

Package ‘TTR’

August 28, 2009

Type Package

Title Technical Trading Rules

Version 0.20-1

Revision 88

Date 2009-08-27

Author Joshua Ulrich

Maintainer Joshua Ulrich <josh.m.ulrich@gmail.com>

Depends xts (>= 0.6.4)

Enhances quantmod

Description Functions and data to construct technical trading rules with R.

License GPL-3

Repository CRAN

Date/Publication 2009-08-28 20:18:23

R topics documented:

adjRatios	2
ADX	3
aroon	4
ATR	6
bollingerBands	7
CCI	9
chaikinAD	10
chaikinVolatility	11
changes	12
CLV	13
CMF	14

CMO	15
DonchianChannel	16
DPO	17
EMV	19
GMMA	20
KST	21
MACD	23
MFI	24
MovingAverages	25
OBV	28
RSI	29
runFun	30
SAR	32
stochastics	33
TDI	35
TRIX	36
TTR	38
ttrc	39
TTRtools	40
VHF	41
volatility	42
WebData	44
williamsAD	46
WPR	47
ZigZag	48
Index	50

adjRatios

Split and dividend adjustment ratios

Description

Create split and dividend adjustment ratio vectors.

Usage

```
adjRatios(splits, dividends, close)
```

Arguments

splits	Split series that is coercible to xts.
dividends	Dividend series that is coercible to xts.
close	Close price series that is coercible to xts.

Details

If only `splits` is provided, the resulting object will only have as many observations as `splits`.

If `splits` and `close` are provided, the resulting object will have as many observations as `max(NROW(splits), NROW(close))`.

`close` is required if `dividends` is provided.

Value

A `xts` object containing the columns:

<code>Split</code>	The split adjustment ratio.
<code>Div</code>	The dividend adjustment ratio.

Author(s)

Joshua Ulrich

ADX

Welles Wilder's Directional Movement Index

Description

Directional Movement Index; developed by J. Welles Wilder.

Usage

```
ADX(HLC, n=14, maType, ...)
```

Arguments

<code>HLC</code>	Object that is coercible to <code>xts</code> or matrix and contains High-Low-Close prices.
<code>n</code>	Number of periods to use for DX calculation (not ADX calculation).
<code>maType</code>	A function or a string naming the function to be called.
<code>...</code>	Other arguments to be passed to the <code>maType</code> function.

Details

The `DIp/DIn` (positive/negative) is the percentage of the true range that is up/down.

Value

A object of the same class as `HLC` or a matrix (if `try.xts` fails) containing the columns:

<code>DIp</code>	The positive Direction Index.
<code>DIn</code>	The negative Direction Index.
<code>DX</code>	The Direction Index.
<code>ADX</code>	The Average Direction Index (trend strength).

Note

A buy/sell signal is generated when the +/-DI crosses up over the -/+DI, when the DX/ADX signals a strong trend. A high/low DX signals a strong/weak trend. DX is usually smoothed with a moving average (i.e. the ADX).

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/DI.htm>
<http://www.fmlabs.com/reference/DX.htm>
<http://www.fmlabs.com/reference/ADX.htm>
<http://www.fmlabs.com/reference/ADXR.htm>
<http://www.equis.com/Customr/Resources/TAAZ/Default.aspx?c=3&p=49>
<http://linnsoft.com/tour/techind/dirInd.htm>
<http://linnsoft.com/tour/techind/adx.htm>
<http://linnsoft.com/tour/techind/adxr.htm>
http://stockcharts.com/education/IndicatorAnalysis/indic_ADX.html

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. The DX calculation uses [ATR](#). See [aroon](#), [CCI](#), [TDI](#), [VHF](#), [GMMA](#) for other indicators that measure trend direction/strength.

Examples

```
data(ttrc)
dmi.adx <- ADX(ttrc[,c("High", "Low", "Close")])
```

aroon

Aroon

Description

The Aroon indicator attempts to identify starting trends. The indicator consists of up and down lines, which measure how long it has been since the highest high/lowest low has occurred in the last *n* periods. Developed by Tushar Chande in 1995.

Usage

```
aroon(HL, n=20)
```

Arguments

HL	Object that is coercible to xts or matrix and contains either a High-Low price series, or a Close price series.
n	Number of periods to use in the calculation.

Details

Aroon up (down) is the elapsed time, expressed as a percentage, between today and the highest (lowest) price in the last *n* periods. If today's price is a new high (low) Aroon up (down) will be 100. Each subsequent period without another new high (low) causes Aroon up (down) to decrease by $(1 / n) \times 100$.

Value

A object of the same class as HL or a matrix (if `try.xts` fails) containing the columns:

<code>aroonUp</code>	The Aroon up indicator.
<code>aroonDn</code>	The Aroon down indicator.
<code>oscillator</code>	The Aroon oscillator (<code>aroonUp - aroonDn</code>).

Note

If High-Low prices are given, the function calculates the max/min using the high/low prices. Otherwise the function calculates the max/min of the single series.

Up (down) trends are indicated when the `aroonUp(Dn)` is between 70 and 100. Strong trends are indicated when when the `aroonUp(Dn)` is above 70 while the `aroonDn(Up)` is below 30. Also, crossovers may be useful.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/Aroon.htm>
<http://www.fmlabs.com/reference/AroonOscillator.htm>
<http://www.linnsoft.com/tour/techind/aroon.htm>
<http://stockcharts.com/education/IndicatorAnalysis/indic-Aroon.htm>

See Also

See [CCI](#), [ADX](#), [TDI](#), [VHF](#), [GMMA](#) for other indicators that measure trend direction/strength.

Examples

```
## Get Data and Indicator ##
data(ttrc)
trend <- aroon( ttrc[,c("High", "Low")], n=20 )
```

ATR

*True Range / Average True Range***Description**

True range (TR) is a measure of volatility of a High-Low-Close series; average true range (ATR) is a Welles Wilder's style moving average of the TR. Developed by J. Welles Wilder in 1978.

Usage

```
ATR(HLC, n=14, maType, ...)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices.
n	Number of periods for moving average.
maType	A function or a string naming the function to be called.
...	Other arguments to be passed to the maType function.

Details

TR incorporates yesterday's close in the calculation (high minus low). E.g. if yesterday's close was higher than today's high, then the TR would equal yesterday's close minus today's low.

The ATR is a component of the Welles Wilder Directional Movement Index (DX, ADX).

Value

A object of the same class as HLC or a matrix (if `try.xts` fails) containing the columns:

tr	The true range of the series.
atr	The average (as specified by ma) true range of the series.
true.high	The true high of the series.
true.low	The true low of the series.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/TR.htm>
<http://www.fmlabs.com/reference/ATR.htm>
<http://www.equis.com/Custom/Resources/TAAZ/?c=3&p=35>
<http://www.linnsoft.com/tour/techind/trueRange.htm>
http://stockcharts.com/education/IndicatorAnalysis/indic_ATR.html

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [DX](#), which uses true range. See [chaikinVolatility](#) for another volatility measure.

Examples

```
data(ttrc)
atr <- ATR(ttrc[,c("High", "Low", "Close")], n=14)
```

`bollingerBands` *Bollinger Bands*

Description

Bollinger Bands are a way to compare a security's volatility and price levels over a period of time. Developed by John Bollinger.

Usage

```
BBands(HLC, n=20, maType, sd=2, ...)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices. If only a univariate series is given, it will be used. See details.
n	Number of periods for moving average.
maType	A function or a string naming the function to be called.
sd	The number of standard deviations to use.
...	Other arguments to be passed to the maType function.

Details

Bollinger Bands consist of three lines:

The middle band is generally a 20-period SMA of the typical price ($(\text{high} + \text{low} + \text{close})/3$). The upper and lower bands are `sd` standard deviations (generally 2) above and below the MA.

The middle band is usually calculated using the typical price, but if a univariate series (e.g. Close, Weighted Close, Median Price, etc.) is provided, it will be used instead.

Value

A object of the same class as HLC or a matrix (if `try.xts` fails) containing the columns:

<code>dn</code>	The lower Bollinger Band.
<code>ma</code>	The middle Moving Average (see notes).
<code>up</code>	The upper Bollinger Band.
<code>pctB</code>	The %B calculation.

Note

Using any moving average other than SMA will result in inconsistencies between the moving average calculation and the standard deviation calculation. Since, by definition, a rolling standard deviation uses a simple moving average.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/Bollinger.htm>

<http://www.fmlabs.com/reference/BollingerWidth.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/?c=3&p=36>

<http://www.linnsoft.com/tour/techind/bb.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_Bbands.html

http://stockcharts.com/education/IndicatorAnalysis/indic_BBWidth.htm

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section.

Examples

```
## The examples below show the differences between using a
## High-Low-Close series, and just a close series when
## calculating Bollinger Bands.
data(ttrc)
bbands.HLC <- BBands( ttrc[,c("High","Low","Close")] )
bbands.close <- BBands( ttrc[, "Close"] )
```

Description

The Commodity Channel Index (CCI) attempts to identify starting and ending trends.

Usage

```
CCI(HLC, n=20, maType, c=0.015, ...)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices. If only a univariate series is given, it will be used. See details.
n	Number of periods for moving average.
maType	A function or a string naming the function to be called.
c	Constant to apply to the mean deviation.
...	Other arguments to be passed to the maType function.

Details

CCI relates the current price and the average of price over n periods. The CCI usually falls in a channel of -100 to 100. A basic CCI trading system is: Buy (sell) if CCI rises above 100 (falls below -100) and sell (buy) when it falls below 100 (rises above -100).

CCI is usually calculated using the typical price, but if a univariate series (e.g. Close, Weighted Close, Median Price, etc.) is provided, it will be used instead.

Value

A object of the same class as HLC or a vector (if `try.xts` fails) containing the CCI values.

Note

If HLC is a High-Low-Close matrix, then typical price will be calculated. If HLC is a vector, then those values will be used instead of the typical price.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/CCI.htm>

<http://www.equis.com/Custom/Resouces/TAAZ/?c=3&p=42>

<http://www.linnsoft.com/tour/techind/cci.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_CCI.html

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [aroon](#), [ADX](#), [TDI](#), [VHF](#), [GMMMA](#) for other indicators that measure trend direction/strength.

Examples

```
data(ttrc)
cci <- CCI(ttrc[,c("High", "Low", "Close")])
```

chaikinAD

Chaikin Accumulation / Distribution

Description

The Chaikin Accumulation / Distribution (AD) line is a measure of the money flowing into or out of a security. It is similar to On Balance Volume (OBV). Developed by Marc Chaikin.

Usage

```
chaikinAD(HLC, volume)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices.
volume	Vector or matrix of volume observations corresponding to the HLC object.

Details

The AD line is similar to OBV; the difference is that OBV sums volume multiplied by +/- 1 if the close is higher/lower than the previous close, while the AD line multiplies volume by the close location value (CLV).

Value

A object of the same class as HLC and volume or a vector (if `try.xts` fails) containing the accumulation / distribution values.

Note

The Accumulation/Distribution Line is interpreted by looking for a divergence in the direction of the indicator relative to price.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/AccumDist.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/?c=3\&p=27>

http://www.linnsoft.com/tour/techind/acc_dis.htm

http://stockcharts.com/education/IndicatorAnalysis/indic_AccumDistLine.html

See Also

See [OBV](#), and [CLV](#).

Examples

```
data(ttrc)
ad <- chaikinAD(ttrc[,c("High", "Low", "Close")], ttrc[, "Volume"])
```

chaikinVolatility *Chaikin Volatility*

Description

Chaikin Volatility measures the rate of change of the security's trading range. Developed by Marc Chaikin.

Usage

```
chaikinVolatility(HL, n=10, maType, ...)
```

Arguments

HL	Object that is coercible to xts or matrix and contains High-Low prices.
n	Number of periods for moving average.
maType	A function or a string naming the function to be called.
...	Other arguments to be passed to the maType function.

Details

The Chaikin Volatility indicator defines volatility as an increase in the difference between the high and low.

Value

A object of the same class as HL or a vector (if `try.xts` fails) containing the Chaikin Volatility values.

Note

A rapid increase in Chaikin Volatility indicates an approaching bottom. A slow decrease in Chaikin Volatility indicates an approaching top.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/ChaikinVolatility.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/Default.aspx?c=3&p=120>

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [TR](#) for another volatility measure.

Examples

```
data(ttrc)
volatility <- chaikinVolatility(ttrc[,c("High", "Low")])
```

changes

Rate of Change / Momentum

Description

Calculate the (rate of) change of a series over n periods.

Usage

```
ROC(x, n=1, type=c("continuous", "discrete"), na.pad=TRUE)
momentum(x, n=1, na.pad=TRUE)
```

Arguments

<code>x</code>	Price, volume, etc. series that is coercible to xts or matrix.
<code>n</code>	Number of periods to use.
<code>type</code>	Compounding type; either "continuous" (the default) or "discrete".
<code>na.pad</code>	Should periods prior to <code>n</code> be appended? Default is TRUE.

Details

The ROC indicator provides the percentage difference of a series over two observations, while the momentum indicator simply provides the difference.

Value

A object of the same class as `x` or a vector (if `try.xts` fails) containing the rate-of-change (or return) values for ROC or a vector containing the differenced price series for momentum.

Author(s)

Joshua Ulrich

Examples

```
data(ttrc)
roc <- ROC(ttrc[, "Close"])
mom <- momentum(ttrc[, "Close"])
```

CLV

Close Location Value

Description

The Close Location Value (CLV) relates the day's close to its trading range.

Usage

```
CLV(HLC)
```

Arguments

<code>HLC</code>	Object that is coercible to xts or matrix and contains High-Low-Close prices.
------------------	---

Details

The CLV will fall in a range of -1 to +1. If the CLV is +/-1, the close is at the high/low; if the CLV is 0, the close is directly between the high and low.

Value

A object of the same class as HLC or a vector (if `try.xts` fails) containing the Close Location Values of a High-Low-Close price series.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

http://stockcharts.com/education/IndicatorAnalysis/indic_AccumDistLine.html

See Also

See [chaikinAD](#), which uses CLV.

Examples

```
data(ttrc)
clv <- CLV(ttrc[,c("High", "Low", "Close")])
```

CMF

Chaikin Money Flow

Description

Chaikin Money Flow compares total volume over the last n time periods to total volume times the Close Location Value (CLV) over the last n time periods. Developed by Marc Chaikin.

Usage

```
CMF(HLC, volume, n = 20)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices.
volume	Vector or matrix of volume observations corresponding to the HLC object.
n	Number of periods to use.

Details

Chaikin Money Flow is calculated by taking dividing the sum of the Chaikin Accumulation / Distribution line over the past n periods by the sum of volume over the past n periods.

Value

A object of the same class as `HLC` and `volume` or a vector (if `try.xts` fails) containing the Chaikin Money Flow values.

Note

When Chaikin Money Flow is above/below +/- 0.25 it is a bullish/bearish signal. If Chaikin Money Flow remains below zero while the price is rising, it indicates a probable reversal.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/ChaikinMoneyFlow.htm>

<http://www.linnsoft.com/tour/techind/cmfm.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_ChaikinMoneyFlow1.html

See Also

See [CLV](#), and [chaikinAD](#).

Examples

```
data(ttrc)
cmf <- CMF(ttrc[,c("High", "Low", "Close")], ttrc[, "Volume"])
```

CMO

Chande Momentum Oscillator

Description

The Chande Momentum Oscillator (CMO) is a modified RSI. Developed by Tushar S. Chande.

Usage

```
CMO(x, n=14)
```

Arguments

`x` Price, volume, etc. series that is coercible to `xts` or matrix.
`n` Number of periods to use.

Details

The CMO divides the total movement by the net movement ($([up - down] / [up + down])$), where RSI divides the upward movement by the net movement ($up / [up + down]$).

Value

A object of the same class as `x` or a vector (if `try.xts` fails) containing Chande Momentum Oscillator values.

Note

There are several ways to interpret the CMO:

- (1) Values over/under +/- 50 indicate overbought/oversold conditions.
- (2) High CMO values indicate strong trends.
- (3) When the CMO crosses above/below a moving average of the CMO, it is a buy/sell signal.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/CMO.htm>

See Also

See [RSI](#).

Examples

```
data(ttrc)
cmo <- CMO(ttrc[, "Close"])
```

DonchianChannel *Donchian Channel*

Description

Donchian Channels were created by Richard Donchian and were used to generate buy and sell signals for the Turtle Trading system.

Usage

```
DonchianChannel(HL, n=10)
```

Arguments

HL	Object that is coercible to xts or matrix and contains High-Low prices.
n	Number of periods for moving average.

Details

Donchian Channels consist of two (sometimes three) lines:

The top line is the highest high of the past n periods. The bottom line is the lowest low of the past n periods. The middle line is the average of the top and bottom lines.

Value

A object of the same class as HL or a matrix (if `try.xts` fails) containing the columns:

high	The highest high series.
mid	The average of high and low.
low	The lowest low series.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.linnsoft.com/tour/techind/donch.htm>

See Also

See [BBands](#).

Examples

```
data(ttrc)
dc <- DonchianChannel( ttrc[,c("High", "Low")] )
```

DPO

De-Trended Price Oscillator

Description

The Detrended Price Oscillator (DPO) removes the trend in prices - or other series - by subtracting a moving average of the price from the price.

Usage

```
DPO(x, n=10, maType, shift=n/2+1, percent=FALSE, ...)
```

Arguments

<code>x</code>	Price, volume, etc. series that is coercible to xts or matrix.
<code>n</code>	Number of periods for moving average.
<code>maType</code>	A function or a string naming the function to be called.
<code>shift</code>	The number of periods to shift the moving average.
<code>percent</code>	logical; if <code>TRUE</code> , the percentage difference between the slow and fast moving averages is returned, otherwise the difference between the respective averages is returned.
<code>...</code>	Other arguments to be passed to the <code>maType</code> function.

Details

The Detrended Price shows cycles and overbought / oversold conditions. Note the calculation shifts the results `shift` periods, so the last `shift` periods will be zero.

Value

A object of the same class as `x` or a vector (if `try.xts` fails) containing the DPO values.

Note

As stated above, the DPO can be used on any univariate series, not just price.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/DPO.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/?c=3&p=48>

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [MACD](#) for a general oscillator.

Examples

```
data(ttrc)
priceDPO <- DPO(ttrc[, "Close"])
volumeDPO <- DPO(ttrc[, "Volume"])
```

 EMV

Arms' Ease of Movement Value

Description

Arms' Ease of Movement Value (EMV) emphasizes days where the security moves easily and minimizes days where the security does not move easily. Developed by Richard W. Arms, Jr.

Usage

```
EMV(HL, volume, n=9, maType, vol.divisor=10000, ...)
```

Arguments

HL	Object that is coercible to xts or matrix and contains High-Low prices.
volume	Vector or matrix of volume observations corresponding to the HL object.
n	Number of periods for moving average.
maType	A function or a string naming the function to be called.
vol.divisor	An increment to make the results larger and easier to work with.
...	Other arguments to be passed to the maType function.

Details

The EMV is calculated by dividing the midpoint ($(\text{high} + \text{low})/2$) move by the 'Box Ratio' (volume divided by the high minus low).

Value

A object of the same class as HL and volume or a matrix (if `try.xts` fails) containing the columns:

emv	The ease of movement values.
emvMA	The smoothed (as specified by ma) ease of movement values.

Note

A buy/sell signal is generated when the EMV crosses above/below zero. When the EMV hovers around zero, there are small price movements and/or high volume, and the price is not moving easily.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/ArmsEMV.htm>

<http://www.equis.com/Custom/Resouces/TAAZ/?c=3&p=51>

<http://linnsoft.com/tour/techind/arms.htm>

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section.

Examples

```
data(ttrc)
emv <- EMV(ttrc[,c("High","Low")], ttrc[, "Volume"])
```

GMMA

Guppy Multiple Moving Averages

Description

Calculate the Guppy Multiple Moving Average of a series.

Usage

```
GMMA(x, short=c(3,5,8,10,12,15),
      long=c(30,35,40,45,50,60), maType)
```

Arguments

x	Price, volume, etc. series that is coercible to xts or matrix.
short	Vector of short-term periods.
long	Vector of long-term periods.
maType	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.

Details

The Guppy Multiple Moving Average signals a changing trend when the `short` and `long` groups of moving averages intersect. An up/down trend exists when the short/long-term moving averages are greater than the long/short-term averages.

Value

A object of the same class as `x` or `price` or a vector (if `try.xts` fails) containing the Guppy Multiple Moving Average.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.investopedia.com/terms/g/guppy-multiple-moving-average.asp>

See Also

See [aroon](#), [CCI](#), [ADX](#), [VHF](#), [TDI](#) for other indicators that measure trend direction/strength.

Examples

```
data(ttrc)
gmma <- GMMA(ttrc[, "Close"])
```

KST

Know Sure Thing

Description

The Know Sure Thing (KST) is a smooth, summed, rate of change indicator. Developed by Martin Pring.

Usage

```
KST(price, n=c(10,10,10,15), nROC=c(10,15,20,30), nSig=9,
     maType, wts=1:NROW(n), ...)
```

Arguments

<code>price</code>	Price series that is coercible to <code>xts</code> or <code>matrix</code> .
<code>n</code>	A vector of the number of periods to use in the MA calculations.
<code>nROC</code>	A vector of the number of periods to use in the ROC calculations.
<code>nSig</code>	The number of periods to use for the KST signal line.
<code>maType</code>	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.

`wts` A vector the same length as `n`, of the weight for each period (need not sum to one).

`...` Other arguments to be passed to the `maType` function in case (1) above.

Details

For each day (week, month, etc.), the KST calculates the ROC over several periods. Those ROCs are smoothed using the given moving averages, then multiplied by their respective weighting values. The resulting values are summed for each day (month, week, etc.).

Value

A object of the same class as `price` or a vector (if `try.xts` fails) containing the Know Sure Thing values.

Note

The KST indicates bullish/bearish momentum as it crosses above/below its moving average. Because the KST tends to lead price action, look for trend confirmation in the price.

The default arguments are for the daily KST. There is also the Long-Term KST, with arguments: `n=c(9, 12, 18, 24)` - where the periods are months, not days - and the moving average periods are 6, 6, 6, and 9 months, respectively.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.pring.com/index.html>
http://www.pring.com/movieweb/daily_kst.htm
<http://www.pring.com/articles/article28.htm>
http://www.pring.com/movieweb/KST_MCM.htm

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [ROC](#) for the rate-of-change function. See [MACD](#) for a generic oscillator.

Examples

```
data(ttrc)
kst <- KST(ttrc[, "Close"])

kst4MA <- KST(ttrc[, "Close"],
  maType=list(list(SMA), list(EMA), list(DEMA), list(WMA)))
```

MACD

*MACD Oscillator***Description**

The MACD was developed by Gerald Appel and is probably the most popular price oscillator. The MACD function documented in this page compares a fast moving average (MA) of a series with a slow MA of the same series. It can be used as a generic oscillator for any univariate series, not only price.

Usage

```
MACD(x, nFast=12, nSlow=26, nSig=9, maType, percent=TRUE, ...)
```

Arguments

<code>x</code>	Object that is coercible to xts or matrix; usually price, but can be volume, etc.
<code>nFast</code>	Number of periods for fast moving average.
<code>nSlow</code>	Number of periods for slow moving average.
<code>nSig</code>	Number of periods for signal moving average.
<code>maType</code>	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
<code>percent</code>	logical; if TRUE, the percentage difference between the fast and slow moving averages is returned, otherwise the difference between the respective averages is returned.
<code>...</code>	Other arguments to be passed to the <code>maType</code> function in case (1) above.

Details

The MACD function either subtracts the fast MA from the slow MA, or finds the rate of change between the fast MA and the slow MA.

Value

A object of the same class as `x` or a matrix (if `try.xts` fails) containing the columns:

<code>macd</code>	The price (volume, etc.) oscillator.
<code>signal</code>	The oscillator signal line (a moving average of the oscillator).

Note

The MACD is a special case of the general oscillator applied to price. The MACD can be used as a general oscillator applied to any series. Time periods for the MACD are often given as 26 and 12, but the function originally used exponential constants of 0.075 and 0.15, which are closer to 25.6667 and 12.3333 periods.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/MACD.htm>

<http://www.fmlabs.com/reference/PriceOscillator.htm>

<http://www.fmlabs.com/reference/PriceOscillatorPct.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_MACD1.html

http://stockcharts.com/education/IndicatorAnalysis/indic_priceOscillator.html

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section.

Examples

```
data(ttrc)

macd <- MACD( ttrc["Close"], 12, 26, 9, maType="EMA" )
macd2 <- MACD( ttrc["Close"], 12, 26, 9,
               maType=list(list(SMA), list(EMA, wilder=TRUE), list(SMA)) )
```

MFI

Money Flow Index

Description

The MFI is a ratio of positive and negative money flow over time.

Usage

```
MFI(HLC, volume, n=14)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices. If only a univariate series is given, it will be used. See details.
volume	Vector or matrix of volume observations corresponding to HLC object.
n	Number of periods to use.

Details

Money Flow (MF) is the product of price and volume. Positive/negative MF occur when today's price is higher/lower than yesterday's price. The MFI is calculated by dividing positive MF by negative MF for the past n periods. It is then scaled between 0 and 100.

MFI is usually calculated using the typical price, but if a univariate series (e.g. Close, Weighted Close, Median Price, etc.) is provided, it will be used instead.

Value

A object of the same class as `HLC` and `volume` or a vector (if `try.xts` fails) containing the MFI values.

Note

Divergence between MFI and price can be indicative of a reversal. In addition, values above/below 80/20 indicate market tops/bottoms.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/default.htm?url=MoneyFlowIndex.htm>

<http://www.linnsoft.com/tour/techind/mfi.htm>

http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:money_flow_index_mfi

See Also

See [OBV](#) and [CMF](#).

Examples

```
data(ttrc)
mfi <- MFI(ttrc[,c("High","Low","Close")], ttrc[, "Volume"])
```

MovingAverages

Moving Averages

Description

Calculate various moving averages (MA) of a series.

Usage

```

SMA(x, n=10)
EMA(x, n=10, wilder=FALSE, ratio=NULL)
WMA(x, n=10, wts=1:n)
DEMA(x, n=10)
EVWMA(price, volume, n=10)
ZLEMA(x, n=10, ratio=NULL)

```

Arguments

<code>x</code>	Price, volume, etc. series that is coercible to xts or matrix.
<code>price</code>	Price series that is coercible to xts or matrix.
<code>volume</code>	Volume series that is coercible to xts or matrix, that corresponds to price series, or a constant. See Notes.
<code>n</code>	Number of periods to average over.
<code>wts</code>	Vector of weights. Length of <code>wts</code> vector must equal the length of <code>x</code> , or <code>n</code> (the default).
<code>wilder</code>	logical; if TRUE, a Welles Wilder type EMA will be calculated; see notes.
<code>ratio</code>	A smoothing/decay ratio to use (overrides <code>wilder</code> in EMA)

Details

SMA calculates the arithmetic mean of the series over the past `n` observations.

EMA calculates an exponentially-weighted mean, giving more weight to recent observations. See **Warning** section below.

WMA is similar to an EMA, but with linear weighting if the length of `wts` is equal to `n`. If the length of `wts` is equal to the length of `x`, the WMA will use the values of `wts` as weights.

DEMA is calculated as: $DEMA = 2 * EMA(x, n) - EMA(EMA(x, n), n)$.

EVWMA uses volume to define the period of the MA.

ZLEMA is similar to an EMA, as it gives more weight to recent observations, but attempts to remove lag by subtracting data prior to $(n-1)/2$ periods (default) to minimize the cumulative effect.

Value

A object of the same class as `x` or `price` or a vector (if `try.xts` fails) containing the columns:

SMA	Simple moving average.
EMA	Exponential moving average.
WMA	Weighted moving average.
DEMA	Double-exponential moving average.
EVWMA	Elastic, volume-weighted moving average.
ZLEMA	Zero lag exponential moving average.

Warning

Some indicators (e.g. EMA, DEMA, EVWMA, etc.) are calculated using the indicators' own previous values, and are therefore unstable in the short-term. As the indicator receives more data, its output becomes more stable. See example below.

Note

For EMA, `wilder=FALSE` (the default) uses an exponential smoothing ratio of $2/(n+1)$, while `wilder=TRUE` uses Welles Wilder's exponential smoothing ratio of $1/n$.

Since WMA can accept a weight vector of length equal to the length of `x` or of length `n`, it can be used as a regular weighted moving average (in the case `wt.s=1:n`) or as a moving average weighted by volume, another indicator, etc.

For EVWMA, if `volume` is a series, `n` should be chosen so the sum of the volume for `n` periods approximates the total number of outstanding shares for the security being averaged. If `volume` is a constant, it should represent the total number of outstanding shares for the security being averaged.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/ExpMA.htm>
<http://www.fmlabs.com/reference/WeightedMA.htm>
<http://www.fmlabs.com/reference/DEMA.htm>
<http://linnsoft.com/tour/techind/evwma.htm>
<http://www.fmlabs.com/reference/ZeroLagExpMA.htm>

See Also

See [wilderSum](#), which is used in calculating a Welles Wilder type MA.

Examples

```
data(ttrc)
  ema.20 <- EMA(ttrc[, "Close"], 20)
  sma.20 <- SMA(ttrc[, "Close"], 20)
  dema.20 <- DEMA(ttrc[, "Close"], 20)
  evwma.20 <- EVWMA(ttrc[, "Close"], ttrc[, "Volume"], 20)
  zlema.20 <- ZLEMA(ttrc[, "Close"], 20)

## Example of short-term instability of EMA
## (and other indicators mentioned above)
x <- rnorm(100)
tail(EMA(x[90:100], 10), 1)
tail(EMA(x[70:100], 10), 1)
tail(EMA(x[50:100], 10), 1)
tail(EMA(x[30:100], 10), 1)
```

```
tail( EMA(x[10:100],10), 1 )  
tail( EMA(x[ 1:100],10), 1 )
```

OBV

On Balance Volume (OBV)

Description

On Balance Volume (OBV) is a measure of the money flowing into or out of a security. It is similar to Chaikin Accumulation / Distribution.

Usage

```
OBV(price, volume)
```

Arguments

`price` Price series that is coercible to xts or matrix.
`volume` Volume series that is coercible to xts or matrix, that corresponds to price object.

Details

OBV is calculated by adding (subtracting) each day's volume to a running cumulative total when the security's price closes higher (lower).

Value

A object of the same class as `price` and `volume` or a vector (if `try.xts` fails) containing the OBV values.

Note

OBV is usually compared with the price chart of the underlying security to look for divergences/confirmation.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/OBV.htm>

<http://www.equis.com/Custom/Resourses/TAAZ?c=3\&p=82>

<http://linnsoft.com/tour/techind/obVol.htm>

<http://stockcharts.com/education/IndicatorAnalysis/indic-obv.htm>

See Also

See [chaikinAD](#).

Examples

```
data(ttrc)
obv <- OBV(ttrc[, "Close"], ttrc[, "Volume"])
```

RSI

Relative Strength Index

Description

The Relative Strength Index (RSI) calculates a ratio of the recent upward price movements to the absolute price movement. Developed by J. Welles Wilder.

Usage

```
RSI(price, n=14, maType, ...)
```

Arguments

<code>price</code>	Price series that is coercible to xts or matrix.
<code>n</code>	Number of periods for moving averages.
<code>maType</code>	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
<code>...</code>	Other arguments to be passed to the <code>maType</code> function in case (1) above.

Details

The RSI calculation is $RSI = 100 - 100 / (1 + RS)$, where *RS* is the smoothed ratio of 'average gains over 'average' losses. The 'averages' aren't true averages, since they're divided by the value of *n* not the number of gain/loss periods.

Value

A object of the same class as `price` or a vector (if `try.xts` fails) containing the RSI values.

Note

The RSI is usually interpreted as an overbought/oversold (over 70 / below 30) indicator. Divergence with price may also be useful. For example, if price is making new highs/lows, but RSI is not, it could indicate a reversal.

You can calculate a stochastic RSI by using the function [stoch](#) on RSI values.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/RSI.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/?c=3&p=100>

<http://linnsoft.com/tour/techind/rsi.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_RSI.html

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [CMO](#) for a variation on RSI.

Examples

```
data(ttrc)
price <- ttrc[, "Close"]

# Default case
rsi <- RSI(price)

# Case of one 'maType' for both MAs
rsiMA1 <- RSI(price, n=14, maType="WMA", wts=ttrc[, "Volume"])

# Case of two different 'maType's for both MAs
rsiMA2 <- RSI(price, n=14, maType=list(maUp=list(EMA, ratio=1/5),
                                       maDown=list(WMA, wts=1:10)))
```

runFun

Analysis of Running/Rolling/Moving Windows

Description

Various functions to analyze data over a moving window of periods.

Usage

```
runSum(x, n=10, cumulative=FALSE)
runMin(x, n=10, cumulative=FALSE)
runMax(x, n=10, cumulative=FALSE)
runMean(x, n=10, cumulative=FALSE)
runMedian(x, n=10, non.unique="mean", cumulative=FALSE)
runCov(x, y, n=10, use="all.obs", sample=TRUE, cumulative=FALSE)
```

```
runCor(x, y, n=10, use="all.obs", sample=TRUE, cumulative=FALSE)
runVar(x, y=NULL, n=10, sample=TRUE, cumulative=FALSE)
runSD(x, n=10, sample=TRUE, cumulative=FALSE)
runMAD(x, n=10, center=NULL, stat="median",
       constant=1.4826, non.unique="mean", cumulative=FALSE)
```

Arguments

<code>x</code>	Object coercible to xts or matrix.
<code>y</code>	Object coercible to xts or matrix.
<code>n</code>	Number of periods to use in the window or, if <code>cumulative=TRUE</code> , the number of observations to use before the first result is returned.
<code>cumulative</code>	Logical, use from-inception calculation?
<code>sample</code>	Logical, sample covariance if <code>TRUE</code> (denominator of $n-1$)
<code>use</code>	Only "all.obs" currently implemented.
<code>non.unique</code>	One of 'mean', 'max', or 'min'; which compute their respective statistics for the two middle values of even-sized samples.
<code>center</code>	The values to use as the measure of central tendency, around which to calculate deviations. The default (<code>NULL</code>) uses the median.
<code>stat</code>	Statistic to calculate, one of 'median' or 'mean' (e.g. median absolute deviation or mean absolute deviation, respectively.)
<code>constant</code>	Scale factor applied to approximate the standard deviation.

Value

A object of the same class as `x` and `y` or a vector (if `try.xts` fails).

<code>runSum</code>	returns sums over a n-period moving window.
<code>runMin</code>	returns minimums over a n-period moving window.
<code>runMax</code>	returns maximums over a n-period moving window.
<code>runMean</code>	returns means over a n-period moving window.
<code>runMedian</code>	returns medians over a n-period moving window.
<code>runCov</code>	returns covariances over a n-period moving window.
<code>runCor</code>	returns correlations over a n-period moving window.
<code>runVar</code>	returns variances over a n-period moving window.
<code>runSD</code>	returns standard deviations over a n-period moving window.
<code>runMAD</code>	returns median/mean absolute deviations over a n-period moving window.

Author(s)

Joshua Ulrich

SAR

Parabolic Stop-and-Reverse

Description

The Parabolic Stop-and-Reverse calculates a trailing stop. Developed by J. Welles Wilder.

Usage

```
SAR(HL, accel=c(0.02, 0.2))
```

Arguments

HL	Object that is coercible to xts or matrix and contains High-Low prices.
accel	accel[1]: Acceleration factor. accel[2]: Maximum acceleration factor.

Details

The calculation for the SAR is quite complex. See the URLs in the references section for calculation notes.

The SAR assumes that you are always in the market, and calculates the Stop And Reverse point when you would close a long position and open a short position or vice versa.

Value

A object of the same class as HL or a vector (if `try.xts` fails) containing the Parabolic Stop and Reverse values.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:
<http://www.linnsoft.com/tour/techind/sar.htm>
<http://www.fmlabs.com/reference/SAR.htm>

See Also

See [ATR](#) and [ADX](#), which were also developed by Welles Wilder.

Examples

```
data(ttrc)
sar <- SAR(ttrc[,c("High", "Low")])
```

Description

The stochastic oscillator is a momentum indicator that relates the location of each day's close relative to the high/low range over the past *n* periods. Developed by George C. Lane in the late 1950s. The SMI relates the close to the midpoint of the high/low range. Developed by William Blau in 1993.

Usage

```
stoch(HLC, nFastK=14, nFastD=3, nSlowD=3, maType, bounded=TRUE, smooth=1, ...)
```

```
SMI(HLC, n=13, nFast=2, nSlow=25, nSig=9, maType, bounded=TRUE, ...)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices. If only a univariate series is given, it will be used. See details.
n	Number of periods to use.
nFastK	Number of periods for fast %K (i.e. the number of past periods to use).
nFastD	Number of periods for fast %D (i.e. the number smoothing periods to apply to fast %K).
nSlowD	Number of periods for slow %D (i.e. the number smoothing periods to apply to fast %D).
smooth	Number of internal smoothing periods to be applied before calculating FastK. See Details.
nFast	Number of periods for initial smoothing.
nSlow	Number of periods for double smoothing.
nSig	Number of periods for signal line.
maType	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
bounded	Logical, should current period's values be used in the calculation?
...	Other arguments to be passed to the maType function in case (1) above.

Details

If a High-Low-Close series is provided, the indicator is calculated using the high/low values. If a vector is provided, the calculation only uses that series. This allows stochastics to be calculated for: (1) series that have no HLC definition (e.g. foreign exchange), and (2) stochastic indicators (e.g. stochastic RSI - see examples).

The `smooth` argument is the number of periods of internal smoothing to apply to the differences in the high-low-close range before calculating Fast K. Thanks to Stanley Neo for the suggestion.

Value

A object of the same class as `HLC` or a matrix (if `try.xts` fails) containing the columns:

<code>fastK</code>	Stochastic Fast %K
<code>fastD</code>	Stochastic Fast %D
<code>slowD</code>	Stochastic Slow %D
<code>SMI</code>	Stochastic Momentum Index
<code>signal</code>	Stochastic Momentum Index signal line

Note

The calculation for William's %R is similar to that of stochastics' fast %K.

The stochastic oscillator and SMI calculate relative value of the close versus the high/low range and the midpoint of the high/low range, respectively.

The stochastic oscillator and the stochastic momentum index are interpreted similarly. Readings below 20 (above 80) are considered oversold (overbought). However, readings below 20 (above 80) are not necessarily bearish (bullish). Lane believed some of the best sell (buy) signals occurred when the oscillator moved from overbought (oversold) back below 80 (above 20).

For the stochastic oscillator, buy (sell) signals can also be given when %K crosses above (below) %D. Crossover signals are quite frequent however, which may result in whipsaws.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document these indicators:

Stochastic Oscillator:

<http://www.fmlabs.com/reference/StochasticOscillator.htm>

<http://www.equis.com/Custom/Resouces/TAAZ?c=3&p=106>

<http://linnsoft.com/tour/techind/stoc.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_stochasticOscillator.html

SMI:

<http://www.fmlabs.com/reference/default.htm?url=SMI.htm>

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note Warning section. See [WPR](#) to compare it's results to fast %K.

Examples

```

data(ttrc)
stochOSC <- stoch(ttrc[,c("High", "Low", "Close")])
stochWPR <- WPR(ttrc[,c("High", "Low", "Close")])

plot(tail(stochOSC[, "fastK"], 100), type="l",
     main="Fast %K and Williams %R", ylab="",
     ylim=range(cbind(stochOSC, stochWPR), na.rm=TRUE) )
lines(tail(stochWPR, 100), col="blue")
lines(tail(1-stochWPR, 100), col="red", lty="dashed")

stoch2MA <- stoch( ttrc[,c("High", "Low", "Close")],
                  maType=list(list(SMA), list(EMA, wilder=TRUE), list(SMA)) )

SMI3MA <- SMI(ttrc[,c("High", "Low", "Close")],
              maType=list(list(SMA), list(EMA, wilder=TRUE), list(SMA)) )

stochRSI <- stoch( RSI(ttrc[, "Close"]) )

```

TDI

Trend Detection Index

Description

The Trend Detection Index (TDI) attempts to identify starting and ending trends. Developed by M. H. Pee.

Usage

```
TDI(price, n=20, multiple=2)
```

Arguments

price	Price series that is coercible to xts or matrix.
n	Number of periods to use.
multiple	Multiple used to calculate (2).

Details

The TDI is the (1) absolute value of the n-day sum of the n-day momentum, minus the quantity of (2) multiple*n-day sum of the absolute value of the n-day momentum, minus (3) n-day sum of the absolute value of the n-day momentum.

I.e. $TDI = (1) - [(2) - (3)]$

The direction indicator is the sum of the n-day momentum over the last n days.

See URL in references section for further details.

Value

A object of the same class as `price` or a matrix (if `try.xts` fails) containing the columns:

<code>tdi</code>	The Trend Detection Index.
<code>di</code>	The Direction Indicator.

Note

Positive/negative TDI values signal a trend/consolidation. A positive/ negative direction indicator signals a up/down trend. I.e. buy if the TDI and the direction indicator are positive, and sell if the TDI is positive while the direction indicator is negative.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:
<http://www.linnsoft.com/tour/techind/tdi.htm>

See Also

See [aroon](#), [CCI](#), [ADX](#), [VHF](#), [GMMA](#) for other indicators that measure trend direction/strength.

Examples

```
data(ttrc)
tdi <- TDI(ttrc[, "Close"], n=30)
```

TRIX

Triple Smoothed Exponential Oscillator

Description

The TRIX indicator calculates the rate of change of a triple exponential moving average. Developed by Jack K. Hutson.

Usage

```
TRIX(price, n=20, nSig=9, maType, percent=TRUE, ...)
```

Arguments

<code>price</code>	Price series that is coercible to <code>xts</code> or <code>matrix</code> .
<code>n</code>	Number of periods for moving average.
<code>nSig</code>	Number of periods for signal line moving average.
<code>maType</code>	Either: (1) A function or a string naming the function to be called, or (2) a <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
<code>percent</code>	logical; if <code>TRUE</code> , the rate of change is calculated using the <code>ROC</code> function, otherwise the <code>momentum</code> function is used.
<code>...</code>	Other arguments to be passed to the <code>maType</code> function in case (1) above.

Details

The TRIX is calculated as follows:
 $3MA = MA(MA(MA(price)))$
 $trix = 100 * [3MA(t) / 3MA(t-1) - 1]$

Value

A object of the same class as `price` or a vector (if `try.xts` fails) containing the TRIX values.

Note

Buy/sell signals are generated when the TRIX crosses above/below zero. A nine-period EMA of the TRIX is used as a default signal line. Buy/sell signals are generated when the TRIX crosses above/below the signal line and is also above/below zero.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:
<http://www.fmlabs.com/reference/default.htm?url=TRIX.htm>
<http://www.equis.com/Custom/Resouces/TAAZ/?c=3&p=114>
<http://www.linnsoft.com/tour/techind/trix.htm>
http://stockcharts.com/education/IndicatorAnalysis/indic_trix.htm

See Also

See [EMA](#), [SMA](#), etc. for moving average options; and note `Warning` section.

Examples

```
data(ttrc)
trix <- TRIX(ttrc[, "Close"])
trix4 <- TRIX(ttrc[, "Close"],
             maType=list(list(SMA), list(EMA, wilder=TRUE), list(SMA), list(DEMA)))
```

TTR

Functions to create Technical Trading Rules (TTR)

Description

This package contains many of the most popular technical analysis functions, as well as functions to retrieve U.S. stock symbols, and data from Yahoo Finance.

Details

Package: TTR
Type: Package
Version: 0.20-1
Date: 2009-08-27
License: GPL Version 3 or later.

Users will probably be most interested in the following functions:

[ADX](#)
[BBands](#)
[changes](#)
[MovingAverages](#)
[MACD](#)
[RSI](#)
[runFun](#)
[stoch](#)
[WebData](#)

Author(s)

Joshua Ulrich

Maintainer: Joshua Ulrich

References

The following sites were used to code/document this package:

<http://www.fmlabs.com/reference/default.htm>

<http://www.equis.com/Custom/Resourses/TAAZ/?p=0>

<http://www.linnsoft.com/tour/technicalindicators.htm>

<http://stockcharts.com/education/IndicatorAnalysis/>

Examples

```
data(ttrc)

# Bollinger Bands
bbands <- BBands( ttrc[,c("High","Low","Close")] )

# Directional Movement Index
adx <- ADX(ttrc[,c("High","Low","Close")])

# Moving Averages
ema <- EMA(ttrc[, "Close"], n=20)
sma <- SMA(ttrc[, "Close"], n=20)

# MACD
macd <- MACD( ttrc[, "Close"] )

# RSI
rsi <- RSI(ttrc[, "Close"])

# Stochastics
stochOsc <- stoch(ttrc[,c("High","Low","Close")])

### Note: you must have a working internet connection
### for the examples below to work!

# Fetch U.S. symbols from the internet
nyseSymbols <- stockSymbols("NYSE")

# Fetch Yahoo! Finance data from the internet
ibm <- getYahooData("IBM", 19990404, 20050607)
```

ttrc

Technical Trading Rule Composite data

Description

Historical Open, High, Low, Close, and Volume data for the periods January 2, 1985 to December 31, 2006. Randomly generated.

Usage

```
data(ttrc)
```

Format

The format is:

```

Date:      Class 'Date'  5480 5481 5482 5485 5486 ...
Open:     num           3.18 3.09 3.11 3.09 3.10 ...
High:     num           3.18 3.15 3.12 3.12 3.12 ...
Low:      num           3.08 3.09 3.08 3.07 3.08 ...
Close:    num           3.08 3.11 3.09 3.10 3.11 ...
Volume:   num           1870906 3099506 2274157 2086758 2166348 ...

```

Details

These data do not represent an actual security. They are provided so examples do not necessitate an internet connection.

Source

Randomly generated.

Examples

```

data(ttrc)
plot(tail(ttrc[, "Close"], 100), type="l")

```

TTRtools

Miscellaneous Tools

Description

Various functions that may be useful in designing technical trading rules.

Usage

```

growth(price, signals, ...)
lags(x, n=1)
wilderSum(x, n=10)

```

Arguments

<code>price</code>	Price series that is coercible to xts or matrix.
<code>signals</code>	Signals to use (defaults to vector of ones). Use '0' for no position, '1' for long position, and '-1' for short position.
<code>x</code>	Object that is coercible to xts or matrix.
<code>n</code>	Number of periods to use.
<code>...</code>	Further arguments to be passed from or to other methods.

Details

`growth` calculates the growth of an investment using given prices and signals.

`lags` calculates the lags of a given series.

`wilderSum` calculates a Welles Wilder style weighted sum.

Value

`growth` returns a vector of the growth of the investment.
`lags` returns a matrix of lagged values of the original vector.
`wilderSum` returns a vector of weighted sums.

Note

In `growth` you can specify the number of periods and type of compounding to use when calculating returns of the price series via the `'...'` argument.

Author(s)

Joshua Ulrich

VHF

Vertical Horizontal Filter

Description

The Vertical Horizontal Filter (VHF) attempts to identify starting and ending trends. Developed by Adam White.

Usage

```
VHF(price, n=28)
```

Arguments

<code>price</code>	Object that is coercible to <code>xts</code> or <code>matrix</code> and contains a Close price series, or a High-Low-Close price series.
<code>n</code>	Number of periods to use.

Details

The VHF is calculated by subtracting the `n`-period lowest low from the `n`-period highest high and dividing that result by the `n`-period rolling sum of the close price changes.

Value

A object of the same class as `price` or a vector (if `try.xts` fails) containing the VHF values.

Note

If Close prices are given, the function calculates the max/min using only those prices (the default). If HLC prices are given, the function calculates the max/min using the high/low prices (added for flexibility).

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.equis.com/Custom/Resourses/TAAZ?c=3&p=119>

See Also

See [aroon](#), [CCI](#), [ADX](#), [TDI](#), [GMMMA](#) for other indicators that measure trend direction/strength.

Examples

```
data(ttrc)
vhf.close <- VHF(ttrc[, "Close"])
vhf.hilow <- VHF(ttrc[, c("High", "Low", "Close")])
```

volatility

Volatility

Description

Selected volatility estimators/indicators; various authors.

Usage

```
volatility(OHLC, n=10, calc="close", N=260, ...)
```

Arguments

OHLC	Object that is coercible to xts or matrix and contains Open-High-Low-Close prices.
n	Number of periods for the volatility estimate.
calc	The calculation (type) of estimator to use.
N	Number of periods per year.
...	Arguments to be passed to/from other methods.

Details

Close-to-Close Volatility (`close`):
Historical volatility calculation using close-to-close prices.

OHLC Volatility: Garman and Klass (`garman.klass`):
The Garman and Klass estimator for estimating historical volatility assumes Brownian motion with zero drift and no opening jumps (i.e. the opening = close of the previous period). This estimator is 7.4 times more efficient than the close-to-close estimator.

High-Low Volatility: Parkinson (`parkinson`):
The Parkinson formula for estimating the historical volatility of an underlying based on high and low prices.

OHLC Volatility: Rogers and Satchell (`rogers.satchell`):
The Roger and Satchell historical volatility estimator allows for non-zero drift, but assumed no opening jump.

OHLC Volatility: Garman and Klass - Yang and Zhang (`gk.yz`):
This estimator is a modified version of the Garman and Klass estimator that allows for opening gaps.

OHLC Volatility: Yang and Zhang (`yang.zhang`):
The Yang and Zhang historical volatility estimator has minimum estimation error, and is independent of drift and opening gaps. It can be interpreted as a weighted average of the Rogers and Satchell estimator, the close-open volatility, and the open-close volatility.

Value

A object of the same class as OHLC or a vector (if `try.xts` fails) containing the chosen volatility estimator values.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document these indicators:

Close-to-Close Volatility (`close`):
<http://www.sitmo.com/eq/172>
OHLC Volatility: Garman Klass (`garman.klass`):

```

http://www.sitmo.com/eq/402
High-Low Volatility: Parkinson (parkinson):
http://www.sitmo.com/eq/173
OHLC Volatility: Rogers Satchell (rogers.satchell):
http://www.sitmo.com/eq/414
OHLC Volatility: Garman Klass - Yang Zhang (gk.yz):
http://www.sitmo.com/eq/409
OHLC Volatility: Yang Zhang (yang.zhang):
http://www.sitmo.com/eq/417

```

See Also

See [TR](#) and [chaikinVolatility](#) for other volatility measures.

Examples

```

data(ttrc)
ohlc <- ttrc[,c("Open", "High", "Low", "Close")]
vClose <- volatility(ohlc, calc="close")
vGK <- volatility(ohlc, calc="garman")
vParkinson <- volatility(ohlc, calc="parkinson")
vRS <- volatility(ohlc, calc="rogers")

```

WebData

Fetch Internet Data

Description

Get investment data from the internet.

Usage

```

getYahooData(symbol, start, end, freq="daily", type="price",
              adjust=TRUE, quiet=FALSE)
stockSymbols(exchange=c("AMEX", "NASDAQ", "NYSE"),
             sort.by=c("Exchange", "Symbol"), quiet=FALSE)

```

Arguments

symbol	Yahoo! Finance instrument symbol.
start	Numeric; first date of desired data, in YYYYMMDD format. Default is first date of series.
end	Numeric; last date of desired data, in YYYYMMDD format. Default is last date of series.
freq	Desired data frequency. One of "daily", "weekly", "monthly".

<code>type</code>	Type of data to return. One of "price", or "split". <code>type="split"</code> will return both split and dividend data.
<code>adjust</code>	Logical; if TRUE, the Open, High, Low, and Close prices will be adjusted for dividends and splits, and Volume will be adjusted for dividends.
<code>quiet</code>	Logical; if TRUE, status messages will be printed to the console.
<code>exchange</code>	Character vector of exchange names on which desired instrument symbols are traded.
<code>sort.by</code>	Character vector of columns by which returned data will be sorted. Must be one or more of "Name", "Symbol", "Market.Cap", or "Exchange".

Details

`getYahooData` fetches individual stock data from the Yahoo! Finance website. It also adjusts price for splits and dividends, and volume for splits.

`stockSymbols` fetches instrument symbols from the nasdaq.com website, and adjusts the symbols to be compatible with the Yahoo! Finance website.

Value

`getYahooData` returns an xts object containing the columns:

Date	Trade date, in CCYYMMDD format.
Open	Open price.
High	High price.
Low	Low price.
Close	Close price.
Volume	Volume.

`stockSymbols` returns a character vector containing all the listed symbols for the given exchanges.

Note

The symbols returned by `stockSymbols` may not be in the format necessary to retrieve data using `getYahooData`.

`getYahooData` has only been tested on daily data. It isn't known if the function correctly adjusts data for any other frequency.

Author(s)

Joshua Ulrich

Examples

```
### Note: you must have a working internet
### connection for these examples to work!
ibm <- getYahooData("IBM", 19990404, 20050607)

nyse.symbols <- stockSymbols("NYSE")
```

`williamsAD`*Williams Accumulation / Distribution*

Description

The Williams Accumulation / Distribution (AD) line is a measure of market momentum. Developed by Larry Williams.

Usage

```
williamsAD(HLC)
```

Arguments

HLC Object that is coercible to xts or matrix and contains High-Low-Close prices.

Details

The Williams AD line differs from OBV and chaikinAD in that it doesn't take volume into account.

Value

A object of the same class as HLC or a vector (if `try.xts` fails) containing the accumulation / distribution values.

Note

The Accumulation/Distribution Line is interpreted by looking for a divergence in the direction of the indicator relative to price.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/WilliamsAD.htm>

<http://www.equis.com/Custommer/Resources/TAAZ/?c=3&p=125>

See Also

See [OBV](#), [chaikinAD](#), and [ATR](#).

Examples

```
data(ttrc)
ad <- williamsAD(ttrc[,c("High", "Low", "Close")])
```

WPR

William's %R

Description

William's % R.

Usage

```
WPR(HLC, n=14)
```

Arguments

HLC	Object that is coercible to xts or matrix and contains High-Low-Close prices. If only a univariate series is given, it will be used. See details.
n	Number of periods to use.

Details

If an High-Low-Close series is provided, the indicator is calculated using the high/low values. If a vector is provided, the calculation only uses that series.

Value

A object of the same class as HLC or a vector (if `try.xts` fails) containing the William's %R values.

Note

The William's %R calculation is similar to stochastics' fast %K.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/WilliamsR.htm>

<http://www.equis.com/Custom/Resourses/TAAZ?c=3&p=126>

<http://linnsoft.com/tour/techind/willR.htm>

http://stockcharts.com/education/IndicatorAnalysis/indic_williamsR.html

See Also

See [stoch](#).

Examples

```

data(ttrc)
stochOsc <- stoch(ttrc[,c("High", "Low", "Close")])
stochWPR<- WPR(ttrc[,c("High", "Low", "Close")])

plot(tail(stochOsc["fastK"], 100), type="l",
      main="Fast %K and Williams %R", ylab="",
      ylim=range(cbind(stochOsc, stochWPR), na.rm=TRUE) )
lines(tail(stochWPR, 100), col="blue")
lines(tail(1-stochWPR, 100), col="red", lty="dashed")

```

ZigZag

Zig Zag

Description

Zig Zag highlights trends by removing price changes smaller than change and interpolating lines between the extreme points.

Usage

```
ZigZag( HL, change=10, percent=TRUE, retrace=FALSE, lastExtreme=TRUE )
```

Arguments

HL	Object that is coercible to xts or matrix and contains either a High-Low price series, or a Close price series.
change	Minimum price movement, either in dollars or percent (see percent).
percent	Use percentage or dollar change?
retrace	Is change a retracement of the previous move, or an absolute change from peak to trough?
lastExtreme	If the extreme price is the same over multiple periods, should the extreme price be the first or last observation?

Details

The Zig Zag is non-predictive. The purpose of the Zig Zag is filter noise and make chart patterns clearer. It's more a visual tool than an indicator.

Value

A object of the same class as HL or a vector (if `try.xts` fails) containing the Zig Zag indicator.

Note

If High-Low prices are given, the function calculates the max/min using the high/low prices. Otherwise the function calculates the max/min of the single series.

Author(s)

Joshua Ulrich

References

The following site(s) were used to code/document this indicator:

<http://www.fmlabs.com/reference/default.htm?url=ZigZag.htm>

<http://www.linnsoft.com/tour/techind/zigzag.htm>

<http://www.linnsoft.com/tour/techind/zigosc.htm>

<http://www.equis.com/Customer/Resources/TAAZ/?c=3\&p=127>

http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:zigzag

Examples

```
## Get Data and Indicator ##  
data(ttrc)  
zz <- ZigZag( ttrc[,c("High", "Low")], change=20 )
```

Index

*Topic **datasets**

ttrc, 39

*Topic **package**

TTR, 37

*Topic **ts**

adjRatios, 2

ADX, 3

aroon, 4

ATR, 5

bollingerBands, 7

CCI, 8

chaikinAD, 9

chaikinVolatility, 11

changes, 12

CLV, 13

CMF, 14

CMO, 15

DonchianChannel, 16

DPO, 17

EMV, 18

GMMA, 20

KST, 21

MACD, 22

MFI, 24

MovingAverages, 25

OBV, 27

RSI, 28

runFun, 30

SAR, 31

stochastics, 32

TDI, 35

TRIX, 36

TTRtools, 40

VHF, 41

volatility, 42

WebData, 44

williamsAD, 45

WPR, 46

ZigZag, 48

%D (*stochastics*), 32

%K (*stochastics*), 32

adjRatios, 2

adjust (*adjRatios*), 2

ADX, 3, 5, 9, 20, 32, 36, 38, 41

aroon, 4, 4, 9, 20, 36, 41

ATR, 4, 5, 32, 46

BBands, 17, 38

BBands (*bollingerBands*), 7

bollingerBands, 7

CCI, 4, 5, 8, 20, 36, 41

chaikinAD, 9, 13, 15, 28, 46

chaikinVolatility, 6, 11, 43

changes, 12, 38

CLV, 10, 13, 15

CMF, 14, 25

CMO, 15, 29

DEMA (*MovingAverages*), 25

DI (*ADX*), 3

Donchian (*DonchianChannel*), 16

DonchianChannel, 16

DPO, 17

DX, 6

DX (*ADX*), 3

EMA, 4, 6, 8, 9, 12, 18, 19, 22, 24, 29, 34, 37

EMA (*MovingAverages*), 25

EMV, 18

EVWMA (*MovingAverages*), 25

garman.klass (*volatility*), 42

getYahooData (*WebData*), 44

gk.yz (*volatility*), 42

GMMA, 4, 5, 9, 20, 36, 41

growth (*TTRtools*), 40

Guppy (*GMMA*), 20

guppy (*GMMA*), 20

- KST, [21](#)

- lags (*TTRtools*), [40](#)

- MA (*MovingAverages*), [25](#)
- MACD, [18](#), [22](#), [22](#), [38](#)
- MFI, [24](#)
- momentum (*changes*), [12](#)
- moneyFlow (*MFI*), [24](#)
- MovingAverages, [25](#), [38](#)

- OBV, [10](#), [25](#), [27](#), [46](#)

- parkinson (*volatility*), [42](#)

- ROC, [22](#)
- ROC (*changes*), [12](#)
- rogers.satchell (*volatility*), [42](#)
- RSI, [16](#), [28](#), [38](#)
- runCor (*runFun*), [30](#)
- runCov (*runFun*), [30](#)
- runFun, [30](#), [38](#)
- runMAD (*runFun*), [30](#)
- runMax (*runFun*), [30](#)
- runMean (*runFun*), [30](#)
- runMedian (*runFun*), [30](#)
- runMin (*runFun*), [30](#)
- runSD (*runFun*), [30](#)
- runSum (*runFun*), [30](#)
- runVar (*runFun*), [30](#)

- SAR, [31](#)
- SMA, [4](#), [6](#), [8](#), [9](#), [12](#), [18](#), [19](#), [22](#), [24](#), [29](#), [34](#), [37](#)
- SMA (*MovingAverages*), [25](#)
- SMI (*stochastics*), [32](#)
- stoch, [29](#), [38](#), [47](#)
- stoch (*stochastics*), [32](#)
- stochastic (*stochastics*), [32](#)
- stochastics, [32](#)
- stockSymbols (*WebData*), [44](#)

- TDI, [4](#), [5](#), [9](#), [20](#), [35](#), [41](#)
- TR, [12](#), [43](#)
- TR (*ATR*), [5](#)
- TRIX, [36](#)
- TTR, [37](#)
- ttrc, [39](#)
- TTRtools, [40](#)

- VHF, [4](#), [5](#), [9](#), [20](#), [36](#), [41](#)

- volatility, [42](#)

- WebData, [38](#), [44](#)
- wilderSum, [27](#)
- wilderSum (*TTRtools*), [40](#)
- williamsAD, [45](#)
- WMA (*MovingAverages*), [25](#)
- WPR, [34](#), [46](#)

- yang.zhang (*volatility*), [42](#)

- ZigZag, [48](#)
- zigzag (*ZigZag*), [48](#)
- ZLEMA (*MovingAverages*), [25](#)