
`plot`	Logical. If TRUE, results are plotted (Default: TRUE).
`...`	Additional parameters passed to the cplot function.

Value

An object of class "predVecAr". The structure depends on the 'aggregate' parameter:

- aggregate = TRUE: A matrix (steps, 3*Nvars+I) of predictions and confidence intervals. Here 'Nvars' is the number of variables in the VAR model; 'I' is one if the VAR includes the intercept term and zero otherwise.
- aggregate = FALSE: An array of dimensions (steps, 3*Nvars+I, scenarios).

The following attributes are attached to the object:

- snames: The names of the series modelled by the VAR.
- ci: The confidence level.
- aggregate: The input parameter.
- formula: List of formula objects. one for each model equation.
- fcast.se: The forecast standard error.
- fitted: fitted values of the VAR model, as returned by fitted(object).

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `fitted.VecAr`, `cplot`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Run 5-step ahead standard prediction
pred = predict(mod, steps=5);

# Simulate 200 scenarios with 50-step ahead predictions.
# All scenarios are averaged.
# Confidence Intervals are computed from the empirical quantiles.
sim = predict(mod
    , steps = 50
    , simulate = TRUE
    , scenarios = 200
    # Plotting overrides
    , shaded = TRUE
```

```
, shade.density = 50
, shade.angle = 30
);
```

`preder` *Prediction error*

Description

Measures for model evaluation

Usage

```
pred_error(target, pred, pc = FALSE)

av_er(target, pred, pc=FALSE)

abs_avdi(target, pred, pc=FALSE)

mse(target, pred)

sde(target, pred)

track_sign(target, pred)

track_sign_exp(target, pred)
```

Arguments

`target`	Vector. Observed target value
`pred`	Vector. Predicted values
`pc`	Logical. If TRUE return results in percentage

Details

- pred_error: Prediction error
- av_er: Average error
- abs_avdi: Absolute average discard
- mse: Mean squared error
- sde: Error standard deviation
- track_sign: Error track signal
- track_sign_exp: Exponential track signal

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`predgar` *Predict Garch model*

Description

Predict method for Garch models

Usage

```
## S3 method for class 'Garch'
predict(object, plot = TRUE, ...)
```

Arguments

`object`	An object of class "Garch".
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods

Value

A numeric matrix nX4 containing:

`Returns_ME`	Predicted values for returns - mean equation
`Lower_SE`	Lower standard error for predicted returns
`Upper_SE`	Upper standard error for predicted returns
`Pred_Variance`	Predicted values for variance - variance equation

The graphical output window is divided in two parts:

`Upper`	Predicted values for returns - mean equation
`Lower`	Predicted values for variance - variance equation

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
## Calculate three different GARCH models and show predictions
    # load example time series
    data(ex_ts)
    x = ex_ts

    # GARCH example
    gg1 = Garch(x, order = c(2,1), type="garch")
    predict(gg1)

    # EGARCH example
    gg2 = Garch(x, type="egarch")
    predict(gg2)
```

```
# TGARCH example
gg3 = Garch(x, type="tgarch")
predict(gg3)
```

`predmreg` *Predict methods for Multi-Regression*

Description

Predict method for class 'reg'/'mreg'.

Usage

```
## S3 method for class 'mreg'
predict(object, ...)

## S3 method for class 'reg'
predict(object
    , na.rm = FALSE
    , newdata = NULL
    , ci = 0.95
    , mode = c("response", "link")
    , plot = FALSE
    , shaded = FALSE
    , xlabels = NULL
    , main = "Linear Model Prediction"
    , legend = NULL
    , theme.params = getCurrentTheme()
    , aggregate = TRUE
    , ...
    )
```

Arguments

`object` An instance of class 'reg'/'mreg'.

`na.rm` Logical. If TRUE, records containing NA are removed (Default: FALSE).

`newdata` Contains the regressors to be used for the prediction. If NULL, the fitted values are used.
The structure must be one of the following:

- A matrix or data frame with columns named as the regressors (these names will be matched to the ones in the model).
- An array of dimensions (Nsteps, Nvars, Nscenarios). Here 'Nsteps' is the number of forecast steps; 'Nvars' is the number of variables used for computing the prediction; 'Nscenarios' is the number of scenarios for which the forecast is computed.

`ci` Confidence Intervals around the preditions

`mode` The type of prediction:

- "response": prediction is on the scale of the response variable.
- "link": prediction is on the scale of the linear predictors.

`plot`	Logical. If TRUE, results are plotted.
`shaded`	Logical. If TRUE, a shaded area is drawed around the confidence intervals.
`xlabels`	Labels for the x-axis.
`main`	Plot Title
`legend`	The legend text.
`theme.params`	RAdamant graphics theme.
`aggregate`	Logical. If TRUE, results are aggregated when the input argument 'newdata' is an array of scenarios.
`...`	Additional arguments passed to cplot and shade.plot.

Details

predict.mreg makes a call to predict.reg for each model defined by object.

Value

A list of entries (one for each model) if object is an instance of class 'mreg'. Each entry is the result of a call to 'predict.reg'. The structure of the result produced by predict.reg depends on the 'aggregate' parameter:

- aggregate = TRUE: A matrix with columns [fit, lwr, upr] (Predition, Lower C.I., Upper C.I.). Confidence intervals are computed assuming normal distribution of the residuals if newdata = NULL or scenarios = 1. When newdata != NULL and scenarios > 1 then the three columns are calculated by average and empirical quantiles across the predictions of all the scenarios.
- aggregate = FALSE: An array of dimensions (NROW(newdata), 3, scenarios). Each scenario 'i' (extracted from obj[, , i]) is a matrix of columns [fit, lwr, upr].

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

mreg, cplot, shade.plot.

Examples

```
# Generate some random data
N = 20;
x1 = 1:N;
x2 = log(x1);

# Define a model
y = x1 - 2*x2 + 0.5*rnorm(N);
# Estimate the model
mod = lm(y ~ x1 + x2);

# Run prediction
predict.reg(mod
            , plot = TRUE
```

```
# Use a different theme
, theme.params = getTheme(2)
# Add shade around confidence intervals
, shaded = TRUE
# Use two colors for the shade
# Colors will be interpolated
, shade.col = 1:2
, shade.stripes = 30
# Make lines thicker
, lwd = 2
)
```

`printes` *Print Expeted Shortfall*

Description

Print method for class 'ES'.

Usage

```
## S3 method for class 'ES'
print(x, ...)
```

Arguments

`x`	Instance of class 'ES'.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`ES`.

Examples

```
data(ex_ptf);
# Compute ES on multiple confidence levels (Normal)
ES(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "Normal");
```

printfft — *Print FFT results*

Description

Print method for class 'FFT'

Usage

```
## S3 method for class 'FFT'
print(x, ...)
```

Arguments

`x`	Instance of class 'FFT'
`...`	Further arguments to and from other methods

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

printfs — *Print fs data*

Description

Print method for Financial Series (fs) object.

Usage

```
## S3 method for class 'fs'
print(x, ...)
```

Arguments

`x`	Instance of class 'fs'
`...`	Not Used. For compatibility with the generics print function.

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

printvar — *Print VaR results*

Description

Print method for class 'VaR'

Usage

```
## S3 method for class 'VaR'
print(x, ...)
```

Arguments

`x`	Instance of class 'VaR'
`...`	Further arguments to and from other methods

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

prnvecar — *Print Vector AutoRegressive Model*

Description

Print method for class 'VecAr'.

Usage

```
## S3 method for class 'VecAr'
print(x, ...)
```

Arguments

`x`	Instance of class 'VecAr'.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `print.mreg`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)
mod
```

`pro` *Price oscillator*

Description

Compute Price oscillator (Technical Analysis)

Usage

```
pro(Close, fast.lag = 5, slow.lag = 10, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`fast.lag`	fast.lag
`slow.lag`	slow.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`project` *Draw Projection Lines*

Description

Draw vertical connecting lines between two time series.

Usage

```
draw.projections(X, Y, Y.fit
, col = getCurrentTheme()[["projection.col"]][1]
, type = getCurrentTheme()[["projection.type"]][1]
, lty = getCurrentTheme()[["projection.lty"]][1]
)
```

Arguments

`X`	The x-axis values (common to Y and Y.fit) where the y-values are evaluated.
`Y`	The y-values of one of the endpoint of the projection lines.
`Y.fit`	The y-values of the other endpoint of the projection lines.
`col`	The color of the line
`type`	The endpoints type
`lty`	The line type

Value

Void

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`cplot`

Examples

```
# Define and plot two series
X1 = 1:10;
X2 = X1 + rnorm(10);
cplot(cbind(X1, X2));
draw.projections(X = 1:10, Y = X2, Y.fit = X1, type = "o");

# Use a different baseline
base = seq(-2, 2, len=10);
cplot(cbind(X1, X2)
    , base = base
    # plot line and points for X1, only points for X2
    , type = c("o", "p")
    # The size of the points for X1 and X2
    , cex = c(0.5, 0.8)
    # Remove x-labels rotation
    , xlab.srt = 0
    );
draw.projections(X = base, Y = X2, Y.fit = X1);
```

`psme` *Print Sample Mean Excess class*

Description

Printing function for Sample Mean Excess class

Usage

```
## S3 method for class 'sme'
print(x, ...)
```

Arguments

`x`	OBJECT of class "sme".
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`ptfoper` *Portfolio operators*

Description

Get portfolio Beta

Usage

```
PtfRet(PTF
     , w = NULL
     , glob = TRUE
     , calc.ret = FALSE
     , ...
     )

PtfVar(PTF
     , w = NULL
     , glob = TRUE
     , vol = FALSE
     , calc.ret = FALSE
     , ...
     )
```

```
PtfBeta(beta
    , w = NULL
    , glob = TRUE
    )
```

Arguments

PTF	Matrix containing one or more series of prices/returns, one time series for each asset
w	Vector of portfolio weights
glob	Logical. If TRUE return the value for the whole portfolio.
vol	Logical. If TRUE returns volatility (standard deviation instead of variance).
calc.ret	Logical. If TRUE the input matrix is considered as a matrix of prices, so returns are calculated.
beta	Value of the Beta coefficient or an object of class "Capm".
...	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# load example portfolio
data(ex_ptf)

# results for each series
PtfRet(ex_ptf, glob=FALSE)
PtfVar(ex_ptf, glob=FALSE)

# results for the whole portfolio
PtfRet(ex_ptf, glob=TRUE)
PtfVar(ex_ptf, glob=TRUE)

# Example with a series of prices instead of returns
data(EuStockMarkets)
PtfRet(PTF = EuStockMarkets
    , w = c(0.3, 0.4, 0.2, 0.1)
    , calc.ret = TRUE
    )

PtfRet(PTF = EuStockMarkets
    , w = c(0.3, 0.4, 0.2, 0.1)
    , glob = FALSE
    , calc.ret = TRUE
    )
```

`ptfopt` *Mean-Variance optimum portfolio*

Description

Calculate mean-variance efficient portfolio

Usage

```
PtfOpt(ret = NULL
, ptf = NULL
, mi = NULL
, SIGMA = NULL
, volatility = TRUE
, ...
)

## S3 method for class 'PtfOpt'
print(x, ...)
```

Arguments

`ret`	Vector containing averge return for each asset
`ptf`	Matrix containing one or more series of prices, one time series for each asset
`mi`	Target return for the portfolio
`SIGMA`	Sample covariance matrix
`volatility`	Logical. If TRUE volatility is returned, else the variance is computed.
`x`	An object of class "PrfOpt".
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`PtfFront`, `PtfUtility`

Examples

```
# Calculate weights from a series of prices
data(EuStockMarkets);
PtfOpt(ptf = EuStockMarkets);

# simulate efficient frontier
PtfFront(PTF = EuStockMarkets
, n_sim=100
, col="yellow"
);
PtfFront(PTF = EuStockMarkets
```

```
, n_sim=30
, col="green"
);

# calculate weights from a vector of returns R and matrix SIGMA
R = c(A = 0.021, B = 0.09);
SIGMA = matrix(c(0.101^2, 0.005,  0.005, 0.208^2)
               , nrow = 2
               , ncol = 2
               );
# set target returns to be 0.05
PtfOpt(ret = R
, ptf = NULL
, SIGMA = SIGMA
, mi = c(0.05)
);

# set two target returns: 0.05 and 0.07
PtfOpt(ret = R
, ptf = NULL
, SIGMA = SIGMA
, mi = c(0.05, 0.07)
);
# simulate efficient frontier
PtfFront(ret = R
 , ptf = NULL
 , SIGMA = SIGMA
 , n_sim=100
 , col="yellow"
 );


## Example with real time series
fromDt = as.Date("2010-01-01");
ACME = get.fs("APKT",SName = "Acme Packet", from = fromDt);
ABTL = get.fs("ABTL", SName = "Autobytel", from = fromDt);
CNAF = get.fs("CNAF", from = fromDt);
BIIB = get.fs("BIIB", SName = "Biogen", from = fromDt);
SONY = get.fs("SNE", SName = "Sony", from = fromDt);
ENI = get.fs("E", SName = "Eni", from = fromDt);
ptf = combine.fs(ACME, ABTL, CNAF, BIIB, SONY, ENI);
head(ptf);

# Compute Minimum Variance portfolio
PtfOpt(ptf = ptf);
```

`ptfront` *Portfolio efficient frontier*

Description

Compute/Simulate portfolio mean-variance efficient frontier

Usage

```
PtfFront(PTF = NULL
    , ret = NULL
    , SIGMA = NULL
    , mi = NULL
    , n_sim = 10
    , volatility = TRUE
    , plot = TRUE
    , main = paste("Frontier Simulation:"
                   , ifelse(is.null(mi)
                            , n_sim
                            , length(mi)
                            )
                   , "points"
                   )
    , xtitle = ifelse(volatility
                      , expression(sigma)
                      , expression(sigma^2)
                      )
    , ytitle = expression(mu)
    , xlab.srt = 0
    , ytitle.srt = 0
    , type = "o"
    , legend = "Mean-Variance Frontier"
    , ...
    )
```

Arguments

`PTF`	PTF
`ret`	ret
`SIGMA`	SIGMA
`mi`	mi
`n_sim`	n_sim
`volatility`	volatility
`plot`	plot
`main`	main
`xtitle`	xtitle
`ytitle`	ytitle
`xlab.srt`	xlab.srt
`ytitle.srt`	ytitle.srt
`type`	type
`legend`	legend
`...`	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`ptfutil` *Portfolio Utility*

Description

Calculate utility and plot for efficient portfolio

Usage

```
PtfUtility(PTF = NULL, W, R = NULL, SIGMA = NULL,
af = 3, plot = TRUE, ...)
```

Arguments

`PTF`	Matrix containing TWO series of returns, one series for each asset.
`W`	Initial vector of weights.
`R`	Vector of PTF returns.
`SIGMA`	PTF sample covariance matrix.
`af`	Numeric (range: 0,1). Adversion factor (Default: 3)
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

`PtfFront`, `PtfOpt`

Examples

```
# vector of returns for two assets A and B
R = c(A=0.021, B=0.09)
# Covariance matrix
SIGMA = matrix(c(0.101^2, 0.005,  0.005, 0.208^2),2,2)
# Calculate and show utility for the two assets
PtfUtility(PTF=NULL, R=R, SIGMA=SIGMA, W=c(0.4,0.6))
```

`pvt` *Price Volume trend indicator*

Description

Compute Price Volume trend indicator (Technical Analysis)

Usage

```
pvt(Close, Volume, lag = 5, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`Volume`	Vector. Asset traded Volume.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`qgev` *Generalised Extreme Value (GEV) - Quantile function*

Description

Generalised Extreme Value (GEV) - Quantile function

Usage

```
qgev(P, mu = 0, xi = 0.1, sigma = 1)
```

Arguments

`P`	P
`mu`	mu
`xi`	xi
`sigma`	sigma

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`qgpd` *Generalised Pareto Distribution (GPD) - Quantile function*

Description

Generalised Pareto Distribution (GPD) - Quantile function

Usage

```
qgpd(P, xi = 0.1, sigma = 1, trsh = 0)
```

Arguments

`P`	P
`xi`	xi
`sigma`	sigma
`trsh`	trsh

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`recref` *Recode and Reformat*

Description

Change the attributes and format of vector or data frame

Usage

```
recode(x, old, new)
reformat(X, classes)
```

Arguments

`x`	Vector input.
`X`	Matrix or Data frame input
`old`	Old (actual) unique values in the vector
`new`	New values to be placed in the vector
`classes`	Vector containing the classes to be applied to X. The vector must contain one class for each column of the input X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# create random numeric vector
old_vec = sample(c(1,2,3), 10, TRUE)
# old values
old = unique(old_vec)
# new values
new = c("low","medium","high")
# new vector
new_vec = recode(old_vec, old=old, new=new)
```

`recycle` *Recycle function for time series*

Description

Recycle an input sequence X to get a new sequence of the specified length V

Usage

```
recycle(X, V = length(X))
```

Arguments

`X`	X
`V`	V

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`relvol` *Relative Volatility oscillator*

Description

Compute Relative Volatility oscillator (Technical Analysis)

Usage

```
RelVol(Close, sdlag = 9, lag = 5)
```

Arguments

`Close`	Vector. Close price.
`sdlag`	sdlag
`lag`	Integer. Number of lag periods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`rema` *Regularised Exponential Moving Averages*

Description

Compute multiple Regularised Exponential Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
rema(X, win.size = NROW(X), alpha = 0.5, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`alpha`	weight in the interval [0, 1]. (Default: 0.7).
`plot`	Logical. Return plot.
`...`	Additional parameters for future development.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
REMA is a second order IIR filter with the two coefficients are regulated by the smoothing factors lambda and alpha.
Smoothing factors: lambda = 2/(win.size+1) and alpha.

Value

A object of class 'ma' with attributes type = "REMA", 'lambda' and 'alpha':
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
rema(x, 10, alpha=0.5)
# compute moving average with multiple lags
rema(x, c(10,20), alpha=0.3)

## Not run:

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
rema(ex_fs, 30, plot=TRUE)
# multiple lags
rema(ex_fs, seq(5,50,10), plot=TRUE)

## End(Not run)
```

---

residreg *Extract Model Residuals for (Multi)-Regression object*

Description

Generic method for extracting model residuals from object of classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'reg'
residuals(object, na.rm = FALSE, ...)

## S3 method for class 'mreg'
residuals(object, na.rm = FALSE, ...)
```

Arguments

object	Instance of class 'reg'/'mreg'.
na.rm	Logical. If TRUE, NA records are removed.
...	Further arguments to or from other methods.

Value

One of the following:

- class 'mreg': A matrix containing all model residuals, one column for each model.
- class 'reg': A matrix containing the model specific residuals.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

mreg.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y1 = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);
Y2 = -2 + 1.2*X1 -X2 + rnorm(N, sd = sigma);
# Add some NA
Y2[1:3] = NA

# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2), X = cbind(X1, X2), plot = FALSE);

# Extract all coefficients
residuals(mod)
residuals(mod, na.rm = TRUE)

# Extract coefficients from the second model
residuals(mod[[2]])
residuals(mod[[2]], na.rm = TRUE)
```

---

resvecar *Extract Model Residuals from Vector AutoRegressive object*

Description

Generic method for extracting model residuals from object of class 'VecAr'.

Usage

```
## S3 method for class 'VecAr'
residuals(object, na.rm = FALSE, ...)
```

Arguments

`object`	Instance of class 'VecAr'.
`na.rm`	Logical. If TRUE, NA records are removed.
`...`	Further arguments to or from other methods.

Value

A matrix containing all model residuals, one column for each model.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `residuals.mreg`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Extract residuals (note NA due to the lagged data)
residuals(mod)
residuals(mod, na.rm = TRUE)
```

`rgev` *Generalised Extreme Value (GEV) - Random Numbers Generator*

Description

Generalised Extreme Value (GEV) - Random Numbers Generator

Usage

```
rgev(N, mu = 0, xi = 0.1, sigma = 1)
```

Arguments

`N`	N
`mu`	mu
`xi`	xi
`sigma`	sigma

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`rgpd` *Generalised Pareto Distribution (GPD) - Random Numbers Generator*

Description

Generalised Pareto Distribution (GPD) - Random Numbers Generator

Usage

```
rgpd(n, xi = 0.1, sigma = 1, trsh = 0)
```

Arguments

`n`	n
`xi`	xi
`sigma`	sigma
`trsh`	trsh

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`roc` *Rate of Change index*

Description

Compute Rate of Change index (Technical Analysis)

Usage

```
roc(X, lag = 5, pc = TRUE, plot = TRUE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`pc`	pc
`plot`	LOGICAL. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

rowmax — *Maximum / Minimum by row*

Description

`rowMax`: Compute parallel max across the rows of X
`rowMin`: Compute parallel min across the rows of X

Usage

```
rowMax(X)
rowMin(X)
```

Arguments

`X`	Input matrix/sequence

Value

A matrix NROW(X) by one, where each row is the max / min of the rows of X).

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

---

rschint — *Interval for uniroot function*

Description

Compute a proper search interval for uniroot function

Usage

```
root.search.interval(from, func = NULL,
type = c("left", "both", "right"), max.iter = 500,
show.warnings = FALSE, debug = FALSE, ...)
```

Arguments

`from`	from
`func`	func
`type`	type
`max.iter`	max.iter
`show.warnings`	show.warnings
`debug`	debug
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`rsi` *Relative strength indicator*

Description

Compute Relative strength indicator (Technical Analysis)

Usage

```
rsi(X, lag, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

runlog — *Error Handling and Log with runner*

Description

write.log: Simple function to write/append log to file (csv format).
error.handling: Error handling function

Usage

```
write.log(log = matrix(NA, nrow = 0, ncol = 0)
          , logfile = "runlog.log"
         )

error.handling(err)
```

Arguments

log	Matrix containing logging information.
logfile	Filename of the log
err	List containing the status code of the error.

Details

Function error.handling is to be called ONLY inside a tryCatch statement.
It assigns three variables:
- log.status = "Failed": the status of the execution is set to "Failed"
- log.message: The error message generated inside the tryCatch
- res = NA: the result is set to NA

Value

VOID

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

run, multirun

`runner` *Runner and Multirunner*

Description

Wrapper function to execute any function.
Run single or multiple functions and provide a list of results.

Usage

```
run(func = NULL
    , args = list()
    , writelog = TRUE
    , logfile = "runlog.log"
    , check.input = TRUE
    , output = c("console", "sing.file")
    )

multirun(func.array = character(0)
    , args.list = list()
    , writelog = TRUE
    , logfile = "runlog.log"
    , output = c("console", "sing.file", "multi.file")
    )
```

Arguments

`func`	Name of the function to run
`func.array`	Array of function names to execute
`args`	Named list of parameters of the function. Each entry is of the form: args[["PARAM.NAME"]] = VALUE.
`args.list`	Array of named list of parameters of the function. Each entry is a list of parameters, as required by the wrapper function "run".
`writelog`	Logical. If TRUE, execution log is written to file.
`logfile`	Filename of the log
`check.input`	Logical. If TRUE, basic checks are performed on input data, and stop code execution in case of wrong data.
`output`	Choose wether to return the results in the console or export the to text file.

Details

When called the function `multirun` the elements of the argument `args.list` can be specified with or without names. If the names are specified the arguments can be put in a different order from the array function.
If `writelog = TRUE` a log containing information about submitted computation is saved in the current working directory. If `output` = "sing.file", a text file containing all the results is saved in current working directory.
The file will be named "Run_time_date.txt" If `output` = "sing.file", a text for each called function is saved in a text file.
The files will be named "Function Name_time_date.txt"

Value

The object returned depends on the function being called.
`multirun` returns a list of results, one entry for each function being executed.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`write.log`, `error.handling`

Examples

```
# Run Exponential Moving Average and Simple Moving Average.
# For each function a list of parameters has been specified
multirun(c("ema","sma")
,list(list(rnorm(150), 5)
  , list(rnorm(100), 10)
  )
, writelog = TRUE
)
# Specifies names in the list of arguments
multirun(func.array = c("ema","sma")
,args.list = list(sma = list(rnorm(150), 5)
 , ema = list(rnorm(100), 30)
 )
, TRUE
)
# Output to text file
multirun(func.array = c("ema","sma")
, args.list = list(sma = list(rnorm(150), 5)
  , ema = list(rnorm(100), 30)
  )
, output = "multi.file"
)
```

---

rvi *Relative Vigor indicator*

Description

Compute Relative Vigor indicator (Technical Analysis)

Usage

```
rvi(Close
    , High = NULL
    , Low = NULL
    , Open = NULL
    , plot = TRUE
    , ...
    )
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Open`	Vector. Open price.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`scaledf` *Apply functions on a scaled window*

Description

`scalApply`: Applies a given function to the pairs (X[n, i], X[n-lag, i]).
`scalMax`: Scaled max on each column of the input matrix. `scalMin`: Scaled min on each column of the input matrix

Usage

```
scalApply(X
        , lag = 0
        , padding = NA
        , na.rm = FALSE
        , func = NULL
        , ...
        )

scalMax(X
        , lag = 1
        , padding = -Inf
        , na.rm = FALSE
        , func = NULL
        )

scalMin(X
        , lag = 1
        , padding = Inf
        , na.rm = FALSE
        , func = NULL
        )
```

Arguments

`X`	Input matrix/sequence
`lag`	vector of integer lags. If lag >= 0 data are shifted to the right, else to the left.(Default: 0)
`padding`	value used to initialise the output matrix (Default: NA)
`na.rm`	Logical. If TRUE, N-lag entries are removed from the output (Default: FALSE)
`func`	function applied to the data (Default: NULL)
`...`	Additional parameters accepted by the function 'func'

Details

Sequences are treated as one-column matrices.

Value

A matrix where func/max/min has been applied on each pair (X[n, i], X[n-lag,i]) for each column i of X.
Number of rows depends on the na.rm parameter. Number of columns is NCOL(X)

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`scorecd` *Score Card*

Description

Create Credit Score Card based on Logistic Regression

Usage

```
Score.card(X, Y, nseg = 2, col.classes=NULL)

## S3 method for class 'scorecard'
print(x, ...)

## S3 method for class 'scorecard'
summary(object, plot=FALSE, ...)
## S3 method for class 'scorecard'
predict(object, ...)
```

Arguments

`X`	DATA.FRAME / MATRIX of regressors.
`Y`	Vector. Target variable in 0-1 format.
`nseg`	INTEGER / VECTOR. Number of segments to factorise numerical variables.
`col.classes`	Vector. Indicate the format to use for each variable (Numeric / Character). If NULL the original input formats are maintained.

`x, object`	an object of class "scorecard"
`plot`	Logical. If TRUE accuracy plots are displayed: • Lift Chart, `Lift` • Cumulative Gain, `Gain` • ROC, `ROCplot` • Sensitivity VS Specificity
`...`	Further arguments to or from other methods.

Details

The input X can contain both numerical and categorical variables.
All the input variables are converted according to the results of Weight of Evidence calculation (`WeightEvid`). Numerical variables are factorised according with the number of segments indicated by the parameter "nseg".

Value

The function returns an object of class "scorecard" containing:

`Scorecard`	: data frame containing the score card results ("Variable", "Segment", "WoE", "Est.Coef", "Wald-Z", "P-Val", "Ods_ratio", "Score", "Round.Score");
`Model`	: an object of class "glm" - "lm" with the results of logistic model (see `glm`);
`WeightOfEvidence`	: A matrix containing the results of Weight of Evidence calculation (see `WeightEvid`);

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`WeightEvid`, `input2woe`, `glm`

Examples

```
# load example data set
data(ex_credit)

## Generate Score Card
data = ex_credit[ ,-1]
target = ex_credit[ ,1]

# Two segments for numerical variables
sc2 = Score.card(X=data, Y=target, nseg = c(2,4))
sc2
# Three segments for numerical variables
sc3 = Score.card(X=data, Y=target, nseg = c(2,3,4))
sc3

# display more detailed results with the method summary
summary(sc2)
```

```
summary(sc3)

# ... show plots
# display more detailed results with the method summary
summary(sc2, plot=TRUE)
summary(sc3, plot=TRUE)
```

---

sensan *Sensitivity Analysis*

Description

Generic method for parameter sensitivity analysis on regression models.

Usage

```
sensAnalysis(X, ...)

## Default S3 method:
sensAnalysis(X, win.size = length(coef(X)), plot = FALSE, ...)
```

Arguments

`X`	A regression model. Instance of class 'lm', 'glm'.
`win.size`	The initial window size for the analysis. See `splitWindow` for details.
`plot`	Logical. If TRUE, results are plotted.
`...`	Further arguments passed to splitWindow and cplot.

Value

An object of class 'sensAnalysis'. This is a list with the following elements:

`coeffs`	Matrix of regression coefficients estimated on each portion of data delimited by the indexes computed by splitWindow.
`weights`	Matrix of regression weights as computed by get.lm.weights.
`pvalues`	Matrix of p-values of the regression coefficiens.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`splitWindow`, `get.lm.weights`, `plot.sensAnalysis`, `cplot`.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);

# Run Regression
mod = lm(Y ~ X1 + X2);

# Perform Sensitivity Analysis, Forward Extended Window (Default)
sensAnalysis(mod
    # Starting with 10 samples
    , win.size = 10
    # Increment by 5 points at each step
    , by = 5
    )
```

---

sensanlm *Sensitivity analysis method for lm*

Description

Sensitivity analysis method for lm

Usage

```
## S3 method for class 'lm'
sensAnalysis(X, ...)
```

Arguments

X	OBJECT of class "lm".
...	Further arguments to or from other methods

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`sensanrg` *Sensitivity Analysis for Multi-Regression Models*

Description

Sensitivity analysis method for classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'reg'
sensAnalysis(X, ...)

## S3 method for class 'mreg'
sensAnalysis(X, ...)
```

Arguments

`X`	A regression model. Instance of class 'reg', 'mreg'.
`...`	Further arguments passed to the default method.

Value

An instance of class 'sensAnalysis' if X has class 'reg', or a list of length(X) objects of class 'sensAnalysis' if X has class 'mreg'.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`sensAnalysis`, `mreg`, `plot.sensAnalysis`, `cplot`.

Examples

```
# Generate some random data
N = 50;
sigma = 0.1;
X1 = cumsum(rnorm(N));
X2 = rnorm(N);
X3 = cumsum(rnorm(N));
X4 = rnorm(N);

# Define a linear model
Y1 = 1.5 + X1 + 2*X3 + rnorm(N, sd = sigma);
# Define a logit model
Y2 = inv.logit(-2.2 + 0.3*X2 - 0.2*X4 + rnorm(N, sd = sigma));

# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2)
, X = cbind(X1, X2, X3, X4)
# Stepwise regression
, type = "stepwise"
```

```
# lm on Y1 and glm on Y2
, mode = c("lm", "glm")
# Set the family.
# It is recycled but family is only used for glm
, family = "binomial"
# Constrain the maximum number of variables
# that can enter the regression
, max.vars = c(3, 2)
# Use another theme
, theme.params = getTheme(2)
);

# Perform Sensitivity Analysis, Backward Sliding Window
sensAnalysis(mod
# Sliding Window with 20 samples
, mode = "SW"
, win.size = 20
# Shift by 5 points backward at each step
, direction = "backward"
, by = 5
# Plot results
, plot = TRUE
# Override theme - show all labels on the x-axis
, x.ticks = "ALL"
)
```

`sensplot` *Plot Sensitivity Analysis*

Description

Plot method for class 'sensAnalysis'.

Usage

```
## S3 method for class 'sensAnalysis'
plot(x
, main = NULL
, xlabels = rownames(x$coeffs)
, xtitle = ""
, theme.params = getCurrentTheme()
, ...
)
```

Arguments

`x`	A Sensitivity Analysis object. Instance of class 'sensAnalysis'.
`main`	Main plot title
`xlabels`	Labels for the x-axis
`xtitle`	Title for the x-axis
`theme.params`	RAdamant graphics theme.
`...`	Further arguments passed to the cplot function.

Value

Void

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

sensAnalysis, mreg, plot.sensAnalysis, cplot.

Examples

```
# Generate some random data
N = 50;
sigma = 0.1;
X1 = cumsum(rnorm(N));
X2 = rnorm(N);
X3 = cumsum(rnorm(N));
X4 = rnorm(N);

# Define a linear model
Y1 = 1.5 + X1 + 2*X3 + rnorm(N, sd = sigma);
# Define a logit model
Y2 = inv.logit(-2.2 + 0.3*X2 - 0.2*X4 + rnorm(N, sd = sigma));

# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2)
, X = cbind(X1, X2, X3, X4)
# Stepwise regression
, type = "stepwise"
# lm on Y1 and glm on Y2
, mode = c("lm", "glm")
# Set the family.
# It is recycled but family is only used for glm
, family = "binomial"
# Constrain the maximum number of variables
# that can enter the regression
, max.vars = c(3, 2)
# Use another theme
, theme.params = getTheme(2)
);

# Perform Sensitivity Analysis, Backward Sliding Window
res = sensAnalysis(mod
# Sliding Window with 20 samples
, mode = "SW"
, win.size = 20
# Shift by 5 points backward at each step
, direction = "backward"
, by = 5
);

# Plot results for the first model
plot(res[[1]]
# Use another theme
```

```
, theme.params = getTheme(2)
# Override theme - show all labels on the x-axis
, x.ticks = "ALL"
)
```

`sharpe` *Sharpe index*

Description

`Sharpe`: Calculate Sharpe index for a portfolio.
`Sharpe.Capm`: Get Sharpe index from an object of class. "Capm"

Usage

```
Sharpe(PTF, ...)
## Default S3 method:
Sharpe(PTF, rfr = 0, ...)
## S3 method for class 'Capm'
Sharpe(PTF, rfr = 0, ...)
```

Arguments

`PTF`	Input portfolio or an object of class "Capm"
`rfr`	risk free rate
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Treynor`, `Jensen`, `Appraisal`

`sinma` *(Normalised) Sine Weighted Moving Averages*

Description

Compute multiple (Normalised) Sine Weighted Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
sinma(X, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. Return plot.
`...`	Further arguments to or from other methods

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
Weights: sin(pi * (1:win.size)/(win.size+1))

Value

A object of class 'ma' with attributes type = "SINMA" and 'win.size' as from the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Movav`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
sinma(x, 10)
# compute moving average with multiple lags
sinma(x, c(10,20))

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
sinma(x, 30, plot = TRUE)
# multiple lags
sinma(x, seq(5,50,10), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
sinma(ex_fs, 30, plot=TRUE)
# multiple lags
sinma(ex_fs, seq(5,50,10), plot=TRUE)
```

```
## End(Not run)
```

---

sma *Simple Moving Average*

Description

Compute multiple Simple Moving Averages on the input data, one for each column of X[, i] and window size win.size[j]

Usage

```
sma(X, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function Mmovav.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'ma' with attributes type = "SMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`ema`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
sma(x, 15)
# compute moving average with multiple lags
```

```
sma(x, c(15,30))

## Not run:
# refine results of moving average
setCurrentTheme(2)
sma(x, 30, plot = TRUE)
# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
sma(ex_fs, 30, plot=TRUE)
# multiple lags
sma(ex_fs, seq(5,50,5), plot=TRUE)

## End(Not run)
```

---

sme *Sample Mean Excess function*

Description

Sample Mean Excess function

Usage

```
sme(X, plot = TRUE, ...)
```

Arguments

X	X
plot	plot
...	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

`specgram` *Spectrogram using short-time Fourier transform*

Description

Computes FFT on each column of X. For Financial series objects (class 'fs'), Close data is extracted.

Usage

```
specgram(X, win.size = max(1, NROW(X)/20), plot = TRUE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	The size of the window used to compute the FFT
`plot`	Logical. If TRUE, spectrogram is plotted.
`...`	Additional parameters passed to splitWindow, FFT and plot.specgram

Details

A forward sliding window of length win.size is used to split the input data into segments, then for each segment the FFT of size NFFT = 2^ceiling(log2(win.size)) is computed.
The sliding of the window is controlled by the 'by' parameter of the splitWindow function (default: by = 1).
The 'by' parameter should take values between 1 and win.size:

- when by = win.size, the input data is split into Nwindows = ceiling(NRowX/win.size) non-overlapping adjacent blocks.
- when by = 1, then Nwindows = NRowX - win.size + 1 overlapping segments are computed.

Value

An object of the class 'specgram'. This is an array with dimensions (NFFT, Nwindows, NColX):

`NFFT`	The FFT length. It is the next power of 2 greater than the length of each segment/window of X.
`Nwindows`	The number of window segments computed. It depends on the 'by' parameter (default is 1) of the splitWindow function (see details).
`NColX`	The number of columns of X.

The following attributes are attached to the object:

`Fs`	The input Fs parameter to the FFT.
`window`	The window function used to smooth the input data.
`freq`	The frequencies where the FFT was evaluated.
`fpoints`	The array indices where the frequency points relative to 'freq' are stored.
`half`	The input half parameter to the FFT.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

splitWindow, FFT, plot.specgram.

Examples

```
# Load sample financial series data
data(ex_fs)

# 3D spectrogram
specgram(ex_fs, plot3d = TRUE)

# Sampling period
Ts = 0.01
# Generate 10 seconds timeline
t = seq(0, 10, by = Ts)
# Sampling frequency
Fs = 1/Ts
# Linear increasing frequency
f = 2*t
#Chirp signal - Cosine of increasing frequency
chirp = as.matrix(cos(2*pi*f*t))
colnames(chirp) = "Chirp"

# 2D spectrogram
specgram(chirp, Fs = Fs)
# 2D spectrogram with non overlapping windows
specgram(chirp, Fs = Fs, win.size = 128, by = 128)
# 3D spectrogram
specgram(chirp, Fs = Fs, win.size = 128, plot3d = TRUE)
```

---

splitwdw *Split Window*

Description

Given an input size N, splits the sequence 1:N into sliding or extended windows and returnn the endpoint indexes of each window.

Usage

```
splitWindow(N
, direction = c("forward", "backward")
, mode = c("EW", "SW")
, from = NULL
, win.size = 1
, by = 1
, labels = 1:N
, ...
)
```

Arguments

`N`	The size of the entire window to be split
`direction`	Controls on which direction the next sub-window is computed. One of "forward" or "backward".
`mode`	Controls how windows endpoint indexes are computed. If "EW" (Extended Windows), starting with an initial window of size win.size at each step the previous sub-window is extended with additional 'by' points on the side specified by 'direction'. If "SW" (Sliding Windows), the size on the windows is constant: at each step the previous sub-window is shifted on by the quantity 'by' on the side specified by 'direction'.
`from`	The starting point from wich the first window is calculated
`win.size`	The initial size of the first window if mode = "EW". The size of all windows if mode = "SW"
`by`	Controls the amount of extension or shift (depending on the mode parameters) of the windows.
`labels`	The labels associated to the N data points of the full window.
`...`	Further arguments to or from other methods.

Value

A matrix with columns [start.idx, end.idx]. Each row represents the endpoints indexes of a corresponding sub-window.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
## Forward Extended Window
splitWindow(N = 30
# Start with a window of size 3
, win.size = 3
# Start from position 5
, from = 10
# Move forward
, direction = "forward"
# Extended mode
, mode = "EW"
# Increase the size by 5 at each step
, by = 5
)

## Backward Extended Window
splitWindow(N = 30
# Start with a window of size 3
, win.size = 3
# Start from position 20
, from = 20
# Move backward
, direction = "backward"
```

```
# Extended mode
, mode = "EW"
# Increase the size by 2 at each step
, by = 2
)

## Forward Sliding Window
splitWindow(N = 30
# windows of size 5
, win.size = 5
# Move forward
, direction = "forward"
# Sliding mode
, mode = "SW"
# Slide forward by 5 at each step. This produces non overlapping windows.
, by = 5
)

## Backward Sliding Window
splitWindow(N = 30
# windows of size 3
, win.size = 3
# Move backward
, direction = "backward"
# Sliding mode
, mode = "SW"
# Slide backward by 5 at each step.
, by = 5
)
```

---

sssym *State Space system simulation*

Description

Generic function for State Space system simulation. The system can be either linear or non linear.

Usage

```
ss.sym(X, F = NULL, G = NULL, H = NULL, D = NULL,
init = 0, SLen = ifelse(is.function(F), NA,
NROW(F)), YLen = ifelse(is.function(H), NA, NROW(H)), ...)
```

Arguments

X	Matrix of data series (one column per variable).
F	[State -> State] transition matrix or [(State, Input) -> State] function (F = function(S, X, n, ...) returning the new state vector S_new based on the current State S and the data X at time period n) (Default: NULL)
G	[Input -> State] transition matrix. Only for linear models (Default: NULL)

`H`	[State -> Output] transition matrix or [(State, Input) -> Output] function (H = function(S, X, n, ...) returning the new output vector Y[, n] based on the new state S[, n] and the data X at time period n) (Default: NULL -> converted in diag(SLen))
`D`	[Input -> Output] transition matrix. Only for linear models (Default: NULL -> converted to a zero matrix SLen by NCOL(X))
`init`	Initial values for the state vactor S (Default: 0, recycled to length SLen if necessary)
`SLen`	Length of the state vector S. (Default: ifelse(is.function(F), NA, NROW(F)))
`YLen`	Number of columns of the output vector Y. (Default: ifelse(is.function(H), NA, NROW(H)))
`...`	Additional parameters accepted by the functions F and H

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'ss' with attributes 'F', 'G', 'H', 'D' as given by the corresponding input parameters:
- matrix of size NROW(X) by YLen, result of the symulation of the given dynamic system subject to input 'X' and initial condition 'init'.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`stacklev` *Retrieve the number of calls in the stack.*

Description

Retrieve the number of calls in the stack. To be called from inside a function.

Usage

```
CallStackLevels()
```

Value

The number of calls in the stack.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Create two nested functions
f1 = function() {
    f2();
}
f2 = function() {
    CallStackLevels()
}

f2(); # Returns 1
f1(); # Returns 2
```

---

starc — *Stoller Starc bands*

Description

Compute Stoller Starc bands (Technical Analysis)

Usage

```
starc(Close, High = NULL, Low = NULL, atr.mult = 2, lag = 5, atr.lag =
14, mov = c("sma", "ema", "wma"), plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`atr.mult`	atr.mult
`lag`	Integer. Number of lag periods.
`atr.lag`	atr.lag
`mov`	mov
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`statbar` *Status bar*

Description

Interactive status bar for updating completion percentage to console.

Usage

```
statusbar(message = "Computing.."
, status = 0
, n = 1
, N = 1
, step = 0.01)
```

Arguments

`message`	The message to be sent to console.
`status`	The percentage of completion (status in [0, 1]).
`n`	The current value of the loop counter.
`N`	The total number of iterations
`step`	The percentage increment by which the status is updated.

Details

This function is meant to be used inside a loop, to inform the user about the current status of the processing.

Value

The updated status for the next iteration.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Number of iterations
N = 1000;
# Set the message
msg = "Still running..";
# Init Status bar
status = 0;
# Set the step to 0.05. The status bar is updated by 5% each time
step = 0.05;

# Start looping
for(n in 1:N) {
    # Do something
    # ... some code ...
```

```
    # Update the status (note how status is reused at each iteration)
    status = statusbar(message = msg
                       , status = status
                       , n = n
                       , N = N
                       , step = step);
}
```

`stepmat` *Step matrix for binomial tree*

Description

Simulate binomial path of a binomial tree

Usage

```
StepMat(init, n_step, up, down)
```

Arguments

`init`	Initial price, step number 0 in the matrix.
`n_step`	Integer. Number of steps.
`up`	Up movement factor
`down`	Down movement factor

Value

Create Step probability matrix of *(n_step+1) x (n_stpe+1)* dimensions

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# simulate binomial path for 10 steps
StepMat(init = 0.5, n_step = 10, up = 0.8, down = 0.6)
```

`strvar` *Structural Vector Autoregressive model*

Description

Estimate Structural Vector Autoregressive model

Usage

```
Strvar.VecAr(X, A = "diag", B = NULL, inter = FALSE, ...)
```

Arguments

`X`	An object of class "VecAr"
`A`	Restriction matrix A.
`B`	Restriction matrix B.
`inter`	Logical. If TRUE restrictions matrix will be manually edited.
`...`	Further arguments to or from other methods

Value

An object list containing the following elements:

`EST_Matrix` List of 2 elements:

- Estimated A parameters
- Estimated B parameters

`SE` List of 2 elements:

- Standard errors of A parameters
- Standard errors of B parameters

`LogLik` Log-Likelihood value.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

optim, VecAr

Examples

```
# load example data sete
data(ex_ptf)
X = ex_ptf[ ,1:4]
# estimate VAR(2) model
vecar = VecAr(X, ar.lags=1:2, type="const")

## Estimate Structural VAR models
# EX. 1
# Default constraints provided by the function:
```

```
# A =      [,1] [,2] [,3] [,4]
#        [1,]    C1    0    0    0
#        [2,]    0    C2    0    0
#        [3,]    0    0    C3    0
#        [4,]    0    0    0    C4
# B =      [,1] [,2] [,3] [,4]
#        [1,]    1    0    0    0
#        [2,]    0    1    0    0
#        [3,]    0    0    1    0
#        [4,]    0    0    0    1

Strvar.VecAr(vecar)

# EX. 2
# Different constraints for A matrix:
# A =      [,1] [,2] [,3] [,4]
#        [1,]    C1    0    0    0
#        [2,]    C2    C3    0    0
#        [3,]    C4    0    C5    0
#        [4,]    C6    0    0    C6
# B =      [,1] [,2] [,3] [,4]
#        [1,]    1    0    0    0
#        [2,]    0    1    0    0
#        [3,]    0    0    1    0
#        [4,]    0    0    0    1

A = diag(NA, 4)
A[ ,1] = NA
Strvar.VecAr(vecar, A=A)
```

`styles` *Styles analysis (portfolio)*

Description

Perform Style analysis for single and multiple time periods

Usage

```
Styles(FUND
, IND
, W
, lower = NULL
, upper = NULL
, ...)

Multi.Styles(FUND
, IND
, W
, n_clust = 5
, lower = NULL
, upper = NULL
, ...)
```

Arguments

FUND	Vector. Benchmark investment fund
IND	Matrix of indices (returns)
W	Initial weghts to be assigned to the indices
n_clust	Number of time periods clusters for multi period analysis
lower	Lower boundary for the optimal weights (used in optim)
upper	Upper boundary for the optimal weights (used in optim)
...	Further arguments to or from other methods.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# load examples portfolio
data(ex_ptf)
# set initial weights
ww = c(0.09, rep(0.13,6))
# single period style analysis
Styles(FUND=ex_ptf[,1]
, IND=ex_ptf[,-1]
, W=ww
, lower=NULL
, upper=NULL)
# multi period style analysis
Multi.Styles(FUND=ex_ptf[,1]
, IND=ex_ptf[,-1]
, n_clust=5
, W=ww
, lower=NULL
, upper=NULL)
```

---

sumdens *Plot summary information*

Description

Plot summary information of a vector with its density

Usage

```
Sum.dens(x, ...)
```

Arguments

x	Vector. Input series.
...	further arguments for "plot" function

Author(s)

RAdamant Development Team <team@r-adamant.org>

sumvecar — *Summary for Vector AutoRegressive Models*

Description

Summary method for class 'VecAr'.

Usage

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

Arguments

`object`	Instance of class 'VecAr'.
`...`	Further arguments to or from other methods.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`VecAr`, `summary.mreg`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2)

# Get a summary
summary(mod)
```

swing — *Swing Index*

Description

Calculate Swing index (Technical Analysis)

Usage

```
Swing(Close, High, Low, Open, ret_cum = FALSE, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`Open`	Vector. Open price.
`ret_cum`	ret_cum
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`symlkup` *Lookup Stock Symbol from Yahoo!*

Description

Lookup stock symbols for which the symbol, name or description matches the input string value.

Usage

```
symbol.lookup(what = "")
```

Arguments

`what`	The string to search for.

Value

A matrix containing the top 10 stock symbols that match the input, with the following columns:

`Symbol`	The stock symbol.
`Name`	The stock name.
`Exchange`	The Exchange symbol.
`Type`	The Exchange Name.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`get.fs`

Examples

```
# lookup the symbol for Apple
symbol.lookup("Apple")
# Apple
APPLE = get.fs("AAPL", from=as.Date("2008-06-01"), to=as.Date("2011-04-01"));
```

---

tema — *Triple EMA*

Description

Compute multiple Triple EMA on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
tema(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
TEMA is a weighted combination of EMA: 3*EMA(X) - 3* EMA(EMA(X)) + EMA(EMA(EMA(X))).
Smoothing factor: lambda = 2/(win.size+1).

Value

A object of class 'ma' with attributes type = "TEMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
tema(x, 10)

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
tema(x, 40, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
tema(ex_fs, 15, plot=TRUE)

## End(Not run)
```

`themutil` *RAdamant Theme Management*

Description

Group of utility functions for themes management.
- Load themes definition from file (loadThemes).
- Return a theme definition given the theme name or id (getTheme).
- Return the current theme definition used by the plotting functions (getCurrentTheme).
- Set the theme to be used by the plotting functions (setCurrentTheme).
- Retrieve specific theme options/attributes from the current theme (getThemeAttr).
- Modify specific theme options/attributes of the current theme (getThemeAttr).

Usage

```
loadThemes(env = getOption("RAdamant")
, path = paste(library(help = RAdamant)$path, "themes", sep = "/"))

getTheme(which = 1, env = getOption("RAdamant"))

getCurrentTheme(env = getOption("RAdamant"))
setCurrentTheme(which = 1, env = getOption("RAdamant"))

getThemeAttr(what = NULL
, env = getOption("RAdamant")
, exact.match = FALSE)
setThemeAttr(..., env = getOption("RAdamant"))
```

Arguments

`env`	The environment where the themes definition are stored.
`path`	The file path where the theme definition files are stored.
`which`	Id or Name of the theme to be returned. Partial match on the theme name is allowed
`what`	The name of the theme attribute to be returned. Partial match is possible (depending on exact.match), in which case multiple attributes are returned.
`exact.match`	Logical. If TRUE, exact match of the attribute name is performed
`...`	Any theme attributes can be modified, using 'name = value' or by passing a list of such tagged values.

Details

Following is a list of all available theme attributes:

- col.main: Plot Title - Color.
- cex.main: Plot Title - Size.
- font.main: Plot Title - Font.
- col: Color palette for the plot. Recycled if necessary.
- ret.col: Color palette for plot of Returns.
- type: Plot type (line (l), points (p), line and points (o), histogram (h), ...). Recycled if necessary.
- pch: Points type. Recycled if necessary.
- cex: Points size. Recycled if necessary.
- lty: Line type. Recycled if necessary.
- lwd: Line width. Recycled if necessary.
- side: Axis scale side: 1 - use left y-axis scale; 2 - use right y-axis scale. Recycled if necessary.
- projection.col: Color palette for the projection plot. Recycled if necessary.
- projection.type: Projection type (line (l), points (p), line and points (o), histogram (h), ...). Recycled if necessary.
- projection.lty: Projection line type. Recycled if necessary.
- shade.col: Area Plot - Color palette for area plot. If a set of colors is provided, values will be interpolated.
- shade.transition: Area Plot - Gradient transition type: linear, exponential, quadratic, sqrt. Partial match is possible.
- shade.stripes: Area Plot - Number of stripes used to create the background gradient effect.
- shade.alpha: Area Plot - Alpha transparency (in the range [0, 1]). If a set of alphas is provided, values will be interpolated.
- shade.angle: Area Plot - Angle (degrees) for the shading pattern.
- shade.density: Area Plot - Density of the color filling (polygon equivalent parameter).
- shade.border: Area Plot - border color of the polygons.
- fg.col: Plot Window - Foreground background color.
- bg.col: Plot Area - Background colors used for the gradient. If a set of colors is provided, values will be interpolated.

- bg.alpha: Plot Area - Alpha transparency (in the range [0, 1]) used for the background. If a set of alphas is provided, values will be interpolated.
- bg.direction: Direction for the background color gradient: horisontal (down to up) or vertical (left to right).
- bg.transition: Gradient transition type: linear, exponential, quadratic, sqrt. Partial match is possible.
- bg.stripes: Number of stripes used to create the background gradient effect.
- plot.max.nrow: Define max number of rows for subplot matrix structure.
- plot.max.ncol: Define max number of columns subplot matrix structure.
- one.side.margin: Plot margins for plots with one y-axis.
- two.side.margin: Plot margins for plots with two y-axis.
- legend.pos: Legend - Position.
- legend.border: Legend - Border color.
- legend.bg: Legend - Background color. If a set of colors is provided, values will be interpolated.
- legend.alpha: Legend - Alpha transparency. If a set of alphas is provided, values will be interpolated.
- legend.cex: Legend - Font Size.
- legend.maxrows: Legend - Max number of rows.
- legend.direction: Legend - Direction for the background color gradient: horisontal (down to up) or vertical (left to right).
- legend.transition: Legend - Gradient transition type: linear, exponential, quadratic, sqrt. Partial match is possible.
- legend.stripes: Legend - Number of stripes used to create the background gradient effect.
- grid.col: Grid Lines - Color.
- grid.vlines: Grid Lines - Number of vertical lines.
- grid.hlines: Grid Lines - Number of horisontal lines.
- axis.col: Axis - Line Color.
- xlab.col: x-Axis - Tick labels color.
- xlab.cex: x-Axis - Label size as a percentage (see cex parameter from ?par).
- xlab.offset: x-Axis - Amount of down shift of the lables from the x-axis line as percentage of the y-range (diff(par('usr')[3:4]))
- x.ticks: x-Axis - Number of tickmarks and labels. If 'ALL', tickmarks and labels are plotted for each value.
- xlab.srt: x-Axis - Tick labels text rotation (degrees).
- xlab.fmt: x-Axis - Format style for the axis label.
- xlab.prefix: x-Axis - Prefix attached to the axis labels.
- xlab.suffix: x-Axis - Suffix attached to the axis labels.
- xtitle.col: x-Axis - Color to be used for the axis title.
- xtitle.srt: x-Axis - Text rotation for the title.
- xtitle.pos: x-Axis - Position of the title. Values in the range [0, 1] where 0 is left, and 1 is right (0.5 for the centre).

- xtitle.offset: x-Axis - Amount of down shift of the title from the x-axis line as percentage of the y-range (diff(par('usr')[3:4])).
- xtitle.cex: x-Axis - Size for the title.
- xtitle.font: x-Axis - Font for the title.
- ytitle.col: y-Axis - Color to be used for the axis title.
- ytitle.srt: y-Axis - Text rotation for the left title.
- ytitle.pos: y-Axis - Position of the left title. Values in the range [0, 1] where 0 is bottom, and 1 is top (0.5 for the middle).
- ytitle.offset: y-Axis - Amount of left shift of the title from the left y-axis line as percentage of the x-range (diff(par('usr')[1:2])).
- ytitle.cex: y-Axis - Size for the left title.
- ytitle.font: y-Axis - Font for the left title.
- ytitle2.col: y-Axis - Color to be used for the right axis title.
- ytitle2.srt: y-Axis - Text rotation for the right axis title.
- ytitle2.pos: y-Axis - Position of the right title. Values in the range [0, 1] where 0 is bottom, and 1 is top (0.5 for the middle).
- ytitle2.offset: y-Axis - Amount of right shift of the title from the right y-axis line as percentage of the x-range (diff(par('usr')[1:2])).
- ytitle2.cex: y-Axis - Size for the right title.
- ytitle2.font: y-Axis - Font for the right title.
- col3d: 3D Plot - Surface Color for the case when fill = "simple". See cplot3d.
- colmap: 3D Plot - Surface Colormap for the case when fill = "colormap" or "gradiend". See cplot3d.
- border: 3D Plot - the color of the line drawn around the surface facets. A value of 'NA' will disable the drawing of borders. See persp.
- theta: 3D Plot - Theta (Rotation).
- phi: 3D Plot - Phi (Azimuth).
- r: 3D Plot - Perspective. The distance of the eyepoint from the centre of the plotting box. See persp.
- d: 3D Plot - Perspective. Varies the strength of the perspective transformation. See persp.
- scale: 3D Plot - Scaling. See persp.
- expand: 3D Plot - Expansion factor applied to the 'z' coordinates. See persp.
- ltheta: 3D Plot - Theta angle (Rotation) for the illumination. See persp.
- lphi: 3D Plot - Phi angle (Azimuth) for the illumination. See persp.
- shade: 3D Plot - Controls the type of illumination. See persp.
- xtitle3d.col: 3D Plot x-Axis - Color for the axis title.
- xtitle3d.srt: 3D Plot x-Axis - Rotation for the axis title. If NULL, rotation is automatically calculated so that the title is parallel to the x-axis line.
- xtitle3d.pos: 3D Plot x-Axis - Position for the title. Values in the range [0, 1] where 0 is left, and 1 is right (0.5 for the centre).
- ytitle3d.col: 3D Plot y-Axis - Color for the axis title.
- ytitle3d.srt: 3D Plot y-Axis - Rotation for the axis title. If NULL, rotation is automatically calculated so that the title is parallel to the y-axis line.

- ytitle3d.pos: 3D Plot y-Axis - Position of the title. Values in the range [0, 1] where 0 is left, and 1 is right (0.5 for the centre).
- ztitle3d.col: 3D Plot z-Axis - Color for the axis title.
- ztitle3d.srt: 3D Plot z-Axis - Rotation for the axis title.
- ztitle3d.pos: 3D Plot z-Axis - Position of the title. Values in the range [0, 1] where 0 is bottom, and 1 is top (0.5 for the middle).
- box: Plot 3D Box - LOGICAL. If TRUE a box is plotted.
- box.col: Plot 3D Box - The color of the box lines.
- box.lty: Plot 3D Box - The line type used for drawing the box.
- box.lwd: Plot 3D Box - The line width used for drawing the box.
- box.half: Plot 3D Box - LOGICAL. If TRUE only the back side of the box is plotted.)
- xlab3d.srt: 3D Plot x-Axis - Tick labels text rotation (degrees).
- xgrid: 3D Plot grid - LOGICAL. If TRUE, grid lines across x-axis are plotted.
- ylab3d.srt: 3D Plot y-Axis - Tick labels text rotation (degrees).
- ygrid: 3D Plot grid - LOGICAL. If TRUE, grid lines across y-axis are plotted.
- zlab3d.srt: 3D Plot z-Axis - Tick labels text rotation (degrees).
- zgrid: 3D Plot grid - LOGICAL. If TRUE, grid lines across z-axis are plotted.
- ylab.col: y-Axis - Tick labels color.
- ylab.cex: y-Axis - Label size as a percentage (see cex parameter from ?par)
- ylab.offset: y-Axis - Amount of left/right shift of the lables from the y-axis line as percentage of the y-range (diff(par('usr')[1:2])).
- y.ticks: y-Axis - Number of tickmarks and labels. If 'ALL', tickmarks and labels are plotted for each value.
- ylab.srt: y-Axis - Tick labels text rotation (degrees).
- ylab.fmt: y-Axis - Format style for the axis label (left side).
- ylab.prefix: y-Axis - Prefix attached to the axis labels (left side).
- ylab.suffix: y-Axis - Suffix attached to the axis labels (left side).
- ylab2.fmt: y-Axis - Format style for the axis label (left right)
- ylab2.prefix: y-Axis - Prefix attached to the axis labels (right side).
- ylab2.suffix: y-Axis - Suffix attached to the axis labels (right side).
- zlab.col: z-Axis - Tick labels color.
- z.ticks: z-Axis - Number of tickmarks and labels. If 'ALL', tickmarks and labels are plotted for each value.
- zlab.prefix: z-Axis - Prefix attached to the axis labels.
- zlab.suffix: z-Axis - Suffix attached to the axis labels.
- zlab.fmt: z-Axis - Format style for the axis label.

Value

getTheme returns a list with all the attributes of the requested theme.

getCurrentTheme returns a list with all the attributes of the currently used theme.

getThemeAttr returns:
- A list of matched attributes if exact.match = FALSE. An empty list is returned if no matches are found.
- The value of the matched attribute if exact.match = TRUE. NULL is returned if no match is found.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Load all avaliable themes from the default directory
# Prints the all themes loaded in the form: Id) ThemeName
# 1) finance
# 2) vanilla
loadThemes();

# Retrieve the theme definition for the theme vanilla
getTheme("Van"); # Partial matching on the name.
# Equivalent to:
getTheme(2);

# Set the theme vanilla as the current theme for plotting
setCurrentTheme(2);
cplot(1:10);

# Change the color and type attributes of the current theme
setThemeAttr(col = c("blue", "red"), type = c("o", "l", "p"));
# Plot three series. Note how the two colors are recycled.
cplot(matrix(1:30, nrow=10, ncol=3));

# Look for all attributes containing the word "title"
getThemeAttr("title");
# Retrieve the current value for the attribute "col"
getThemeAttr("col", exact.match = TRUE);

# Restore all theme changes to default
setCurrentTheme(2);
```

---

thigh *True High oscillator*

Description

Compute True High oscillator (Technical Analysis)

Usage

```
thigh(Close, High = NULL, lag = 5, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

tirlev *Trione levels*

Description

Compute Trione levels (Technical Analysis)

Usage

```
tirLev(High, Low, Close, lag = 5, plot = FALSE, ...)
```

Arguments

`High`	Vector. High price.
`Low`	Vector. Low price.
`Close`	Vector. Close price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team <team@r-adamant.org>

---

tlow *True Low oscillator*

Description

Compute True Low oscillator (Technical Analysis)

Usage

```
tlow(Close, Low = NULL, lag = 5, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`tma` *Triangular Moving Averages*

Description

Compute multiple Triangular Moving Averages on the input data, one for each column of X[, i] and window size
win.size[j]

Usage

```
tma(X, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function Mmovav.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'ma' with attributes type = "TMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

Movav

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
tma(x, 15)
# compute moving average with multiple lags
tma(x, c(15,30))

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
tma(x, 30, plot = TRUE)
# multiple lags
tma(x, seq(5,50,10), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
tma(ex_fs, 30, plot=TRUE)
# multiple lags
tma(ex_fs, seq(5,50,10), plot=TRUE)

## End(Not run)
```

---

treynor *Treynor index*

Description

`Treynor`: Calculate Treynor index for a portfolio
`Treynor.Capm`: Get Treynor index from an object of class "Capm"

Usage

```
Treynor(PTF, ...)
## Default S3 method:
Treynor(PTF, PTF_M, rfr = 0, rf = NULL, ...)
## S3 method for class 'Capm'
Treynor(PTF, rfr = 0, ...)
```

Arguments

`PTF`	Input portfolio or an object of class "Capm"
`PTF_M`	Market/benchmark portfolio
`rfr`	risk free rate
`rf`	risk free asset
`...`	Further arguments to or from other methods

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`Jensen`, `Sharpe`, `Appraisal`

`trf` *(Average) True range*

Description

Compute (Average) True range (Technical Analysis)

Usage

```
trf(Close, High = NULL, Low = NULL, lag = 1,
average = TRUE, avg.lag = 14, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`average`	average
`avg.lag`	avg.lag
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

triangle	*Triangle window*

Description

Computes Triangle window of given length

Usage

```
triangle(N, normalized = TRUE)
```

Arguments

N	Window length.
normalized	Logical. If TRUE (default), window is normalised to have unitary norm.

Value

An object of the class 'Window'. It is a simple sequence of N samples of the Triangle window.

Author(s)

RAdamant Development Team <team@r-adamant.org>

Examples

```
# Generate a Normalised Triangle window of size 100
x = triangle(100)
# Plot the window
cplot(x
    , main = "Triangle Window"
    , legend = attr(x, "type")
    )
# Generate a non-normalised window
y = triangle(100, FALSE)
# Compare the two
cplot(cbind(x, y)
    , main = "Triangle Window"
    , legend = paste(attr(x, "type"), c("Normalised", "Not Normalised"))
    , type = c("l", "o")
    , xlab.srt = 0
    )
```

ttma *T3 EMA*

Description

Compute multiple T3 EMA on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
ttma(X, win.size = NROW(X), alpha = 0.7, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`alpha`	weight in the interval [0, 1]. (Default: 0.7).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
T3 EMA is a three times application of GDEMA: GDEMA(GDEMA(GDEMA(X, alpha), alpha), alpha).
Smoothing factor: lambda = 2/(win.size+1).

Value

A object of class 'ma' with attributes type = "TTMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`ema`, `gdema`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
ttma(x, 10)

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
ttma(x, 40, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
ttma(ex_fs, 15, plot=TRUE)

## End(Not run)
```

`typ` *Typical price*

Description

Compute Typical price (Technical Analysis)

Usage

```
tyP(Close, High, Low, plot = FALSE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`ulcer` *Ulcer index*

Description

Compute Ulcer index (Technical Analysis)

Usage

```
ulcer(X, lag, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`ultima` *Ultima oscillator*

Description

Compute Ultima oscillator (Technical Analysis)

Usage

```
ultima(Close
, High = NULL
, Low = NULL
, lag = 1
, win1 = 7
, win2 = 14
, win3 = 28
, plot = TRUE
, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`win1`	win1
`win2`	win2
`win3`	win3
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`univar` *Univariate analysis*

Description

Perform univariate analisys of the dependent variable Y versus each independent variable X, plotting the results

Usage

```
univar(Y
    , X
    , stress.period.idx = c()
    , Y.logit = FALSE
    , Y.logit.adj = 0.00005
    , theme.params = getCurrentTheme()
    , plot = TRUE
    , overrides = list(...)
    , ...
    )
```

Arguments

`Y`	The dependent variable. This must be a one column matrix.
`X`	Matrix containing all independent variables (one column per variable)
`stress.period.idx`	Vector of positions specifing the stress regime. If provided, the system will run a modified LS to capture the two regimes

`Y.logit`	Logical. If TRUE, the dependent variable is transformed using the Logit transform. Results are then transformed back using the inverse Logit. (Default: FALSE)
`Y.logit.adj`	Cut-off value. The range of the Y variable is restricted within the interval [Y.logit.adj, 1-Y.logit.adj] (Default: 0.00005)
`theme.params`	Theme parameters (Default: getCurrentTheme())
`plot`	Logical. If TRUE, results are plotted.
`overrides`	List of parameters to override the theme. Must match by name the parameters defined by the theme (Default: list(...)).
`...`	Alternative way to quickly override theme parameters.

Value

An object of class 'univar'. This is a list with the following components:

`Y.logit`	The input Y.logit parameter.
`stress.idx`	The input stress.period.idx parameter.
`model`	A list of NCOL(X) entries. Each entry is a linear model object (of class 'lm'): regression Y on the corresponding column of X.
`summary`	A summary data frame with columns [regressor, formula, eq, sigma.squared, adj.r.squared, pvalue].

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`plot.univar`, `print.univar`

Examples

```
# Load sample time series data
data(ex_ptf)
# Define the dependent variable
Y = ex_ptf[, 1, drop = FALSE];
# Define the independent variables
X = ex_ptf[, -1];
# Define x-axis labels
time.labels = paste("t[", 1:length(Y), "]", sep = "")
# Univar Analysis
univar(Y, X
    , xlabels = parse(text = time.labels)
    # Remove x-labels rotation
    , xlab.srt = 0
    # Set more space between x-labels and the x-axis line
        # (10% of diff(par("usr")[3:4]))
    , xlab.offset = 0.1
    # Set more space between x-title and the x-axis line
        # (20% of diff(par("usr")[3:4]))
    , xtitle.offset = 0.2
    # Only 4 tickmarks on the y-axis
    , y.ticks = 4
    )
```

var *Value at Risk*

Description

General VaR, computed on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
VaR(X, ...)

## Default S3 method:
VaR(X
    , p = 0.05
    , probf = c("Normal", "T-Student", "Cornish-Fisher", "GPD-POT")
    , df = max(4, (kurt(X)+3))
    , trsh = -hVaR(X)
    , ...
    )
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	Vector of probabilities (Default: 0.05)
`probf`	Probability distribution (see details). Case insensitive, partial matching is supported.
`df`	Degrees of freedom for the Student T distribution (Default: max(4, (kurt(X)+3)))
`trsh`	vector of NCOL(X) thresholds used to identify the tail data for the GPD-POT method
`...`	Additional parameters passed to the functions 'cofit' and 'gpd.VaR'.

Details

Accepted probability distributions:

- "Normal": Normal distribution.
- "T-Student": Student'T distribution.
- "Cornish-Fisher": Cornish-Fischer formula for quantiles estimation.
- "GPD-POT": Peak Over Threshold method, based on Generalised Pareto Distribution (EVT).

Value

A matrix length(p) by NCOL(X) of computed VaR values, based on the input distribution.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

gpd.VaR, mqt, cofit.

Examples

```
# Load sample asset data
data(ex_ptf);
# Compute VaR on multiple confidence levels (Normal)
VaR(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "Normal");

# T-Student
VaR(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "T");

# Extreme Value Theory (GPD)
VaR(ex_ptf, p = seq(0.03, 0.05, by = 0.01), probf = "GPD");
```

---

varptf *Portfolio Value at Risk*

Description

General VaR, computed for an input portfolio

Usage

```
VaRPtf(X, p = 0.05, weights = rep(1/NCOL(X), NCOL(X)), ...)
```

Arguments

X	Input matrix/sequence. Sequences are treated as one column matrices.
p	Vector of probabilities (Default: 0.05)
weights	Portfolio weigths (Default: rep(1/NCOL(X), NCOL(X)))
...	Additional parameters passed to the 'VaR' function

Value

A matrix length(p) by 1 of computed portfolio VaR values.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

VaR.

Examples

```
# Load sample asset data
data(ex_ptf);
# Compute VaR on multiple confidence levels (GPD)
VaRPtf(ex_ptf[, -1], p = seq(0.01, 0.05, by = 0.01), probf = "GPD");
```

vcmof — *Variable Chande Momentum Oscillator*

Description

Compute Variable Chande Momentum Oscillator (Technical Analysis)

Usage

```
vcmof(X, lag = 5, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

vecar — *Vector Autoregressive Model*

Description

Estimate Vector Autoregressive model

Usage

```
VecAr(X, ...)

## Default S3 method:
VecAr(X
     , ar.lags = 1:2
     , type = c("const", "trend", "constrend", "none")
     , regtype = "simple"
     , exog = NULL
     , ...
     )
```

Arguments

`X`	Input matrix of time series.
`ar.lags`	Vector indicating which lags should be included in the VAR model.
`type`	One of the following: • "const": an intercept term is included in the model; • "trend": a trend is included in the model; • "constrend": both intercept and trend are included in the model.
`regtype`	One of ("simple", "stepwise"). Controls the type of regression. See `mreg` for details.
`exog`	Matrix of exogenous variables to include in the model (Default: NULL).
`...`	Further arguments to or from other methods.

Value

An object of class "VecAr". This is a list containing the following elements:

`Model` The estimated model, instance of class 'mreg'.

`Info_Criteria`
One column matrix with components:

- Number of Observations
- Number of Variables
- Number of Parameters
- AIC information criteria
- BIC information criteria

The following attributes are attached to the object:

- Data: The full data model
- Xlag.names: Column names of the lagged components
- nser: The number of series modelled by the VAR
- nobs: The total number of observations (including NA) used for the model estimation (nobs = NROW(X)).
- npar: The number of model regressors entering the model
- exog.names: Column names of the exogenous variables
- Lag: The maximum order of the model
- Type: The input argument 'type'
- LogLike: List of NCOL(X) elements. Each entry id the Log-Likelihood of the corresponding OLS model

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

See Also

`mreg`, `Strvar.VecAr`, `fitted.VecAr`, `residuals.VecAr` , `coef.VecAr`, `summary.VecAr`, `estVar.VecAr`, `vcov.VecAr`.

Examples

```
# Collect series data
X = cbind(BJsales, BJsales.lead);

# Generate simple VAR(2) model
mod = VecAr(X, ar.lags = 1:2);
mod

# Only Lags 2 and 4 will enter the model
mod2 = VecAr(X, ar.lags = c(2, 4));
mod2

# Find the best fitting model, with no more than 4 lags
    # , including intercept and trend.
mod3 = VecAr(X
    # No more than 4 lags
    , ar.lags = 1:4
    # Stepwise model selection
    , regtype = "stepwise"
    # Include intercept and trend components
    , type = "constrend"
    # Constrain the maximum number of variables in the model
        # (3 for BJsales and 4 for BJsales.lead)
    , max.vars = c(3, 4)
    );
mod3
```

`vhff` *Vertical Horizontal Filter*

Description

Compute Vertical Horizontal Filter (Technical Analysis)

Usage

```
vhff(X, lag = 9, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

vidyaf — *Variable Index Dynamic Average*

Description

Compute Variable Index Dynamic Average (Technical Analysis)

Usage

```
vidyaf(X, lag = 5, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

vwma — *Volume Weighted Moving Averages*

Description

Compute multiple Volume Weighted Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
vwma(X, Vol = NULL, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`Vol`	Matrix of volumes (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
If X is a financial time series (class = 'fs'), and Vol = NULL then Vol = X[, 'Volume'] (Default: NULL).

Value

A object of class 'ma' with attributes type = "VWMA" and 'win.size' as from the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

sma

Examples

```
## load a dataset provided by RAdamant
data(ex_fs)
# extract Close price and Volume
x = ex_fs[,1]
Vol = ex_fs[,5]
# compute moving average with single lag
vwma(x, Vol, 10)
# compute moving average with multiple lags
vwma(x, Vol, c(10,20))

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
vwma(x, Vol, 15, plot = TRUE)
# multiple lags
vwma(x, Vol, c(10,20), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
vwma(ex_fs, Vol=NULL, 10, plot=TRUE, cex=0.7, rm.transient=FALSE)
# multiple lags
vwma(ex_fs, Vol=NULL, seq(5, 50, 10), plot=TRUE)


## End(Not run)
```

wad — *Williams Advance Decline*

Description

Compute Williams Advance Decline (Technical Analysis)

Usage

```
wad(Close
    , High = NULL
    , Low = NULL
    , lag = 5
    , na.rm = FALSE
    , plot = TRUE
    , ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`na.rm`	na.rm
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

weigevid — *Weight of Evidence*

Description

Calculate weight of evidence for a matrix with target variable

Usage

```
WeightEvid(data, target, nseg, missing = FALSE, na.replace=NULL, ...)
```

Arguments

`data`	MATRIX or DATA.FRAME. Input data.
`target`	Vector. Target variable in binary format 0-1
`nseg`	Integer of Vector. Number of segment to split the numerical variables.
`missing`	Logical. If TRUE missing values are considered in the calculation as a separate class.
`na.replace`	CHARACTER / NUMERIC. Value to replace missing. If NULL missing values are not considered in the computation.
`...`	Further parameter for the function `Factorise`

Value

A matrix containing the following columns:

- "Variable"
- "Segment"
- "Obs"
- "PC.Obs"
- "Good"
- "PC.Good"
- "Bad"
- "Pc.Bad"
- "Rate"
- "Weight.Evidence"
- "Info.Value.Within"
- "Info.Value"

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# load example data set "credit"
data(ex_credit)
# calculate weight of evidence
input = ex_credit[ ,-1]
target = ex_credit[ ,1]
woe = WeightEvid(data=input, target=target, nseg = 2:3, missing=FALSE)
# quick look of the results got from WeightEvid
woe
```

wghtmreg *Extract Model Weights for (Multi)-Regression object*

Description

Generic method for extracting model weights from object of classes 'reg' and 'mreg'.

Usage

```
## S3 method for class 'reg'
weights(object, na.rm = FALSE, ...)

## S3 method for class 'mreg'
weights(object, na.rm = FALSE, ...)
```

Arguments

object	Instance of class 'reg'/'mreg'.
na.rm	Logical. If TRUE, NA records are removed.
...	Further arguments to or from other methods.

Value

One of the following:

- class 'mreg': A matrix containing all model weights, one column for each model.
- class 'reg': A matrix containing the model specific weights.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

mreg.

Examples

```
# Generate some random data
N = 50;
sigma = 1;
X1 = cumsum(rnorm(N));
X2 = cumsum(rnorm(N));

# Define a linear model
Y1 = 1.5 + X1 + 2*X2 + rnorm(N, sd = sigma);
Y2 = -2 + 1.2*X1 -X2 + rnorm(N, sd = sigma);
# Add some NA
Y2[1:3] = NA

# Define Weights (Equal weights for the first model,
    # linear weights for the second)
```

```
W = cbind(1/N, 1:N);
# Run Multi-Regression
mod = mreg(Y = cbind(Y1, Y2)
            , X = cbind(X1, X2)
            , plot = FALSE
            , weights = W);

# Extract all coefficients
weights(mod)
# Removes entries where NA are present
weights(mod, na.rm = TRUE)

# Extract coefficients from the second model
weights(mod[[2]])
# Removes entries where NA are present
weights(mod[[2]], na.rm = TRUE)
```

`whes` *Weighted Historical Expected Shortfall*

Description

Compute Weighted historical ES on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
whES(X, p = 0.05, lambda = 0.9, centered = FALSE)
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	Vector of probabilities (Default: 0.05).
`lambda`	Controls the exponential window lambda^((NROW(X)-1):0) (Default: 0.9).
`centered`	Logical. If TRUE, input data are standardised.

Value

A matrix length(p) by NCOL(X) of computed historical weighted ES.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load sample Financial series data
data(ex_fs);
# Compute Historical Weighted ES (5% confidence level) on 1-day Returns
whES(Ret(ex_fs));

# Generate some random data
X = cbind(rnorm(1000), rnorm(1000, sd = 2))
```

```
# Compute multiple Historical Weighted ES (1%, 2.5%, 5% confidence levels)
whES(X, p = c(1, 2.5, 5)/100);
```

`whvar` *Weighted Historical Value at Risk*

Description

Compute Weighted historical VaR on each column of the input matrix. If input is a Financial Series object (class 'fs'), then 'Close' data are processed.

Usage

```
whVaR(X, p = 0.05, lambda = 0.9, centered = FALSE)
```

Arguments

`X`	Input matrix/sequence. Sequences are treated as one column matrices.
`p`	Vector of probabilities (Default: 0.05).
`lambda`	Controls the exponential window lambda^((NROW(X)-1):0) (Default: 0.9).
`centered`	Logical. If TRUE, input data are standardised.

Value

A matrix length(p) by NCOL(X) of computed historical weighted VaR.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
# Load sample Financial series data
data(ex_fs);
# Compute Historical Weighted VaR (5% confidence level) on 1-day Returns
whVaR(Ret(ex_fs));

# Generate some random data
X = cbind(rnorm(1000), rnorm(1000, sd = 2))
# Compute multiple Historical Weighted VaR (1%, 2.5%, 5% confidence levels)
whVaR(X, p = c(1, 2.5, 5)/100);
```

`wildavg` *Wilder Moving Average*

Description

Compute Wilder Moving Average (Technical Analysis)

Usage

```
wildAvg(X, lag = 5, plot = FALSE, ...)
```

Arguments

`X`	X
`lag`	lag
`plot`	plot
`...`	...

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

`wildsum` *Wilder Summation*

Description

Compute Wilder Summation (Technical Analysis)

Usage

```
wildSum(x, lag = 5)
```

Arguments

`x`	x
`lag`	lag

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

wma *Weighted Moving Averages*

Description

Compute multiple Weighted Moving Averages on the input data, one for each column of X[, i] and window size
win.size[j]

Usage

```
wma(X, win.size = 10, plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: 10).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by the function Mmovav.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.

Value

A object of class 'ma' with attributes type = "WMA" and 'win.size' as given by the corresponding input parameter:
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

Examples

```
## load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
wma(x, 10)
# compute moving average with multiple lags
wma(x, c(10,20))

## Not run:
# refine results of moving average
setCurrentTheme(1)
# single lag
```

```
wma(x, 30, plot = TRUE)
# multiple lags
wma(x, seq(5,50,10), plot=TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(2)
data(ex_fs)
# single lag
wma(ex_fs, 30, plot=TRUE)
# multiple lags
wma(ex_fs, seq(5,50,10), plot=TRUE)

## End(Not run)
```

`wro` *Williams R*

Description

Compute Williams R (Technical Analysis)

Usage

```
wro(Close, High = NULL, Low = NULL, lag = 5, plot = TRUE, ...)
```

Arguments

`Close`	Vector. Close price.
`High`	Vector. High price.
`Low`	Vector. Low price.
`lag`	Integer. Number of lag periods.
`plot`	Logical. If TRUE plot is returned.
`...`	Further arguments to or from other methods.

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

zind — *Z index*

Description

Compute the Z-score of X (Standardize each column of X)

Usage

```
Zind(x, sigma = 1, mi = 2)
```

Arguments

`x`	x
`sigma`	sigma
`mi`	mi

Note

TO BE COMPLETED

Author(s)

RAdamant Development Team `<team@r-adamant.org>`

zlma — *Zero lag Moving Average*

Description

Compute multiple Zero-Lag Exponential Moving Averages on the input data, one for each column of X[, i] and window size win.size[j].

Usage

```
zlma(X, win.size = NROW(X), plot = FALSE, ...)
```

Arguments

`X`	Matrix of data series (one column per variable).
`win.size`	vector of moving average window sizes (lags) to be applied on the data X. (Default: NROW(X)).
`plot`	Logical. Return plot.
`...`	Additional parameters accepted by function ema.

Details

For financial time series (class = 'fs'), only 'Close' column is processed.
ZLMA is a combination of EMA: EMA(X) + EMA(X - EMA(X)).

Value

A object of class 'ma' with attributes type = "EMAT" and lambda = 2/(win.size+1):
- matrix of size NROW(X) by NCOL(X)*length(win.size) where each column is the moving average of length win.size[i] of the corresponding column of X.

Author(s)

RAdamant Development Team <team@r-adamant.org>

See Also

ema

Examples

```
# load a dataset provided by R
data(EuStockMarkets)
# extract sample (log) time series
x = log(EuStockMarkets[500:800,2, drop=FALSE])
# compute moving average with single lag
zlma(x, 10)

## Not run:
# refine results of moving average
setCurrentTheme(2)
# single lag
zlma(x, 15, plot = TRUE)

# calculate moving average for an object of class "fs"
setCurrentTheme(1)
data(ex_fs)
# single lag
zlma(ex_fs, 30, plot=TRUE)

## End(Not run)
```

---

zscore *Z Score*

Description

Compute the Z-score of X (Standardize each column of X)

Usage

```
Zscore(X, means = NULL, sigma = NULL)
```

Arguments

X	Matrix of data series (one column per variable)
means	Mean value
sigma	Standard deviation

Value

Matrix of standardised variables

Author(s)

RAdamant Development Team <team@r-adamant.org>

Index