

Package ‘talib’

October 5, 2026

Title Interface to 'TA-Lib' for Technical Analysis and Candlestick Patterns

Version 0.9-4

Description Interface to the 'TA-Lib' (Technical Analysis Library) 'C' library, providing access to 150+ indicators (e.g. Average Directional Movement Index (ADX), Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), Stochastic Oscillator, Bollinger Bands), candlestick pattern recognition, and rolling-window utilities. Core computations are implemented in 'C' for fast Open-High-Low-Close-Volume (OHLCV) time-series feature engineering and rule-based signal generation, with optional interactive visualization via 'plotly'.

X-schema.org-keywords technical analysis, TA-Lib, technical indicators, algorithmic trading, trading signals, quantitative finance, financial time series, OHLCV, candlestick patterns, RSI, MACD, Bollinger Bands, Stochastic, moving averages, plotly

X-schema.org-applicationCategory Finance

X-schema.org-applicationSubCategory Algorithmic Trading

SystemRequirements CMake

License BSD_3_clause + file LICENSE

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Suggests ggplot2, knitr, plotly, rmarkdown, testthat (>= 3.0.0), xts, zoo

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

VignetteBuilder knitr

URL <https://serkor1.github.io/ta-lib-R/>,
<https://github.com/serkor1/ta-lib-R>, <https://ta-lib.org/>

BugReports <https://github.com/serkor1/ta-lib-R/issues>

Author Serkan Korkmaz [cre, aut, cph] (ORCID:

[<https://orcid.org/0000-0002-5052-0982>](https://orcid.org/0000-0002-5052-0982)),

Mario Fortier [cph] (Copyright holder of the bundled TA-Lib C library
(src/ta-lib/))

Maintainer Serkan Korkmaz <serkor1@duck.com>

Repository CRAN

Date/Publication 2026-10-05 19:50:02 UTC

Contents

abandoned_baby	5
absolute_price_oscillator	8
acceleration_bands	11
advance_block	13
aroon	16
aroon_oscillator	18
ATOM	20
average_deviation	21
average_directional_movement_index	23
average_directional_movement_index_rating	25
average_price	27
average_true_range	29
balance_of_power	31
belt_hold	33
bollinger_bands	36
break_away	39
BTC	42
chaikin_accumulation_distribution_line	43
chaikin_accumulation_distribution_oscillator	45
chaikin_money_flow	47
chande_momentum_oscillator	49
chart	51
chart_themes	54
closing_marubozu	56
commodity_channel_index	59
concealing_baby_swallow	61
counter_attack	64
dark_cloud_cover	67
directional_movement_index	70
doji	73
doji_star	76
dominant_cycle_period	79
dominant_cycle_phase	81
double_exponential_moving_average	83
dragonfly_doji	85

engulfing	89
evening_doji_star	92
evening_star	95
exponential_moving_average	98
extended_moving_average_convergence_divergence	100
extended_parabolic_stop_and_reverse	103
fast_stochastic	106
fixed_moving_average_convergence_divergence	108
gaps_side_white	111
GOOGL	114
gravestone_doji	115
hammer	118
hanging_man	121
harami	124
harami_cross	127
high_wave	130
hikakke	133
hikakke_mod	137
homing_pigeon	140
hull_moving_average	143
indicator	145
intraday_movement_index	147
inverted_hammer	149
in_neck	153
kaufman_adaptive_moving_average	156
kicking	158
kicking_baby_length	161
ladder_bottom	165
long_legged_doji	168
long_line	171
lookback	174
marubozu	175
matching_low	178
mat_hold	181
median_price	184
mesa_adaptive_moving_average	186
midpoint_period	188
midpoint_price	191
minus_directional_indicator	193
minus_directional_movement	195
momentum	197
money_flow_index	199
morning_doji_star	202
morning_star	205
moving_average_convergence_divergence	208
negative_volume_index	210
normalized_average_true_range	213
NVDA	215

on_balance_volume	215
on_neck	218
parabolic_stop_and_reverse	221
percentage_price_oscillator	223
percentage_volume_oscillator	226
phasor_components	228
piercing	230
plus_directional_indicator	233
plus_directional_movement	236
positive_volume_index	238
ratio_of_change	240
relative_strength_index	242
rickshaw_man	244
rise_fall_3_methods	248
rolling_beta	251
rolling_correlation	252
rolling_maximum	253
rolling_minimum	255
rolling_standard_deviation	256
rolling_sum	258
rolling_variance	259
separating_lines	260
set_theme	263
shooting_star	265
short_line	268
simple_moving_average	271
sine_wave	274
spinning_top	276
SPY	279
stalled_pattern	280
stick_sandwich	283
stochastic	286
stochastic_relative_strength_index	288
t3_exponential_moving_average	291
takuri	294
tasuki_gap	297
three_black_crows	300
three_identical_crows	303
three_inside	306
three_line_strike	309
three_outside	312
three_stars_in_the_south	315
three_white_soldiers	318
thrusting	322
trading_volume	325
trendline	327
trend_cycle_mode	329
triangular_moving_average	331

triple_exponential_average	334
triple_exponential_moving_average	336
tristar	339
true_range	342
two_crows	344
typical_price	347
ultimate_oscillator	349
unique_3_river	351
unsmoothed_chande_momentum_oscillator	354
upside_gap_2_crows	356
variable_moving_average_period	360
volume_weighted_moving_average	362
weighted_close_price	365
weighted_moving_average	367
williams_oscillator	369
xside_gap_3_methods	371

Index

375

abandoned_baby	<i>Abandoned Baby</i>
----------------	-----------------------

Description

`abandoned_baby()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend

encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
abandoned_baby(x, cols, penetration = 0.3, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>penetration</code>	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLABANDONEDBABY [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::abandoned_baby(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::abandoned_baby
  )
}
```

absolute_price_oscillator

Absolute Price Oscillator

Description

`absolute_price_oscillator()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

`absolute_price_oscillator()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single

NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
absolute_price_oscillator(
  x,
  cols,
  fastPeriod = 12,
  slowPeriod = 26,
  maType = 1,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>fastPeriod</code>	(integer). Period of the fast MA. Defaults to 12.
<code>slowPeriod</code>	(integer). Period of the slow MA. Defaults to 26.
<code>maType</code>	(integer). Type of Moving Average. Defaults to 1 (EMA). Can also be passed as <code>talib::EMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

APO `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::absolute_price_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::absolute_price_oscillator
```

```
)
}
```

acceleration_bands	<i>Acceleration Bands</i>
--------------------	---------------------------

Description

`acceleration_bands()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
acceleration_bands(x, cols, timePeriod = 20, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 3-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .

timePeriod (integer). Lookback period (window size). A positive integer of length 1.

na.bridge (logical). A logical of length 1. FALSE by default. When FALSE, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same class and length of x:

UpperBand double

MiddleBand double

LowerBand double

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `extended_parabolic_stop_and_reverse()`, `hull_moving_average()`, `kaufman_adaptive_moving_average()`, `mesa_adaptive_moving_average()`, `midpoint_period()`, `midpoint_price()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_moving_average()`, `variable_moving_average_period()`, `volume_weighted_moving_average()`, `weighted_moving_average()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::acceleration_bands(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
```

```
## chart OHLC-V
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::acceleration_bands
)
}
```

advance_block

Advance Block

Description

`advance_block()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
advance_block(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLADVANCEBLOCK [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::advance_block(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::advance_block
  )
}
```

aroon

Aroon

Description

aroon() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
aroon(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 2-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

AroonDown [double](#)

AroonUp [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::aroon(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::aroon
  )
}
```

aroon_oscillator

Aroon Oscillator

Description

aroon_oscillator() is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
aroon_oscillator(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

AROONOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::aroon_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::aroon_oscillator
  )
}
```

 ATOM

Cosmos (ATOM)

Description

Daily OHLCV price data for Cosmos (ATOM), denominated in USDC, covering 2022-01-01 to 2022-12-31. Includes a full bear-market cycle useful for testing candlestick pattern recognition on cryptocurrency data.

Usage

ATOM

Format

A [data.frame](#) with 366 rows and 5 columns.

open Opening price for the trading day.

high Highest price reached during the trading day.

low Lowest price reached during the trading day.

close Closing price for the trading day.

volume Total trading volume for the day.

Source

Loaded using [cryptoQuotes](#).

Examples

```
## Load the dataset
data(ATOM, package = "talib")

## Scan for Doji patterns on ATOM
talib::doji(ATOM)
```

average_deviation	<i>Average Deviation</i>
-------------------	--------------------------

Description

`average_deviation()` is a generic S3 function that preserves the input **class**: [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

`average_deviation()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
average_deviation(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

AVGDEV [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_price\(\)](#), [median_price\(\)](#), [typical_price\(\)](#), [weighted_close_price\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::average_deviation(BTC)

## display the results
utils::tail(output)
```

average_directional_movement_index

Average Directional Movement Index

Description

average_directional_movement_index() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
average_directional_movement_index(
  x,
  cols,
  timePeriod = 14,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low + close.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

ADX [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::average_directional_movement_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::average_directional_movement_index
  )
}
```

average_directional_movement_index_rating

Average Directional Movement Index Rating

Description

average_directional_movement_index_rating() is a generic S3 function that preserves the input class: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
average_directional_movement_index_rating(
  x,
  cols,
  timePeriod = 14,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

ADXR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::average_directional_movement_index_rating(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::average_directional_movement_index_rating
  )
}
```

Description

`average_price()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
average_price(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

AVGPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_deviation\(\)](#), [median_price\(\)](#), [typical_price\(\)](#), [weighted_close_price\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::average_price(BTC)

## display the results
utils::tail(output)
```

average_true_range	<i>Average True Range</i>
--------------------	---------------------------

Description

`average_true_range()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
average_true_range(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

ATR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicators: [normalized_average_true_range\(\)](#), [true_range\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::average_true_range(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::average_true_range
  )
}
```

balance_of_power

Balance Of Power

Description

`balance_of_power()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
balance_of_power(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

BOP [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#),


```

extended_moving_average_convergence_divergence(), fast_stochastic(), fixed_moving_average_convergence,
intraday_movement_index(), minus_directional_indicator(), minus_directional_movement(),
momentum(), money_flow_index(), moving_average_convergence_divergence(), percentage_price_oscillator(),
plus_directional_indicator(), plus_directional_movement(), ratio_of_change(), relative_strength_index(),
stochastic(), stochastic_relative_strength_index(), triple_exponential_average(),
ultimate_oscillator(), unsmoothed_chande_momentum_oscillator(), williams_oscillator()

```

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::balance_of_power(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::balance_of_power
  )
}

```

belt_hold

Belt-hold

Description

belt_hold() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
belt_hold(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLBELTHOLD [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: `abandoned_baby()`, `advance_block()`, `break_away()`, `closing_marubozu()`, `concealing_baby_swallow()`, `counter_attack()`, `dark_cloud_cover()`, `doji()`, `doji_star()`, `dragonfly_doji()`, `engulfing()`, `evening_doji_star()`, `evening_star()`, `gaps_side_white()`, `gravestone_doji()`, `hammer()`, `hanging_man()`, `harami()`, `harami_cross()`, `high_wave()`, `hikakke()`, `hikakke_mod()`, `homing_pigeon()`, `in_neck()`, `inverted_hammer()`, `kicking()`, `kicking_baby_length()`, `ladder_bottom()`, `long_legged_doji()`, `long_line()`, `marubozu()`, `mat_hold()`, `matching_low()`, `morning_doji_star()`, `morning_star()`, `on_neck()`, `piercing()`, `rickshaw_man()`, `rise_fall_3_methods()`, `separating_lines()`, `shooting_star()`, `short_line()`, `spinning_top()`, `stalled_pattern()`, `stick_sandwich()`, `takuri()`, `tasuki_gap()`, `three_black_crows()`, `three_identical_crows()`, `three_inside()`, `three_line_strike()`, `three_outside()`, `three_stars_in_the_south()`, `three_white_soldiers()`, `thrusting()`, `tristar()`, `two_crows()`, `unique_3_river()`, `upside_gap_2_crows()`, `xside_gap_3_methods()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::belt_hold(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::belt_hold
  )
}
```

Description

`bollinger_bands()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`bollinger_bands()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
bollinger_bands(
  x,
  cols,
  timePeriod = 20,
  deviationsUp = 2,
  deviationsDown = 2,
  maType = 0,
  na.bridge = FALSE,
  ...
)
```

Arguments

`x` An OHLC-V series coercible to `data.frame`. Alternatively, `x` may also be supplied as a `double` vector.

cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
deviationsUp	(double). Deviation multiplier for upper band. Defaults to 2.
deviationsDown	(double). Deviation multiplier for lower band. Defaults to 2.
maType	(integer). Type of Moving Average. Defaults to 0 (SMA). Can also be passed as talib::SMA.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

UpperBand [double](#)

MiddleBand [double](#)

LowerBand [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
```

```

output <- talib::bollinger_bands(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::bollinger_bands
  )
}

```

break_away

Breakaway

Description

`break_away()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several

real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
break_away(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLBREAKAWAY [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::break_away(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::break_away
  )
}
```

BTC	<i>Bitcoin (BTC)</i>
-----	----------------------

Description

Daily OHLCV price data for Bitcoin (BTC), denominated in USDC, covering 2024-01-01 to 2024-12-31. Useful for testing technical analysis indicators and candlestick pattern recognition on cryptocurrency data.

Usage

BTC

Format

- A [data.frame](#) with 366 rows and 5 columns.
- open** Opening price for the trading day.
- high** Highest price reached during the trading day.
- low** Lowest price reached during the trading day.

close Closing price for the trading day.

volume Total trading volume for the day.

Source

Loaded using [cryptoQuotes](#).

Examples

```
## Load the dataset
data(BTC, package = "talib")

## Compute RSI on Bitcoin closing prices
talib::relative_strength_index(BTC)
```

```
chaikin_accumulation_distribution_line
Chaikin A/D Line
```

Description

`chaikin_accumulation_distribution_line()` is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
chaikin_accumulation_distribution_line(x, cols, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low + close + volume.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

AD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::chaikin_accumulation_distribution_line(BTC)

## display the results
utils::tail(output)
```

```

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::chaikin_accumulation_distribution_line
  )
}

```

chaikin_accumulation_distribution_oscillator

Chaikin A/D Oscillator

Description

chaikin_accumulation_distribution_oscillator() is a generic S3 function that preserves the input class: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
chaikin_accumulation_distribution_oscillator(
  x,
  cols,
  fastPeriod = 3,
  slowPeriod = 10,
  na.bridge = FALSE,
  ...
)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low + close + volume.
fastPeriod	(integer). Period of the fast MA. Defaults to 3.
slowPeriod	(integer). Period of the slow MA. Defaults to 10.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

ADOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::chaikin_accumulation_distribution_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::chaikin_accumulation_distribution_oscillator
  )
}
```

chaikin_money_flow	<i>Chaikin Money Flow</i>
--------------------	---------------------------

Description

chaikin_money_flow() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
chaikin_money_flow(x, cols, timePeriod = 20, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close + volume</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

CMF [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::chaikin_money_flow(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::chaikin_money_flow
  )
}
```

chande_momentum_oscillator

Chande Momentum Oscillator

Description

chande_momentum_oscillator() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

chande_momentum_oscillator() also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single

NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
chande_momentum_oscillator(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

CMO [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::chande_momentum_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::chande_momentum_oscillator
  )
}
```

chart

Create an OHLC Chart

Description

`chart()` creates interactive candlestick or OHLC bar charts from financial price data. It initializes the charting environment so that subsequent calls to [indicator\(\)](#) can attach technical indicators as subcharts.

Calling `chart()` without any arguments resets the charting environment, clearing all stored chart state (main chart, subcharts, and data).

See `vignette(topic = "charting", package = "talib")` for a comprehensive guide on building multi-panel technical analysis charts.

Usage

```
chart(x, type = "candlestick", idx = NULL, title, ...)
```

Arguments

<code>x</code>	An OHLC-V data.frame (or object coercible to one) with columns named open, high, low, close, and optionally volume. Column names are case-sensitive.
<code>type</code>	A character string, either "candlestick" (default) or "ohlc". Candlestick charts use filled/hollow bodies with wicks; OHLC charts use vertical bars with horizontal open/close ticks.
<code>idx</code>	An optional vector with the same length as the number of rows in <code>x</code> . Replaces the default x-axis labels (row names or integer index). Useful for custom date formatting or non-standard index types.
<code>title</code>	An optional character string for the chart title. If omitted, the title is inferred from the variable name passed to <code>x</code> .
<code>...</code>	Additional parameters passed to the backend chart constructor (e.g., <code>plotly::plot_ly()</code>).

Details

`chart()` acts as the entry point for the package's charting system. It stores the OHLC data and the main price chart internally so that subsequent `indicator()` calls can attach panels below the price chart without requiring the data to be passed again.

The chart title is automatically inferred from the name of the object passed to `x` (e.g., `chart(BTC)` produces the title "BTC"). The title also displays the number of observations and, when available, the date range.

Two rendering backends are supported:

"plotly" (default) Produces interactive HTML charts with hover tooltips, pan/zoom, and built-in drawing tools (lines, rectangles). Requires the **plotly** package.

"ggplot2" Produces static charts suitable for reports and publications. Requires the **ggplot2** package.

Options:

The following `options()` control chart appearance and behavior:

`talib.chart.backend` [**character**] "plotly" by default. Set to "ggplot2" for static charts.

`talib.chart.slider` [**logical**] FALSE by default. If TRUE, a range slider is added below the x-axis for interactive zooming (plotly backend only).

`talib.chart.slider.size` [**numeric**] 0.05 by default. Controls the height of the range slider as a fraction of the total chart height.

`talib.chart.legend` [**logical**] TRUE by default. If FALSE, legends are hidden on all panels.

`talib.chart.scale` [**numeric**] 1 by default. A scaling factor applied to all font sizes. Values greater than 1 increase font size.

`talib.chart.main` [**numeric**] 0.7 by default. The fraction of total chart height allocated to the main price panel when subcharts are present.

Colors are controlled via `set_theme()`. See `set_theme()` for available themes and color customization.

State and concurrency:

The `chart()` + `indicator()` pair follows the same active-target model as base R's `plot()` / `lines()`. When `chart()` is called it stashes a per-pipeline state object in its caller's evaluation frame; subsequent `indicator()` calls retrieve the state by walking up the call stack.

The practical consequences:

- `chart()` and the subsequent `indicator()` calls must live in the **same enclosing frame** - the same REPL session, the same function body, the same `renderPlot()` / `renderPlotly()` block, the same `local({...})` expression, the same `testthat::test_that({...})` block, etc. Splitting them across unrelated helpers is not supported.
- Two unrelated function bodies (or two parallel `renderPlot()` callbacks in a Shiny app, or two `future::future()` blocks) each get their own frame, so their chart states are isolated by construction - without `talib` taking any dependency on Shiny, promises, or futures.
- Calling `chart()` with no arguments clears the state in the caller's frame, mirroring a fresh `plot()` call on a new device.

Value

A chart object whose class depends on the active backend:

- "plotly" backend: a plotly object (interactive HTML widget).
- "ggplot2" backend: a gg object (static plot).

When called without arguments, returns NULL invisibly.

Author(s)

Serkan Korkmaz

See Also

`indicator()` to attach technical indicators, `set_theme()` to customize chart colors.

Other Charting: `chart_themes`, `indicator()`, `set_theme()`

Examples

```
## charting OHLC data with {talib}
data(BTC, package = "talib")

## candlestick chart (default)
talib::chart(BTC)

## OHLC bar chart
```

```
talib::chart(BTC, type = "ohlc")

## chart with a custom title
talib::chart(BTC, title = "Bitcoin / USD")

## reset the charting environment
talib::chart()
```

chart_themes

Chart Themes

Description

The charting system ships with a set of built-in color themes. Each theme controls candle colors, background, text, grid lines, and a 10-color palette (colorway) used to distinguish indicator lines.

Use `set_theme()` to apply or list themes.

Details

Available Themes:

default A dark theme with cyan and steel-blue tones. Light (#E0FFFF) bullish candles on a near-black (#141414) background with a cool blue (#4682B4) bearish candle. The colorway spans icy blues through teal and soft purple.

hawks_and_doves A light, grayscale theme with a white background. Candles use shades of gray, making it suitable for print or presentations where color is secondary. The colorway uses muted, accessible tones.

payout A dark teal theme on a near-black (#1A1A1A) background. Bullish candles are teal (#008080), bearish candles are dark slate (#2F4F4F). The colorway follows the default Plotly palette.

tp_slapped A bright theme on a light gray (#ECF0F1) background. Red (#E74C3C) bearish and teal (#1ABC9C) bullish candles provide strong visual contrast. The colorway uses vivid, saturated colors.

trust_the_process A subtle, earth-toned theme on a light gray (#F5F5F5) background. Both bull and bear candles use shades of gray, keeping the focus on indicator lines. The colorway uses muted natural tones.

bloomberg_terminal A dark theme on a near-black (#0E1017) background with orange (#FF8F40) bullish and neutral-gray (#BBB9B2) bearish candles. Inspired by the Bloomberg Terminal interface. Colorblind-friendly (see below).

limit_up A dark monochrome theme on a near-black (#121212) background. Candles use only luminance to encode direction (light-gray bullish vs dark-gray bearish). Colorblind-friendly (see below).

bid_n_ask A light theme on an azure (#F0FFFF) background with steel-blue (#4682B4) bullish and tomato-red (#FF6347) bearish candles. The classic blue-vs-red trading pair. Colorblind-friendly (see below).

Colorblind-Friendly Themes:

The following themes encode bull/bear direction in ways that remain distinguishable under the most common color-vision deficiencies. The colorways for `bloomberg_terminal`, `limit_up`, and `bid_n_ask` are derived from the Okabe & Ito (2008) qualitative palette, the de-facto standard for accessible scientific visualization.

`limit_up`, `hawks_and_doves`, `trust_the_process` Encode direction with luminance only. Safe under deuteranopia, protanopia, tritanopia, and full achromatopsia.

`bloomberg_terminal` Orange + neutral gray. Safe under all three CVD types thanks to Okabe-Ito-style hue separation.

`default`, `payout` Cyan/teal + blue or dark slate. Safe under all three CVD types — the color pair sits inside the blue-yellow axis that CVD users perceive normally.

`bid_n_ask` Blue + red. Safe under deuteranopia and protanopia (the most common forms, affecting ~8% of males); the pair separates more weakly under tritanopia.

`tp_slapped` is the only built-in that uses a teal/red pair adjacent to the red-green CVD axis; prefer the themes above when accessibility matters.

Theme Properties:

Each theme sets the following color properties:

Candle colors `bearish_body`, `bearish_wick`, `bearish_border`, `bullish_body`, `bullish_wick`, `bullish_border`

General colors `background_color`, `foreground_color` (axes and borders), `text_color`

Grid and reference `gridcolor`, `threshold_color` (horizontal reference lines such as overbought/oversold levels)

Colorway A vector of 10 colors cycled through for indicator lines and legends

Any of these properties can be individually overridden via the `...` argument to `set_theme()`.

References

Okabe, M. & Ito, K. (2008). *Color Universal Design (CUD): How to make figures and presentations that are friendly to colorblind people*. <https://jfly.uni-koeln.de/color/>

See Also

Other Charting: `chart()`, `indicator()`, `set_theme()`

Examples

```
## list available themes
talib::set_theme()

## apply a theme by name
talib::set_theme("payout")

## apply a theme with custom overrides
talib::set_theme(
  "hawks_and_doves",
  background_color = "#FAFAFA"
```

```

)

## override individual properties
## without switching theme
talib::set_theme(
  bearish_body = "#FF4444",
  bullish_body = "#44FF44"
)

## reset to default theme
talib::set_theme("default")

```

closing_marubozu

Closing Marubozu

Description

closing_marubozu() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
closing_marubozu(x, cols, na.bridge = FALSE, ...)
```


Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (`N = 5`, `alpha = 0.6`). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLCLOSINGMARUBOZU [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::closing_marubozu(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::closing_marubozu
  )
}
```

commodity_channel_index

Commodity Channel Index

Description

commodity_channel_index() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
commodity_channel_index(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 3-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low + close.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

CCI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")
```

```

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::commodity_channel_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::commodity_channel_index
  )
}

```

concealing_baby_swallow

Concealing Baby Swallow

Description

`concealing_baby_swallow()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
concealing_baby_swallow(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLCONCEALBABYSWALL [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::concealing_baby_swallow(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::concealing_baby_swallow
  )
}
```

counter_attack

Counterattack

Description

counter_attack() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
counter_attack(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLCOUNTERATTACK [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#),

```
stick_sandwich(), takuri(), tasuki_gap(), three_black_crows(), three_identical_crows(),
three_inside(), three_line_strike(), three_outside(), three_stars_in_the_south(), three_white_soldiers(),
thrusting(), tristar(), two_crows(), unique_3_river(), upside_gap_2_crows(), xside_gap_3_methods()
```

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::counter_attack(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::counter_attack
  )
}
```

dark_cloud_cover

Dark Cloud Cover

Description

dark_cloud_cover() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
dark_cloud_cover(x, cols, penetration = 0.5, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>penetration</code>	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLDARKCLOUDCOVER [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::dark_cloud_cover(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::dark_cloud_cover
  )
}
```

Description

`directional_movement_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
directional_movement_index(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 3-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of <code>length</code> 1.
<code>na.bridge</code>	(<code>logical</code>). A <code>logical</code> of <code>length</code> 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

DX `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::directional_movement_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::directional_movement_index
```



```
)
}
```

doji

Doji

Description

`doji()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
doji(x, cols, na.bridge = FALSE, ...)
```

Arguments

`x` An OHLC-V series coercible to `data.frame`.

`cols` (`formula`). An optional 4-variable `formula` selecting columns from `x` via `model.frame`. For `xts` input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches `TICKER.Close`), case-insensitively - and columns are consumed in formula order. Defaults to `~open + high + low + close`.

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLDOJI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::doji(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::doji
)
}

```

doji_star

Doji Star

Description

doji_star() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; xts in, xts out. Other inputs coercible to `data.frame` (e.g. zoo) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
doji_star(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (`N = 5`, `alpha = 0.6`). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLDOJISTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::doji_star(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::doji_star
  )
}
```

dominant_cycle_period *Hilbert Transform - Dominant Cycle Period*

Description

dominant_cycle_period() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

dominant_cycle_period() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
dominant_cycle_period(x, cols, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

HT_DCPERIOD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicators: [dominant_cycle_phase\(\)](#), [phasor_components\(\)](#), [sine_wave\(\)](#), [trend_cycle_mode\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::dominant_cycle_period(BTC)

## display the results
utils::tail(output)

## visualize the indicator
```



```
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::dominant_cycle_period
  )
}
```

dominant_cycle_phase *Hilbert Transform - Dominant Cycle Phase*

Description

dominant_cycle_phase() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

dominant_cycle_phase() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend

encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
dominant_cycle_phase(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

HT_DCPHASE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicators: [dominant_cycle_period\(\)](#), [phasor_components\(\)](#), [sine_wave\(\)](#), [trend_cycle_mode\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::dominant_cycle_phase(BTC)
```

```
## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::dominant_cycle_phase
  )
}
```

double_exponential_moving_average

Double Exponential Moving Average

Description

double_exponential_moving_average() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

double_exponential_moving_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
double_exponential_moving_average(
  x,
  timePeriod = 30,
  cols,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without `'x'`, `double_exponential_moving_average` functions as an 'Moving Average'-specification which is used in, for example, `stochastic` when constructing the smoothing lines.

When called without `'x'` it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of `x`:

DEMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::double_exponential_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::double_exponential_moving_average
  )
}
```

Description

`dragonfly_doji()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
dragonfly_doji(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLDRAGONFLYDOJI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::dragonfly_doji(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::dragonfly_doji
  )
}
```


Description

engulfing() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
engulfing(x, cols, na.bridge = FALSE, ...)
```

Arguments

- | | |
|-------------------|--|
| <code>x</code> | An OHLC-V series coercible to <code>data.frame</code> . |
| <code>cols</code> | (<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> . |

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLENGULFING [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::engulfing(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::engulfing
)
}

```

evening_doji_star

Evening Doji Star

Description

`evening_doji_star()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
evening_doji_star(x, cols, penetration = 0.3, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~open + high + low + close.
penetration	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means "<=20% of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLEVENINGDOJISTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::evening_doji_star(BTC)

## display the results
utils::tail(output)
```

```
## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::evening_doji_star
  )
}
```

evening_star

Evening Star

Description

`evening_star()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
evening_star(x, cols, penetration = 0.3, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>penetration</code>	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLEVENINGSTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::evening_star(BTC)
```

```
## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::evening_star
  )
}
```

exponential_moving_average

Exponential Moving Average

Description

exponential_moving_average() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

exponential_moving_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
exponential_moving_average(x, timePeriod = 30, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without `'x'`, [exponential_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without `'x'` it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of `x`:

EMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::exponential_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::exponential_moving_average
  )
}
```

extended_moving_average_convergence_divergence
MACD with controllable MA type

Description

`extended_moving_average_convergence_divergence()` is a generic S3 function that preserves the input **class**: [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

`extended_moving_average_convergence_divergence()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
extended_moving_average_convergence_divergence(
  x,
  cols,
  fastPeriod = 12,
  fastMa = 0,
  slowPeriod = 26,
  slowMa = 0,
  signalPeriod = 9,
  signalMa = 0,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches

	TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
fastPeriod	(integer). Period of the fast MA. Defaults to 12.
fastMa	(integer). Type of Moving Average for fast MA. Defaults to 0 (SMA). Can also be passed as talib::SMA.
slowPeriod	(integer). Period of the slow MA. Defaults to 26.
slowMa	(integer). Type of Moving Average for slow MA. Defaults to 0 (SMA). Can also be passed as talib::SMA.
signalPeriod	(integer). Smoothing for the signal line (period length). Defaults to 9.
signalMa	(integer). Type of Moving Average for signal line. Defaults to 0 (SMA). Can also be passed as talib::SMA.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame. Unused for xts input (a warning is emitted when supplied).

Value

An object of same class and length of x:

MACD double

MACDSignal double

MACDHist double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::extended_moving_average_convergence_divergence(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::extended_moving_average_convergence_divergence
  )
}
```

extended_parabolic_stop_and_reverse

Parabolic SAR - Extended

Description

extended_parabolic_stop_and_reverse() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
extended_parabolic_stop_and_reverse(
  x,
  cols,
  startValue = 0,
  offsetOnReverse = 0,
  afInitLong = 0.02,
  afLong = 0.02,
  afMaxLong = 0.2,
  afInitShort = 0.02,
  afShort = 0.02,
  afMaxShort = 0.2,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .
<code>startValue</code>	(double). Start value and direction. 0 for Auto, >0 for Long, <0 for Short. Defaults to 0.
<code>offsetOnReverse</code>	(double). Percent offset added/removed to initial stop on short/long reversal. Defaults to 0.
<code>afInitLong</code>	(double). Acceleration Factor initial value for the Long direction. Defaults to 0.02.
<code>afLong</code>	(double). Acceleration Factor for the Long direction. Defaults to 0.02.
<code>afMaxLong</code>	(double). Acceleration Factor maximum value for the Long direction. Defaults to 0.2.

afInitShort	(double). Acceleration Factor initial value for the Short direction. Defaults to 0.02.
afShort	(double). Acceleration Factor for the Short direction. Defaults to 0.02.
afMaxShort	(double). Acceleration Factor maximum value for the Short direction. Defaults to 0.2.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into <code>model.frame</code> . Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

SAREXT double

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: `acceleration_bands()`, `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `hull_moving_average()`, `kaufman_adaptive_moving_average()`, `mesa_adaptive_moving_average()`, `midpoint_period()`, `midpoint_price()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_moving_average()`, `variable_moving_average_period()`, `volume_weighted_moving_average()`, `weighted_moving_average()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::extended_parabolic_stop_and_reverse(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
```

```
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::extended_parabolic_stop_and_reverse
  )
}
```

fast_stochastic

Stochastic Fast

Description

`fast_stochastic()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
fast_stochastic(
  x,
  cols,
  fastKPeriod = 5,
  fastDPeriod = 3,
  fastDMA = 0,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>fastKPeriod</code>	(integer). Time period for building the Fast-K line. Defaults to 5.
<code>fastDPeriod</code>	(integer). Smoothing for making the Fast-D line. Usually set to 3. Defaults to 3.
<code>fastDMA</code>	(integer). Type of Moving Average for Fast-D. Defaults to 0 (SMA). Can also be passed as <code>talib::SMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

FastK [double](#)

FastD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::fast_stochastic(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::fast_stochastic
  )
}
```

fixed_moving_average_convergence_divergence

Moving Average Convergence/Divergence Fix 12/26

Description

`fixed_moving_average_convergence_divergence()` is a generic S3 function that preserves the input class: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`fixed_moving_average_convergence_divergence()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
fixed_moving_average_convergence_divergence(
  x,
  cols,
  signalPeriod = 9,
  na.bridge = FALSE,
  ...
)
```

Arguments

- | | |
|---------------------------|---|
| <code>x</code> | An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector. |
| <code>cols</code> | (formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> . |
| <code>signalPeriod</code> | (integer). Smoothing for the signal line (period length). Defaults to 9. |

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

MACD double

MACDSignal double

MACDHist double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::fixed_moving_average_convergence_divergence(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
```

```
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::fixed_moving_average_convergence_divergence
  )
}
```

gaps_side_white

Up/Down-gap side-by-side white lines

Description

`gaps_side_white()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
gaps_side_white(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLGAPSIDESIDEWHITE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::gaps_side_white(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::gaps_side_white
  )
}
```

GOOGL

Alphabet Inc. (GOOGL)

Description

Daily OHLCV price data for Alphabet Inc. (GOOGL), denominated in USD, covering 2019-01-01 to 2021-12-31. Stored as an xts object with a Date index and quantmod-style prefixed column names - the exact shape returned by `quantmod::getSymbols()` - and used by the unit tests for the xts methods.

Usage

GOOGL

Format

An xts object with 756 rows and 6 columns.

GOOGL.Open Opening price for the trading day.

GOOGL.High Highest price reached during the trading day.

GOOGL.Low Lowest price reached during the trading day.

GOOGL.Close Closing price for the trading day.

GOOGL.Volume Total trading volume for the day.

GOOGL.Adjusted Adjusted closing price for the trading day.

Source

Loaded using [quantmod](#).

Examples

```
## Load the dataset
data(GOOG, package = "talib")

## Scan for Doji patterns on GOOG
talib::doji(GOOG)
```

gravestone_doji	<i>Gravestone Doji</i>
-----------------	------------------------

Description

`gravestone_doji()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
gravestone_doji(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLGRAVESTONEDOJI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::gravestone_doji(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::gravestone_doji
  )
}
```

hammer

Hammer

Description

hammer() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

na.bridge = TRUE NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with na.bridge = TRUE over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with which(is.na(x)) before enabling on low-quality time series.

Usage

```
hammer(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHAMMER [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::hammer(BTC)

## display the results
```



```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::hammer
  )
}

```

hanging_man

Hanging Man

Description

`hanging_man()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless

across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
hanging_man(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHANGINGMAN [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")
```

```
## calculate the indicator
## for Bitcoin (BTC)
output <- talib::hanging_man(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::hanging_man
  )
}
```

harami

Harami Pattern

Description

`harami()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a

series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
harami(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHARAMI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::harami(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::harami
  )
}
```

harami_cross

Harami Cross Pattern

Description

`harami_cross()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
harami_cross(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHARAMICROSS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#),

```

spinning_top(), stalled_pattern(), stick_sandwich(), takuri(), tasuki_gap(), three_black_crows(),
three_identical_crows(), three_inside(), three_line_strike(), three_outside(), three_stars_in_the_south(),
three_white_soldiers(), thrusting(), tristar(), two_crows(), unique_3_river(), upside_gap_2_crows(),
xside_gap_3_methods()

```

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::harami_cross(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::harami_cross
  )
}

```

high_wave

High-Wave Candle

Description

high_wave() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
high_wave(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHIGHWAVE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: `abandoned_baby()`, `advance_block()`, `belt_hold()`, `break_away()`, `closing_marubozu()`, `concealing_baby_swallow()`, `counter_attack()`, `dark_cloud_cover()`, `doji()`, `doji_star()`, `dragonfly_doji()`, `engulfing()`, `evening_doji_star()`, `evening_star()`, `gaps_side_white()`, `gravestone_doji()`, `hammer()`, `hanging_man()`, `harami()`, `harami_cross()`, `hikakke()`, `hikakke_mod()`, `homing_pigeon()`, `in_neck()`, `inverted_hammer()`, `kicking()`, `kicking_baby_length()`, `ladder_bottom()`, `long_legged_doji()`, `long_line()`, `marubozu()`, `mat_hold()`, `matching_low()`, `morning_doji_star()`, `morning_star()`, `on_neck()`, `piercing()`, `rickshaw_man()`, `rise_fall_3_methods()`, `separating_lines()`, `shooting_star()`, `short_line()`, `spinning_top()`, `stalled_pattern()`, `stick_sandwich()`, `takuri()`, `tasuki_gap()`, `three_black_crows()`, `three_identical_crows()`, `three_inside()`, `three_line_strike()`, `three_outside()`, `three_stars_in_the_south()`, `three_white_soldiers()`, `thrusting()`, `tristar()`, `two_crows()`, `unique_3_river()`, `upside_gap_2_crows()`, `xside_gap_3_methods()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::high_wave(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::high_wave
  )
}
```

Description

`hikakke()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
hikakke(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHIKAKE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: `abandoned_baby()`, `advance_block()`, `belt_hold()`, `break_away()`, `closing_marubozu()`, `concealing_baby_swallow()`, `counter_attack()`, `dark_cloud_cover()`, `doji()`, `doji_star()`, `dragonfly_doji()`, `engulfing()`, `evening_doji_star()`, `evening_star()`, `gaps_side_white()`, `gravestone_doji()`, `hammer()`, `hanging_man()`, `harami()`, `harami_cross()`, `high_wave()`, `hikakke_mod()`, `homing_pigeon()`, `in_neck()`, `inverted_hammer()`, `kicking()`, `kicking_baby_length()`, `ladder_bottom()`, `long_legged_doji()`, `long_line()`, `marubozu()`, `mat_hold()`, `matching_low()`, `morning_doji_star()`, `morning_star()`, `on_neck()`, `piercing()`, `rickshaw_man()`, `rise_fall_3_methods()`, `separating_lines()`, `shooting_star()`, `short_line()`, `spinning_top()`, `stalled_pattern()`, `stick_sandwich()`, `takuri()`, `tasuki_gap()`, `three_black_crows()`, `three_identical_crows()`, `three_inside()`, `three_line_strike()`, `three_outside()`, `three_stars_in_the_south()`, `three_white_soldiers()`, `thrusting()`, `tristar()`, `two_crows()`, `unique_3_river()`, `upside_gap_2_crows()`, `xside_gap_3_methods()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::hikakke(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::hikakke
  )
}
```


Description

`hikakke_mod()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
hikakke_mod(x, cols, na.bridge = FALSE, ...)
```

Arguments

- `x` An OHLC-V series coercible to `data.frame`.
- `cols` (`formula`). An optional 4-variable `formula` selecting columns from `x` via `model.frame`. For `xts` input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches `TICKER.Close`), case-insensitively - and columns are consumed in formula order. Defaults to `~open + high + low + close`.

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHIKKAKEMOD [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::hikakke_mod(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::hikakke_mod
)
}

```

homing_pigeon

Homing Pigeon

Description

`homing_pigeon()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
homing_pigeon(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLHOMINGPIGEON [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::homing_pigeon(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::homing_pigeon
  )
}
```

hull_moving_average	<i>Hull Moving Average</i>
---------------------	----------------------------

Description

`hull_moving_average()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`hull_moving_average()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
hull_moving_average(x, timePeriod = 20, cols, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

When passed without 'x', [hull_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

HMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::hull_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::hull_moving_average
  )
}
```

indicator

Add Technical Indicators to a Chart

Description

`indicator()` attaches one or more technical indicators to an existing `chart()`, or renders an indicator as a standalone chart. Each indicator is displayed in its own subchart panel below the main price chart.

If `chart()` has not been called beforehand, `indicator()` creates a standalone indicator chart — in this case, data must be provided explicitly.

See `vignette(topic = "charting", package = "talib")` for a comprehensive guide.

Usage

```
indicator(FUN, ...)
```

Arguments

<code>FUN</code>	An indicator function or an indicator call. In single indicator mode, pass the bare function (e.g., <code>RSI</code>); arguments for the indicator go in <code>...</code> . In multi-indicator mode, pass a call with parentheses (e.g., <code>RSI(n = 14)</code>); additional indicator calls go in <code>...</code> .
<code>...</code>	In single indicator mode: arguments passed to <code>FUN</code> (e.g., <code>n = 14</code> , <code>data = BTC</code>). In multi-indicator mode: additional indicator calls to merge onto the same panel (e.g., <code>RSI(n = 21)</code> , <code>MACD()</code>).

Details

`indicator()` operates in two modes depending on how `FUN` is passed:

Single Indicator Mode:

Pass a bare function name (without parentheses) and its arguments via `...`:

```
chart(BTC)
indicator(RSI, n = 14)
```

Multi-Indicator Mode:

Pass one or more indicator calls (with parentheses) to merge them onto a single subchart panel:

```
chart(BTC)
indicator(RSI(n = 10), RSI(n = 14), RSI(n = 21))
```

Each indicator retains its own arguments. Different indicator types can be freely combined on the same panel:

```
indicator(RSI(n = 14), MACD())
```

Multi-indicator mode requires an existing `chart()` — it cannot be used standalone.

Standalone Mode:

When no `chart()` has been called, a standalone indicator chart is created. The data argument is required in this case:

```
indicator(RSI, data = BTC, n = 14)
```

The chart title is automatically derived from the indicator function name, converting snake_case to Title Case.

Panel Layout:

When subcharts are present, the main price panel occupies 70% of the total height by default (configurable via `options(talib.chart.main = ...)`), and the remaining space is divided equally among subchart panels.

Value

A chart object whose class depends on the active backend:

- "plotly" backend: a plotly object containing the assembled multi-panel chart.
- "ggplot2" backend: a `talib_chart` object (when combined with `chart()`) or a `gg` object (standalone).

Author(s)

Serkan Korkmaz

See Also[chart\(\)](#) to create the main price chart, [set_theme\(\)](#) to customize chart colors.Other Charting: [chart\(\)](#), [chart_themes](#), [set_theme\(\)](#)**Examples**

```
## indicator charts with {talib}
data(BTC, package = "talib")

## standalone indicator chart
## (no prior chart() call needed)
talib::indicator(
  talib::RSI,
  data = BTC
)

## attach an indicator to a price chart
talib::chart(BTC)
talib::indicator(talib::RSI, timePeriod = 14)

## multiple indicators on the same panel
talib::chart(BTC)
talib::indicator(
  talib::RSI(timePeriod = 10),
  talib::RSI(timePeriod = 14),
  talib::RSI(timePeriod = 21)
)

## reset chart state
talib::chart()
```

intraday_movement_index*Intraday Momentum Index*

Description

`intraday_movement_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data. `frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
intraday_movement_index(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

IMI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::intraday_movement_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::intraday_movement_index
  )
}
```

Description

`inverted_hammer()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
inverted_hammer(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of <i>length</i> 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLINVERTEDHAMMER [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::inverted_hammer(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::inverted_hammer
  )
}
```


in_neck

In-Neck Pattern

Description

`in_neck()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
in_neck(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLINNECK [integer](#)

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::in_neck(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```

```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::in_neck
)
}
```

kaufman_adaptive_moving_average

Kaufman Adaptive Moving Average

Description

kaufman_adaptive_moving_average() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

kaufman_adaptive_moving_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
kaufman_adaptive_moving_average(
  x,
  timePeriod = 30,
  cols,
  na.bridge = FALSE,
  ...
)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

When passed without 'x', [kaufman_adaptive_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

KAMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::kaufman_adaptive_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::kaufman_adaptive_moving_average
  )
}
```

kicking

Kicking

Description

`kicking()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
kicking(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha *double*. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same *class* and *length* of x:

CDLKICKING *integer*

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::kicking(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::kicking
  )
}
```

kicking_baby_length	<i>Kicking - bull/bear determined by the longer marubozu</i>
---------------------	--

Description

`kicking_baby_length()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
kicking_baby_length(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLKICKINGBYLENGTH [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::kicking_baby_length(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::kicking_baby_length
  )
}
```

ladder_bottom

Ladder Bottom

Description

`ladder_bottom()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
ladder_bottom(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLLADDERBOTTOM [integer](#)

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::ladder_bottom(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::ladder_bottom
)
}

```

long_legged_doji

Long Legged Doji

Description

long_legged_doji() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
long_legged_doji(x, cols, na.bridge = FALSE, ...)
```


Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLLONGLEGGEDDOJI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::long_legged_doji(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
```

```
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::long_legged_doji
  )
}
```

long_line

Long Line Candle

Description

long_line() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
long_line(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLLONGLINE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::long_line(BTC)

## display the results
```

```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::long_line
  )
}

```

lookback

Calculate lookback period

Description

The function calculates the lookback period for a given indicator. Its primarily meant as a helper function for downstream packages that wants to use a customized control-flow.

Usage

```
lookback(FUN, ...)
```

Arguments

FUN	A call or function .
...	Additional parameters passed into the indicator function. See examples for more details.

Value

The minimum lookback required to calculate the indicator. If the indicator specification and input data are invalid the function returns [NA](#), otherwise it returns an [integer](#) of [length](#) 1.

Author(s)

Serkan Korkmaz

Examples

```
## calculate the lookback
## for the bollinger bands
talib::lookback(
  talib::bollinger_bands,
  x = talib::BTC
)
```

marubozu

Marubozu

Description

marubozu() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
marubozu(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLMARUBOZU [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::marubozu(BTC)

## display the results
utils::tail(output)

## visualize the indicator
```

```
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::marubozu
  )
}
```

matching_low

Matching Low

Description

`matching_low()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
matching_low(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLMATCHINGLOW [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::matching_low(BTC)

## display the results
```

```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::matching_low
  )
}

```

mat_hold

Mat Hold

Description

mat_hold() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
mat_hold(x, cols, penetration = 0.5, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~open + high + low + close.
penetration	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLMATHOLD [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::mat_hold(BTC)
```

```
## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::mat_hold
  )
}
```

median_price

Median Price

Description

median_price() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g.

a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
median_price(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

MEDPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_deviation\(\)](#), [average_price\(\)](#), [typical_price\(\)](#), [weighted_close_price\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::median_price(BTC)
```

```
## display the results
utils::tail(output)
```

```
mesa_adaptive_moving_average
```

MESA Adaptive Moving Average

Description

`mesa_adaptive_moving_average()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`mesa_adaptive_moving_average()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
mesa_adaptive_moving_average(
  x,
  timePeriod = 30,
  fastLimit = 0.5,
  slowLimit = 0.05,
```

```
cols,
na.bridge = FALSE,
...
)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
fastLimit	(double). Upper limit use in the adaptive algorithm. Defaults to 0.5.
slowLimit	(double). Lower limit use in the adaptive algorithm. Defaults to 0.05.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

When passed without 'x', [mesa_adaptive_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

MAMA [double](#)

FAMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::mesa_adaptive_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::mesa_adaptive_moving_average
  )
}
```

midpoint_period

MidPoint over period

Description

`midpoint_period()` is a generic S3 function that preserves the input **class**: [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

`midpoint_period()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
midpoint_period(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

MIDPOINT [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::midpoint_period(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::midpoint_period
  )
}
```

midpoint_price	<i>Midpoint Price over period</i>
----------------	-----------------------------------

Description

midpoint_price() is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

na.bridge = TRUE NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with na.bridge = TRUE over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with which(is.na(x)) before enabling on low-quality time series.

Usage

```
midpoint_price(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 2-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

MIDPRICE double

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: `acceleration_bands()`, `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `extended_parabolic_stop_and_reverse()`, `hull_moving_average()`, `kaufman_adaptive_moving_average()`, `mesa_adaptive_moving_average()`, `midpoint_period()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_moving_average()`, `variable_moving_average()`, `volume_weighted_moving_average()`, `weighted_moving_average()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::midpoint_price(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)
```



```
## chart indicator
## with default values
talib::indicator(
  talib::midpoint_price
)
}
```

minus_directional_indicator

Minus Directional Indicator

Description

`minus_directional_indicator()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
minus_directional_indicator(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

MINUS_DI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::minus_directional_indicator(BTC)
```

```

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::minus_directional_indicator
  )
}

```

minus_directional_movement

Minus Directional Movement

Description

minus_directional_movement() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

na.bridge = TRUE NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with na.bridge = TRUE over a series containing a month-long gap will compute using 14 observations that span several

real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
minus_directional_movement(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

MINUS_DM [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::minus_directional_movement(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::minus_directional_movement
  )
}
```

momentum

Momentum

Description

momentum() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

momentum() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single

NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
momentum(x, cols, timePeriod = 10, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

MOM [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::momentum(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::momentum
  )
}
```

money_flow_index

*Money Flow Index***Description**

`money_flow_index()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
money_flow_index(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close + volume</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

MFI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::money_flow_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::money_flow_index
  )
}
```

morning_doji_star	<i>Morning Doji Star</i>
-------------------	--------------------------

Description

`morning_doji_star()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
morning_doji_star(x, cols, penetration = 0.3, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>penetration</code>	(<code>double</code>). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A <code>double</code> of <code>length 1</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLMORNINGDOJISTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::morning_doji_star(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::morning_doji_star
)
}

```

morning_star

Morning Star

Description

`morning_star()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
morning_star(x, cols, penetration = 0.3, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>penetration</code>	(double). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A double of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLMORNINGSTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::morning_star(BTC)

## display the results
```

```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::morning_star
  )
}

```

moving_average_convergence_divergence

Moving Average Convergence/Divergence

Description

moving_average_convergence_divergence() is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

moving_average_convergence_divergence() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a

series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
moving_average_convergence_divergence(
  x,
  cols,
  fastPeriod = 12,
  slowPeriod = 26,
  signalPeriod = 9,
  na.bridge = FALSE,
  ...
)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
fastPeriod	(integer). Period of the fast MA. Defaults to 12.
slowPeriod	(integer). Period of the slow MA. Defaults to 26.
signalPeriod	(integer). Smoothing for the signal line (period length). Defaults to 9.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

MACD [double](#)

MACDSignal [double](#)

MACDHist [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::moving_average_convergence_divergence(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::moving_average_convergence_divergence
  )
}
```

Description

`negative_volume_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
negative_volume_index(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 2-variable <i>formula</i> selecting columns from <code>x</code> via <i>model.frame</i> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close + volume</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <i>model.frame</i> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

NVI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::negative_volume_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::negative_volume_index
  )
}
```

normalized_average_true_range

Normalized Average True Range

Description

normalized_average_true_range() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
normalized_average_true_range(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 3-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(<i>integer</i>). Lookback period (window size). A positive <i>integer</i> of length 1.

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same **class** and **length** of `x`:

NATR **double**

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicators: `average_true_range()`, `true_range()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::normalized_average_true_range(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::normalized_average_true_range
  )
}
```

NVDA	<i>NVIDIA Corporation (NVDA)</i>
------	----------------------------------

Description

Daily OHLCV price data for NVIDIA Corporation (NVDA), covering 2022-01-01 to 2023-12-31. Includes a period of high volatility useful for testing momentum and trend-following indicators.

Usage

NVDA

Format

A [matrix](#) with 501 rows and 5 columns.

open Opening price for the trading day.

high Highest price reached during the trading day.

low Lowest price reached during the trading day.

close Closing price for the trading day.

volume Total trading volume for the day.

Source

Loaded using [quantmod](#).

Examples

```
## Load the dataset
data(NVDA, package = "talib")

## Compute MACD on NVDA
talib::moving_average_convergence_divergence(NVDA)
```

on_balance_volume	<i>On Balance Volume</i>
-------------------	--------------------------

Description

`on_balance_volume()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
on_balance_volume(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 2-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close + volume</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

OBV [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::on_balance_volume(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::on_balance_volume
  )
}
```

on_neck

On-Neck Pattern

Description

on_neck() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
on_neck(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLONNECK [integer](#)

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::on_neck(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```

```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::on_neck
)
}
```

parabolic_stop_and_reverse

Parabolic SAR

Description

parabolic_stop_and_reverse() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

na.bridge = TRUE NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with na.bridge = TRUE over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with which(is.na(x)) before enabling on low-quality time series.

Usage

```
parabolic_stop_and_reverse(
  x,
```

```

    cols,
    accelerationFactor = 0.02,
    afMaximum = 0.2,
    na.bridge = FALSE,
    ...
)

```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .
<code>accelerationFactor</code>	(double). Acceleration Factor used up to the Maximum value. Defaults to <code>0.02</code> .
<code>afMaximum</code>	(double). Acceleration Factor Maximum value. Defaults to <code>0.2</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

SAR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::parabolic_stop_and_reverse(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::parabolic_stop_and_reverse
  )
}
```

percentage_price_oscillator

Percentage Price Oscillator

Description

percentage_price_oscillator() is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

percentage_price_oscillator() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data `.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
percentage_price_oscillator(
  x,
  cols,
  fastPeriod = 12,
  slowPeriod = 26,
  maType = 1,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>fastPeriod</code>	(integer). Period of the fast MA. Defaults to 12.
<code>slowPeriod</code>	(integer). Period of the slow MA. Defaults to 26.
<code>maType</code>	(integer). Type of Moving Average. Defaults to 1 (EMA). Can also be passed as <code>talib::EMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation

and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into [model.frame](#). Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

PPO [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::percentage_price_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)
```

```
## chart indicator
## with default values
talib::indicator(
  talib::percentage_price_oscillator
)
}
```

percentage_volume_oscillator

Percentage Volume Oscillator

Description

percentage_volume_oscillator() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

percentage_volume_oscillator() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
percentage_volume_oscillator(
  x,
  cols,
  fastPeriod = 12,
  slowPeriod = 26,
  maType = 1,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~volume</code> .
<code>fastPeriod</code>	(integer). Period of the fast MA. Defaults to 12.
<code>slowPeriod</code>	(integer). Period of the slow MA. Defaults to 26.
<code>maType</code>	(integer). Type of Moving Average. Defaults to 1 (EMA). Can also be passed as <code>talib::EMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

PVO [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [positive_volume_index\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::percentage_volume_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::percentage_volume_oscillator
  )
}
```

phasor_components

Hilbert Transform - Phasor Components

Description

phasor_components() is a generic S3 function that preserves the input **class**: **data.frame** in, **data.frame** out; **matrix** in, **matrix** out; **xts** in, **xts** out. Other inputs coercible to **data.frame** (e.g. **zoo**) are computed on the coerced **data.frame**.

phasor_components() also accepts a **double** vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a **double** vector; otherwise the full n by k **matrix** is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the **na.bridge** argument:

na.bridge = FALSE (default) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single

NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
phasor_components(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

InPhase [double](#)

Quadrature [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicators: [dominant_cycle_period\(\)](#), [dominant_cycle_phase\(\)](#), [sine_wave\(\)](#), [trend_cycle_mode\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::phasor_components(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::phasor_components
  )
}
```

piercing

Piercing Pattern

Description

`piercing()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
piercing(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLPIERCING [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: `abandoned_baby()`, `advance_block()`, `belt_hold()`, `break_away()`, `closing_marubozu()`, `concealing_baby_swallow()`, `counter_attack()`, `dark_cloud_cover()`, `doji()`, `doji_star()`, `dragonfly_doji()`, `engulfing()`, `evening_doji_star()`, `evening_star()`, `gaps_side_white()`, `gravestone_doji()`, `hammer()`, `hanging_man()`, `harami()`, `harami_cross()`, `high_wave()`, `hikakke()`, `hikakke_mod()`, `homing_pigeon()`, `in_neck()`, `inverted_hammer()`, `kicking()`, `kicking_baby_length()`, `ladder_bottom()`, `long_legged_doji()`, `long_line()`, `marubozu()`, `mat_hold()`, `matching_low()`, `morning_doji_star()`, `morning_star()`, `on_neck()`, `rickshaw_man()`, `rise_fall_3_methods()`, `separating_lines()`, `shooting_star()`, `short_line()`, `spinning_top()`, `stalled_pattern()`, `stick_sandwich()`, `takuri()`, `tasuki_gap()`, `three_black_crows()`, `three_identical_crows()`, `three_inside()`, `three_line_strike()`, `three_outside()`, `three_stars_in_the_south()`, `three_white_soldiers()`, `thrusting()`, `tristar()`, `two_crows()`, `unique_3_river()`, `upside_gap_2_crows()`, `xside_gap_3_methods()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::piercing(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::piercing
  )
}
```

plus_directional_indicator

Plus Directional Indicator

Description

`plus_directional_indicator()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
plus_directional_indicator(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 3-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of length 1.
<code>na.bridge</code>	(<code>logical</code>). A <code>logical</code> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same `class` and `length` of `x`:

PLUS_DI `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::plus_directional_indicator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::plus_directional_indicator
```

```
)
}
```

```
plus_directional_movement
```

```
Plus Directional Movement
```

Description

`plus_directional_movement()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE (default)` The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
plus_directional_movement(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 2-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low</code> .

timePeriod (integer). Lookback period (window size). A positive integer of length 1.

na.bridge (logical). A logical of length 1. FALSE by default. When FALSE, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same class and length of x:

PLUS_DM double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `unsmoothed_chande_momentum_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::plus_directional_movement(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
```

```
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::plus_directional_movement
  )
}
```

positive_volume_index *Positive Volume Index*

Description

`positive_volume_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
positive_volume_index(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close + volume</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

PVI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [trading_volume\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::positive_volume_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
```

```
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::positive_volume_index
  )
}
```

ratio_of_change	Rate of change ratio: (price/prevPrice)
-----------------	---

Description

ratio_of_change() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

ratio_of_change() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
ratio_of_change(x, cols, timePeriod = 10, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

ROCR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::ratio_of_change(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::ratio_of_change
  )
}
```

relative_strength_index

Relative Strength Index

Description

`relative_strength_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`relative_strength_index()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
relative_strength_index(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

RSI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::relative_strength_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::relative_strength_index
  )
}
```

Description

`rickshaw_man()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
rickshaw_man(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLRICKSHAWMAN [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::rickshaw_man(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::rickshaw_man
  )
}
```

rise_fall_3_methods *Rising/Falling Three Methods*

Description

rise_fall_3_methods() is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
rise_fall_3_methods(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLRISEFALL3METHODS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::rise_fall_3_methods(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
```

```

talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::rise_fall_3_methods
)
}

```

rolling_beta

Beta

Description

rolling_beta() is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

`na.bridge = TRUE` Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_beta(x, y, timePeriod = 5, na.bridge = FALSE, ...)
```

Arguments

<code>x, y</code>	((<code>double</code>), (<code>double</code>)). A pair of <code>double</code> vectors of equal <code>length</code> , or single-column xts objects.
<code>timePeriod</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of <code>length</code> 1.
<code>na.bridge</code>	(<code>logical</code>). A <code>logical</code> of <code>length</code> 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters.

Value

A `double` vector (or single-column xts) with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: [rolling_correlation\(\)](#), [rolling_maximum\(\)](#), [rolling_minimum\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the rolling statistic
## between Open and Close
output <- talib::rolling_beta(x = BTC[[1]], y = BTC[[4]])

## display the results
utils::tail(output)
```

rolling_correlation	<i>Pearson's Correlation Coefficient (r)</i>
---------------------	---

Description

`rolling_correlation()` is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

`na.bridge = TRUE` Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_correlation(x, y, timePeriod = 30, na.bridge = FALSE, ...)
```

Arguments

x, y	((double), (double)). A pair of double vectors of equal length , or single-column xts objects.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
...	Additional parameters.

Value

A **double** vector (or single-column xts) with the same **length** of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: [rolling_beta\(\)](#), [rolling_maximum\(\)](#), [rolling_minimum\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the rolling statistic
## between Open and Close
output <- talib::rolling_correlation(x = BTC[[1]], y = BTC[[4]])

## display the results
utils::tail(output)
```

Description

rolling_maximum() is a generic S3 function that preserves the input **class**: **double** vector in, **double** vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on na.bridge:

na.bridge = FALSE (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

na.bridge = TRUE Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_maximum(x, timePeriod = 30, na.bridge = FALSE, ...)
```

Arguments

x	(double). A double vector, or a single-column xts.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
...	Additional parameters.

Value

A **double** vector (or single-column xts) with the same **length** of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_minimum\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## Open
output <- talib::rolling_maximum(x = BTC[[1]])

## display the results
utils::tail(output)
```

rolling_minimum	<i>Lowest value over a specified period</i>
-----------------	---

Description

rolling_minimum() is a generic S3 function that preserves the input **class**: **double** vector in, **double** vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on na.bridge:

na.bridge = FALSE (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

na.bridge = TRUE Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_minimum(x, timePeriod = 30, na.bridge = FALSE, ...)
```

Arguments

x	(double). A double vector, or a single-column xts.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
...	Additional parameters.

Value

A **double** vector (or single-column xts) with the same **length** of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: `rolling_beta()`, `rolling_correlation()`, `rolling_maximum()`, `rolling_standard_deviation()`, `rolling_sum()`, `rolling_variance()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## Open
output <- talib::rolling_minimum(x = BTC[[1]])

## display the results
utils::tail(output)
```

rolling_standard_deviation

Standard Deviation

Description

`rolling_standard_deviation()` is a generic S3 function that preserves the input **class**: **double** vector in, **double** vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

`na.bridge = TRUE` Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_standard_deviation(
  x,
  timePeriod = 5,
  deviations = 1,
  na.bridge = FALSE,
  ...
)
```

Arguments

x	(double). A double vector, or a single-column xts.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
deviations	(double). Number of deviations. Defaults to 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE, input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When TRUE, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
...	Additional parameters.

Value

A double vector (or single-column xts) with the same length of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_maximum\(\)](#), [rolling_minimum\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## Open
output <- talib::rolling_standard_deviation(x = BTC[[1]])

## display the results
utils::tail(output)
```

rolling_sum

*Summation***Description**

rolling_sum() is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

`na.bridge = TRUE` Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_sum(x, timePeriod = 30, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	(<code>double</code>). A <code>double</code> vector, or a single-column xts.
<code>timePeriod</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of <code>length</code> 1.
<code>na.bridge</code>	(<code>logical</code>). A <code>logical</code> of <code>length</code> 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters.

Value

A `double` vector (or single-column xts) with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: `rolling_beta()`, `rolling_correlation()`, `rolling_maximum()`, `rolling_minimum()`, `rolling_standard_deviation()`, `rolling_variance()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## Open
output <- talib::rolling_sum(x = BTC[[1]])

## display the results
utils::tail(output)
```

rolling_variance	<i>Variance</i>
------------------	-----------------

Description

rolling_variance() is a generic S3 function that preserves the input **class**: **double** vector in, **double** vector out; single-column xts in, xts out.

Handling of NA values:

Leading NAs are always produced for the initial lookback period where insufficient data is available. If the input itself contains NAs, the behaviour depends on na.bridge:

na.bridge = FALSE (**default**) NAs propagate through the TA-Lib C routine. Because rolling statistics smooth across time, a single NA in the input typically poisons every subsequent value.

na.bridge = TRUE Input NAs are stripped, the statistic is computed on the dense series, and NAs are re-inserted at the original positions. Output length matches input length, but the computation treats non-consecutive observations as if they were adjacent - fine for sparse missing values, misleading across clustered gaps.

Usage

```
rolling_variance(x, timePeriod = 5, deviations = 1, na.bridge = FALSE, ...)
```

Arguments

x	(double). A double vector, or a single-column xts.
timePeriod	(integer). Lookback period (window size). A positive integer of length 1.
deviations	(double). Number of deviations. Defaults to 1.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (the rolling computation typically fills the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the statistic to treat non-consecutive non-NA observations as if they were adjacent - see the Handling of NA values section above for the consequences.
...	Additional parameters.

Value

A [double](#) vector (or single-column xts) with the same [length](#) of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistics: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_maximum\(\)](#), [rolling_minimum\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## Open
output <- talib::rolling_variance(x = BTC[[1]])

## display the results
utils::tail(output)
```

separating_lines

Separating Lines

Description

`separating_lines()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
separating_lines(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSEPARATINGLINES [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#),

```

stalled_pattern(), stick_sandwich(), takuri(), tasuki_gap(), three_black_crows(), three_identical_crows(),
three_inside(), three_line_strike(), three_outside(), three_stars_in_the_south(), three_white_soldiers(),
thrusting(), tristar(), two_crows(), unique_3_river(), upside_gap_2_crows(), xside_gap_3_methods()

```

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::separating_lines(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::separating_lines
  )
}

```

set_theme

Set or List Chart Themes

Description

Apply a named color theme to the charting system, override individual theme properties, or list all available theme names.

Theme changes take effect immediately and apply to all subsequent `chart()` and `indicator()` calls.

Usage

```
set_theme(name, ...)
```

Arguments

name	An optional character string matching one of the available theme names. Partial matching is supported. If omitted (and no ... overrides are given), returns the available theme names instead.
...	Named color overrides applied after the base theme. Valid names include any theme property: bearish_body, bearish_wick, bearish_border, bullish_body, bullish_wick, bullish_border, background_color, foreground_color, text_color, gridcolor, threshold_color, and colorway (a character vector of up to 10 colors).

Details

set_theme supports three usage patterns:

set_theme() Returns a character vector of available theme names.

set_theme("payout") Applies the named theme.

set_theme\$payout Applies the named theme via \$ syntax (supports tab-completion in interactive sessions).

Themes can be combined with individual color overrides. When both name and ... are provided, the base theme is applied first, then the overrides are applied on top:

```
# Apply "payout" but with a custom background
set_theme("payout", background_color = "#000000")
```

It is also possible to override individual properties without selecting a theme:

```
# Change only the bearish candle color
set_theme(bearish_body = "#FF0000")
```

See [chart_themes](#) for a full description of all available themes and their color properties.

Value

When called without arguments, a [character](#) vector of available theme names. Otherwise, invisibly returns NULL; the theme is applied as a side effect to the internal chart-variables state.

See Also

[chart_themes](#) for descriptions of each theme, [chart\(\)](#) for creating charts.

Other Charting: [chart\(\)](#), [chart_themes](#), [indicator\(\)](#)

Examples

```
## list available themes
talib::set_theme()

## apply a theme by name
talib::set_theme("payout")

## apply a theme with custom overrides
talib::set_theme(
  "hawks_and_doves",
  background_color = "#FAFAFA"
)

## override individual properties
## without switching theme
talib::set_theme(
  bearish_body = "#FF4444",
  bullish_body = "#44FF44"
)

## reset to default theme
talib::set_theme("default")
```

shooting_star

Shooting Star

Description

`shooting_star()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a

series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
shooting_star(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSHOOTINGSTAR [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::shooting_star(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::shooting_star
  )
}
```

short_line

Short Line Candle

Description

short_line() is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

na.bridge = FALSE (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
short_line(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSHORTLINE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#),

```

spinning_top(), stalled_pattern(), stick_sandwich(), takuri(), tasuki_gap(), three_black_crows(),
three_identical_crows(), three_inside(), three_line_strike(), three_outside(), three_stars_in_the_south(),
three_white_soldiers(), thrusting(), tristar(), two_crows(), unique_3_river(), upside_gap_2_crows(),
xside_gap_3_methods()

```

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::short_line(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::short_line
  )
}

```

simple_moving_average *Simple Moving Average*

Description

simple_moving_average() is a generic S3 function that preserves the input **class**: **data.frame** in, **data.frame** out; **matrix** in, **matrix** out; **xts** in, **xts** out. Other inputs coercible to **data.frame** (e.g. **zoo**) are computed on the coerced **data.frame**.

simple_moving_average() also accepts a **double** vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a **double** vector; otherwise the full n by k **matrix** is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE (default)` The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data. frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
simple_moving_average(x, timePeriod = 30, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without 'x', [simple_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

SMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::simple_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
```

```

    talib::simple_moving_average
  )
}

```

sine_wave

Hilbert Transform - SineWave

Description

sine_wave() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

sine_wave() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE (default)` The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
sine_wave(x, cols, na.bridge = FALSE, ...)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

Sine [double](#)

LeadSine [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicators: [dominant_cycle_period\(\)](#), [dominant_cycle_phase\(\)](#), [phasor_components\(\)](#), [trend_cycle_mode\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::sine_wave(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
```

```
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::sine_wave
  )
}
```

spinning_top

Spinning Top

Description

spinning_top() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
spinning_top(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (`N = 5`, `alpha = 0.6`). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSPINNINGTOP [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::spinning_top(BTC)

## display the results
utils::tail(output)

## visualize the indicator
```

```
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::spinning_top
  )
}
```

SPY	<i>SPDR S&P 500 ETF (SPY)</i>
-----	-----------------------------------

Description

Daily OHLCV price data for the SPDR S&P 500 ETF (SPY), covering 2023-01-01 to 2024-12-31. A widely used benchmark for U.S. equity market performance and a common test case for technical analysis strategies.

Usage

SPY

Format

- A [matrix](#) with 501 rows and 5 columns.
- open** Opening price for the trading day.
- high** Highest price reached during the trading day.
- low** Lowest price reached during the trading day.
- close** Closing price for the trading day.
- volume** Total trading volume for the day.

Source

Loaded using [quantmod](#).

Examples

```
## Load the dataset
data(SPY, package = "talib")

## Compute Bollinger Bands on SPY
talib::bollinger_bands(SPY)
```

stalled_pattern

Stalled Pattern

Description

`stalled_pattern()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
stalled_pattern(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSTALLEDPATTERN [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::stalled_pattern(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```

```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::stalled_pattern
)
}
```

stick_sandwich

Stick Sandwich

Description

stick_sandwich() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
stick_sandwich(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (`N = 5`, `alpha = 0.6`). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLSTICKSANDWICH [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::stick_sandwich(BTC)

## display the results
utils::tail(output)

## visualize the indicator
```

```
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::stick_sandwich
  )
}
```

stochastic

Stochastic

Description

`stochastic()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
stochastic(
  x,
  cols,
  fastKPeriod = 5,
  slowKPeriod = 3,
  slowKMa = 0,
  slowDPeriod = 3,
  slowDMa = 0,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>fastKPeriod</code>	(integer). Time period for building the Fast-K line. Defaults to 5.
<code>slowKPeriod</code>	(integer). Smoothing for making the Slow-K line. Usually set to 3. Defaults to 3.
<code>slowKMa</code>	(integer). Type of Moving Average for Slow-K. Defaults to 0 (SMA). Can also be passed as <code>talib::SMA</code> .
<code>slowDPeriod</code>	(integer). Smoothing for making the Slow-D line. Defaults to 3.
<code>slowDMa</code>	(integer). Type of Moving Average for Slow-D. Defaults to 0 (SMA). Can also be passed as <code>talib::SMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

SlowK [double](#)

SlowD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::stochastic(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::stochastic
  )
}
```

stochastic_relative_strength_index

Stochastic Relative Strength Index

Description

`stochastic_relative_strength_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`stochastic_relative_strength_index()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
stochastic_relative_strength_index(
  x,
  cols,
  timePeriod = 14,
  fastKPeriod = 5,
  fastDPeriod = 3,
  fastDMa = 0,
  na.bridge = FALSE,
  ...
)
```

Arguments

`x` An OHLC-V series coercible to `data.frame`. Alternatively, `x` may also be supplied as a `double` vector.

<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>fastKPeriod</code>	(integer). Time period for building the Fast-K line. Defaults to 5.
<code>fastDPeriod</code>	(integer). Smoothing for making the Fast-D line. Usually set to 3. Defaults to 3.
<code>fastDMA</code>	(integer). Type of Moving Average for Fast-D. Defaults to 0 (SMA). Can also be passed as <code>talib::SMA</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

FastK [double](#)

FastD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")
```

```

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::stochastic_relative_strength_index(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::stochastic_relative_strength_index
  )
}

```

t3_exponential_moving_average

Triple Exponential Moving Average (T3)

Description

t3_exponential_moving_average() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

t3_exponential_moving_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full n by k `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
t3_exponential_moving_average(
  x,
  timePeriod = 5,
  volumeFactor = 0.7,
  cols,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1 .
<code>volumeFactor</code>	(double). Volume Factor. Defaults to 0.7.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1 . FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without `'x'`, `t3_exponential_moving_average` functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

T3 [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::t3_exponential_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::t3_exponential_moving_average
  )
}
```

takuri

Takuri (Dragonfly Doji with very long lower shadow)

Description

`takuri()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
takuri(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLTAKURI [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::takuri(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```



```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::takuri
)
}
```

tasuki_gap

Tasuki Gap

Description

tasuki_gap() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
tasuki_gap(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (`N = 5`, `alpha = 0.6`). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLTASUKIGAP [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::tasuki_gap(BTC)

## display the results
utils::tail(output)

## visualize the indicator
```

```
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::tasuki_gap
  )
}
```

three_black_crows	<i>Three Black Crows</i>
-------------------	--------------------------

Description

`three_black_crows()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_black_crows(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (`N = 10`, `alpha = 0.1`). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (`N = 5`, `alpha = 0.2`). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3BLACKCROWS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_black_crows(BTC)

## display the results
```

```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_black_crows
  )
}

```

three_identical_crows *Identical Three Crows*

Description

three_identical_crows() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_identical_crows(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~open + high + low + close.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means "<=20% of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLIDENTICAL3CROWS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_identical_crows(BTC)

## display the results
```

```

utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_identical_crows
  )
}

```

three_inside

Three Inside Up/Down

Description

`three_inside()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless

across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_inside(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (`N = 10`, `alpha = 1.0`). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (`N = 10`, `alpha = 3.0`). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (`N = 10`, `alpha = 1.0`). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (`N = 10`, `alpha = 0.1`). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (`N = 0`, `alpha = 1.0`). Shadow is long when it's longer than the real body.

ShadowVeryLong (`N = 0`, `alpha = 2.0`). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (`N = 0`, `alpha = 1.0`). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3INSIDE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")
```

```

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_inside(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_inside
  )
}

```

three_line_strike	<i>Three-Line Strike</i>
-------------------	--------------------------

Description

`three_line_strike()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a

series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_line_strike(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3LINESTRIKE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_line_strike(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_line_strike
  )
}
```

three_outside

Three Outside Up/Down

Description

`three_outside()` is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; [xts](#) in, [xts](#) out. Other inputs coercible to [data.frame](#) (e.g. [zoo](#)) are computed on the coerced [data.frame](#).

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_outside(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3OUTSIDE [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#),

```
short_line(), spinning_top(), stalled_pattern(), stick_sandwich(), takuri(), tasuki_gap(),
three_black_crows(), three_identical_crows(), three_inside(), three_line_strike(),
three_stars_in_the_south(), three_white_soldiers(), thrusting(), tristar(), two_crows(),
unique_3_river(), upside_gap_2_crows(), xside_gap_3_methods()
```

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_outside(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_outside
  )
}
```

three_stars_in_the_south

Three Stars In The South

Description

three_stars_in_the_south() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_stars_in_the_south(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3STARSINSOUTH [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_stars_in_the_south(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_stars_in_the_south
  )
}
```

Description

`three_white_soldiers()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
three_white_soldiers(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of <i>length</i> 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL3WHITESOLDIERS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::three_white_soldiers(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::three_white_soldiers
  )
}
```

thrusting

Thrusting Pattern

Description

thrusting() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
thrusting(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLTHRUSTING [integer](#)

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::thrusting(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```

```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::thrusting
)
}
```

trading_volume

Trading Volume

Description

trading_volume() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
trading_volume(
  x,
  cols,
```

```

maType = list(SMA(timePeriod = 7), SMA(timePeriod = 15)),
na.bridge = FALSE,
...
)

```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~volume + open + close</code> .
<code>maType</code>	A list of <code>maType</code> specifications on the form <code>SMA(timePeriod = 7)</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

VOLUME [double](#)

SMA7 [double](#)

SMA15 [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicators: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [chaikin_money_flow\(\)](#), [negative_volume_index\(\)](#), [on_balance_volume\(\)](#), [percentage_volume_oscillator\(\)](#), [positive_volume_index\(\)](#)

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator

```

```
## for Bitcoin (BTC)
output <- talib::trading_volume(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::trading_volume
  )
}
```

trendline

Hilbert Transform - Instantaneous Trendline

Description

`trendline()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`trendline()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length

with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
trendline(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

HT_TRENDLINE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#)


```
triangular_moving_average(), triple_exponential_moving_average(), variable_moving_average_period(),
volume_weighted_moving_average(), weighted_moving_average())
```

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::trendline(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::trendline
  )
}
```

trend_cycle_mode

Hilbert Transform - Trend vs Cycle Mode

Description

trend_cycle_mode() is a generic S3 function that preserves the input [class: data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

trend_cycle_mode() also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the na.bridge argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data `.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
trend_cycle_mode(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

HT_TRENDMODE [integer](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicators: [dominant_cycle_period\(\)](#), [dominant_cycle_phase\(\)](#), [phasor_components\(\)](#), [sine_wave\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::trend_cycle_mode(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::trend_cycle_mode
  )
}
```

triangular_moving_average*Triangular Moving Average*

Description

`triangular_moving_average()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out; xts in, xts out. Other inputs coercible to [data.frame](#) (e.g. zoo) are computed on the coerced [data.frame](#).

`triangular_moving_average()` also accepts a [double](#) vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a [double](#) vector; otherwise the full n by k [matrix](#) is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
triangular_moving_average(x, timePeriod = 30, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without 'x', [triangular_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

TRIMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::triangular_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
```

```

    talib::triangular_moving_average
  )
}

```

triple_exponential_average

1-day Rate-Of-Change (ROC) of a Triple Smooth EMA

Description

triple_exponential_average() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

triple_exponential_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
triple_exponential_average(x, cols, timePeriod = 30, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

TRIX [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
```

```

output <- talib::triple_exponential_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::triple_exponential_average
  )
}

```

triple_exponential_moving_average

Triple Exponential Moving Average

Description

triple_exponential_moving_average() is a generic S3 function that preserves the input **class**: **data.frame** in, **data.frame** out; **matrix** in, **matrix** out; **xts** in, **xts** out. Other inputs coercible to **data.frame** (e.g. zoo) are computed on the coerced **data.frame**.

triple_exponential_moving_average() also accepts a **double** vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a **double** vector; otherwise the full n by k **matrix** is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length

with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
triple_exponential_moving_average(
  x,
  timePeriod = 30,
  cols,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

When passed without `'x'`, `triple_exponential_moving_average` functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without `'x'` it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

TEMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [variable_moving_average_period\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::triple_exponential_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::triple_exponential_moving_average
  )
}
```

tristar

Tristar Pattern

Description

tristar() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
tristar(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining

output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N *integer*. Controls the number of candles to consider when identifying patterns.

alpha *double*. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same *class* and *length* of x:

CDLTRISTAR *integer*

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::tristar(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
```

```

    talib::tristar
  )
}

```

true_range

True Range

Description

true_range() is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
true_range(x, cols, na.bridge = FALSE, ...)
```

Arguments

x An OHLC-V series coercible to `data.frame`.

cols (*formula*). An optional 3-variable *formula* selecting columns from `x` via `model.frame`. For `xts` input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches `TICKER.Close`), case-insensitively - and columns are consumed in formula order. Defaults to `~high + low + close`.

na.bridge (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Value

An object of same **class** and **length** of x:

TRANGE double

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicators: `average_true_range()`, `normalized_average_true_range()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::true_range(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::true_range
  )
}
```

two_crows

Two Crows

Description

`two_crows()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
two_crows(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .

`na.bridge` (logical). A logical of length 1. **FALSE** by default. When **FALSE**, input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N integer. Controls the number of candles to consider when identifying patterns.

alpha double. A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDL2CROWS [integer](#)

Pattern codes depend on options(talib.normalize):

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [unique_3_river\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::two_crows(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
```

```
## series with talib::chart()
talib::chart(BTC)

## chart indicator
## with default values
talib::indicator(
  talib::two_crows
)
}
```

typical_price

Typical Price

Description

`typical_price()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
typical_price(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

TYPPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_deviation\(\)](#), [average_price\(\)](#), [median_price\(\)](#), [weighted_close_price\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::typical_price(BTC)

## display the results
utils::tail(output)
```

ultimate_oscillator *Ultimate Oscillator*

Description

ultimate_oscillator() is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
ultimate_oscillator(
  x,
  cols,
  firstPeriod = 7,
  secondPeriod = 14,
  thirdPeriod = 28,
  na.bridge = FALSE,
  ...
)
```

Arguments

`x` An OHLC-V series coercible to `data.frame`.

cols	(formula). An optional 3-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~high + low + close.
firstPeriod	(integer). Number of bars for 1st period. Defaults to 7.
secondPeriod	(integer). Number of bars for 2nd period. Defaults to 14.
thirdPeriod	(integer). Number of bars for 3rd period. Defaults to 28.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

ULTOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#), [williams_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::ultimate_oscillator(BTC)
```

```
## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::ultimate_oscillator
  )
}
```

unique_3_river

Unique 3 River

Description

unique_3_river() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend

encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
unique_3_river(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLUNIQUE3RIVER [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [upside_gap_2_crows\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")
```

```
## calculate the indicator
## for Bitcoin (BTC)
output <- talib::unique_3_river(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::unique_3_river
  )
}
```

unsmoothed_chande_momentum_oscillator

Chande Momentum Oscillator (Unsmoothed)

Description

`unsmoothed_chande_momentum_oscillator()` is a generic S3 function that preserves the input **class**: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

`unsmoothed_chande_momentum_oscillator()` also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
unsmoothed_chande_momentum_oscillator(
  x,
  cols,
  timePeriod = 14,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame . Alternatively, <code>x</code> may also be supplied as a double vector.
<code>cols</code>	(formula). An optional 1-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

CMOU [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: `absolute_price_oscillator()`, `aroon()`, `aroon_oscillator()`, `average_directional_movement_index()`, `average_directional_movement_index_rating()`, `balance_of_power()`, `chande_momentum_oscillator()`, `commodity_channel_index()`, `directional_movement_index()`, `extended_moving_average_convergence_divergence()`, `fast_stochastic()`, `fixed_moving_average_convergence_divergence()`, `intraday_movement_index()`, `minus_directional_indicator()`, `minus_directional_movement()`, `momentum()`, `money_flow_index()`, `moving_average_convergence_divergence()`, `percentage_price_oscillator()`, `plus_directional_indicator()`, `plus_directional_movement()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `triple_exponential_average()`, `ultimate_oscillator()`, `williams_oscillator()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::unsmoothed_chande_momentum_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::unsmoothed_chande_momentum_oscillator
  )
}
```

Description

`upside_gap_2_crows()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
upside_gap_2_crows(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> .
<code>cols</code>	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from <code>x</code> via <code>model.frame</code> . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (<code>close</code> matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(<i>logical</i>). A <i>logical</i> of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into <code>model.frame</code> . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See `vignette("candlestick")` for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLUPSIDEGAP2CROWS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#), [short_line\(\)](#), [spinning_top\(\)](#), [stalled_pattern\(\)](#), [stick_sandwich\(\)](#), [takuri\(\)](#), [tasuki_gap\(\)](#), [three_black_crows\(\)](#), [three_identical_crows\(\)](#), [three_inside\(\)](#), [three_line_strike\(\)](#), [three_outside\(\)](#), [three_stars_in_the_south\(\)](#), [three_white_soldiers\(\)](#), [thrusting\(\)](#), [tristar\(\)](#), [two_crows\(\)](#), [unique_3_river\(\)](#), [xside_gap_3_methods\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::upside_gap_2_crows(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::upside_gap_2_crows
  )
}
```

variable_moving_average_period

Moving average with variable period

Description

variable_moving_average_period() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

variable_moving_average_period() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
variable_moving_average_period(
  x,
  cols,
  periods,
  minimumPeriod = 2,
  maximumPeriod = 30,
  maType = 0,
  na.bridge = FALSE,
```



```
    ...  
)
```

Arguments

x	An OHLC-V series coercible to data.frame . Alternatively, x may also be supplied as a double vector.
cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
periods	(numeric). Vector of periods, one per observation of the input series.
minimumPeriod	(integer). Value less than minimum will be changed to Minimum period. Defaults to 2.
maximumPeriod	(integer). Value higher than maximum will be changed to Maximum period. Defaults to 30.
maType	(integer). Type of Moving Average. Defaults to 0 (SMA). Can also be passed as talib::SMA.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of x:

MAVP [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [volume_weighted_moving_average\(\)](#), [weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## define a random series
## of periods to evaluate
## each candle
periods <- runif(
  n = nrow(BTC),
  min = 5,
  max = 10
)

## calculate the indicator
## for Bitcoin (BTC)
utils::tail(
  talib::variable_moving_average_period(
    BTC,
    periods = periods
  )
)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::variable_moving_average_period,
    periods = periods
  )
}
```

volume_weighted_moving_average

Volume Weighted Moving Average

Description

volume_weighted_moving_average() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
volume_weighted_moving_average(
  x,
  volume,
  timePeriod = 30,
  cols,
  na.bridge = FALSE,
  ...
)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>volume</code>	(numeric). The volume series. Defaults to the 'volume' column of the 'cols' selection.
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~close + volume</code> .
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining

output with NA). When **TRUE**, input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the **Handling of NA values** section above for the consequences.

... Additional parameters passed into `model.frame`. Unused for xts input (a warning is emitted when supplied).

Details

When passed without 'x', `volume_weighted_moving_average` functions as an 'Moving Average'-specification which is used in, for example, `stochastic` when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same `class` and `length` of x:

VWMA `double`

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: `acceleration_bands()`, `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `extended_parabolic_stop_and_reverse()`, `hull_moving_average()`, `kaufman_adaptive_moving_average()`, `mesa_adaptive_moving_average()`, `midpoint_period()`, `midpoint_price()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_moving_average()`, `variable_moving_average()`, `weighted_moving_average()`

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::volume_weighted_moving_average(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
```

```
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::volume_weighted_moving_average
  )
}
```

weighted_close_price	<i>Weighted Close Price</i>
----------------------	-----------------------------

Description

`weighted_close_price()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
weighted_close_price(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches <code>TICKER.Close</code>), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

WCLPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_deviation\(\)](#), [average_price\(\)](#), [median_price\(\)](#), [typical_price\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::weighted_close_price(BTC)

## display the results
utils::tail(output)
```

weighted_moving_average

Weighted Moving Average

Description

weighted_moving_average() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

weighted_moving_average() also accepts a `double` vector, in which case the indicator is calculated directly without column selection. When the result has a single column it is simplified to a `double` vector; otherwise the full `n` by `k` `matrix` is returned.

Handling of NA values:

Every indicator always emits **leading** NAs for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source data.frame by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
weighted_moving_average(x, timePeriod = 30, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to <code>data.frame</code> . Alternatively, <code>x</code> may also be supplied as a <code>double</code> vector.
<code>timePeriod</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of <code>length</code> 1.

cols	(formula). An optional 1-variable formula selecting columns from x via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to ~close.
na.bridge	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
...	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Details

When passed without 'x', [weighted_moving_average](#) functions as an 'Moving Average'-specification which is used in, for example, [stochastic](#) when constructing the smoothing lines.

When called without 'x' it will return a named list which is used for the indicators that supports various Moving Average specifications.

Value

An object of same [class](#) and [length](#) of x:

WMA [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Studies: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [hull_moving_average\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [midpoint_period\(\)](#), [midpoint_price\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_average\(\)](#), [variable_moving_average\(\)](#), [volume_weighted_moving_average\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::weighted_moving_average(BTC)
```



```

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::weighted_moving_average
  )
}

```

williams_oscillator	<i>Williams' %R</i>
---------------------	---------------------

Description

`williams_oscillator()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend

encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
williams_oscillator(x, cols, timePeriod = 14, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 3-variable formula selecting columns from <code>x</code> via model.frame . For xts input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~high + low + close</code> .
<code>timePeriod</code>	(integer). Lookback period (window size). A positive integer of length 1.
<code>na.bridge</code>	(logical). A logical of length 1. FALSE by default. When FALSE , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When TRUE , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for xts input (a warning is emitted when supplied).

Value

An object of same [class](#) and [length](#) of `x`:

WILLR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicators: [absolute_price_oscillator\(\)](#), [aroon\(\)](#), [aroon_oscillator\(\)](#), [average_directional_movement_index\(\)](#), [average_directional_movement_index_rating\(\)](#), [balance_of_power\(\)](#), [chande_momentum_oscillator\(\)](#), [commodity_channel_index\(\)](#), [directional_movement_index\(\)](#), [extended_moving_average_convergence_divergence\(\)](#), [fast_stochastic\(\)](#), [fixed_moving_average_convergence_divergence\(\)](#), [intraday_movement_index\(\)](#), [minus_directional_indicator\(\)](#), [minus_directional_movement\(\)](#), [momentum\(\)](#), [money_flow_index\(\)](#), [moving_average_convergence_divergence\(\)](#), [percentage_price_oscillator\(\)](#), [plus_directional_indicator\(\)](#), [plus_directional_movement\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_oscillator\(\)](#), [unsmoothed_chande_momentum_oscillator\(\)](#)

Examples

```
## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::williams_oscillator(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::williams_oscillator
  )
}
```

xside_gap_3_methods *Upside/Downside Gap Three Methods*

Description

`xside_gap_3_methods()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out; `xts` in, `xts` out. Other inputs coercible to `data.frame` (e.g. `zoo`) are computed on the coerced `data.frame`.

Handling of NA values:

Every indicator always emits **leading NAs** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how NAs already present in the input are handled, which is controlled by the `na.bridge` argument:

`na.bridge = FALSE` (**default**) The input is passed to the underlying TA-Lib C routine as-is. Because most indicators smooth across time (EMA, RSI, MACD, Bollinger Bands, ...), a single NA in the input typically propagates forward and **poisons every subsequent value** - it is common for one missing observation to produce an output that is entirely NA from that position onward. This mode is the right choice when you want to *see* the missing data in the output rather than silently compute around it.

`na.bridge = TRUE` NA rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with NA inserted at every position the input had NA. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.

Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as NA) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

```
xside_gap_3_methods(x, cols, na.bridge = FALSE, ...)
```

Arguments

<code>x</code>	An OHLC-V series coercible to data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . For <code>xts</code> input each formula variable is matched against the column names directly - exact matches first, then quantmod-style suffix matches (close matches TICKER.Close), case-insensitively - and columns are consumed in formula order. Defaults to <code>~open + high + low + close</code> .
<code>na.bridge</code>	(logical). A logical of length 1. <code>FALSE</code> by default. When <code>FALSE</code> , input NAs propagate through the TA-Lib C routine (most indicators will fill the remaining output with NA). When <code>TRUE</code> , input NA rows are stripped before computation and re-inserted at the original positions in the output, causing the indicator to treat non-consecutive non-NA observations as if they were adjacent — see the Handling of NA values section above for the consequences.
<code>...</code>	Additional parameters passed into model.frame . Unused for <code>xts</code> input (a warning is emitted when supplied).

Details

General options for candlestick pattern recognition:

N [integer](#). Controls the number of candles to consider when identifying patterns.

alpha [double](#). A sensitivity parameter when identifying patterns.

Available options and their defaults:

BodyLong (N = 10, alpha = 1.0). Real body is long when it's longer than the average of the 10 previous candles' real body.

BodyVeryLong (N = 10, alpha = 3.0). Real body is very long when it's longer than 3 times the average of the 10 previous candles' real body.

BodyShort (N = 10, alpha = 1.0). Real body is short when it's shorter than the average of the 10 previous candles' real bodies.

BodyDoji (N = 10, alpha = 0.1). Real body is like doji's body when it's shorter than 10% the average of the 10 previous candles' high-low range.

ShadowLong (N = 0, alpha = 1.0). Shadow is long when it's longer than the real body.

ShadowVeryLong (N = 0, alpha = 2.0). Shadow is very long when it's longer than 2 times the real body.

ShadowShort (N = 0, alpha = 1.0). Shadow is short when it's shorter than half the average of the 10 previous candles' sum of shadows.

ShadowVeryShort (N = 10, alpha = 0.1). Shadow is very short when it's shorter than 10% the average of the 10 previous candles' high-low range.

Near (N = 5, alpha = 0.2). When measuring distance between parts of candles or width of gaps "near" means " $\leq 20\%$ of the average of the 5 previous candles high low range."

Far (N = 5, alpha = 0.6). When measuring distance between parts of candles or width of gaps "far" means " $\geq 60\%$ of the average of the 5 previous candles high-low range."

Equal (N = 5, alpha = 0.05). When measuring distance between parts of candles or width of gaps "equal" means " $\leq 5\%$ of the average of the 5 previous candles high-low range."

The options can be modified by running `options(talib.BodyLong.N = 5, talib.BodyLong.alpha = 0.2)`. See vignette("candlestick") for more details.

Value

An object of same [class](#) and [length](#) of x:

CDLXSIDEGAP3METHODS [integer](#)

Pattern codes depend on `options(talib.normalize)`:

- If TRUE: 1 = identified pattern; -1 = identified bearish pattern.
- If FALSE: 100 = identified pattern; -100 = identified bearish pattern.
- 0 = no pattern.

Author(s)

Serkan Korkmaz

See Also

Other Pattern Recognition: [abandoned_baby\(\)](#), [advance_block\(\)](#), [belt_hold\(\)](#), [break_away\(\)](#), [closing_marubozu\(\)](#), [concealing_baby_swallow\(\)](#), [counter_attack\(\)](#), [dark_cloud_cover\(\)](#), [doji\(\)](#), [doji_star\(\)](#), [dragonfly_doji\(\)](#), [engulfing\(\)](#), [evening_doji_star\(\)](#), [evening_star\(\)](#), [gaps_side_white\(\)](#), [gravestone_doji\(\)](#), [hammer\(\)](#), [hanging_man\(\)](#), [harami\(\)](#), [harami_cross\(\)](#), [high_wave\(\)](#), [hikakke\(\)](#), [hikakke_mod\(\)](#), [homing_pigeon\(\)](#), [in_neck\(\)](#), [inverted_hammer\(\)](#), [kicking\(\)](#), [kicking_baby_length\(\)](#), [ladder_bottom\(\)](#), [long_legged_doji\(\)](#), [long_line\(\)](#), [marubozu\(\)](#), [mat_hold\(\)](#), [matching_low\(\)](#), [morning_doji_star\(\)](#), [morning_star\(\)](#), [on_neck\(\)](#), [piercing\(\)](#), [rickshaw_man\(\)](#), [rise_fall_3_methods\(\)](#), [separating_lines\(\)](#), [shooting_star\(\)](#),

```

short_line(), spinning_top(), stalled_pattern(), stick_sandwich(), takuri(), tasuki_gap(),
three_black_crows(), three_identical_crows(), three_inside(), three_line_strike(),
three_outside(), three_stars_in_the_south(), three_white_soldiers(), thrusting(), tristar(),
two_crows(), unique_3_river(), upside_gap_2_crows()

```

Examples

```

## load Bitcoin (BTC)
## series
data(BTC, package = "talib")

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::xside_gap_3_methods(BTC)

## display the results
utils::tail(output)

## visualize the indicator
## with talib::chart()
##
## see ?talib::chart or ?talib::indicator
## for more details
{
  ## chart OHLC-V
  ## series with talib::chart()
  talib::chart(BTC)

  ## chart indicator
  ## with default values
  talib::indicator(
    talib::xside_gap_3_methods
  )
}

```

Index

- * **Charting**
 - chart, [51](#)
 - chart_themes, [54](#)
 - indicator, [145](#)
 - set_theme, [263](#)
- * **Cycle Indicators**
 - dominant_cycle_period, [79](#)
 - dominant_cycle_phase, [81](#)
 - phasor_components, [228](#)
 - sine_wave, [274](#)
 - trend_cycle_mode, [329](#)
- * **Financial Data**
 - ATOM, [20](#)
 - BTC, [42](#)
 - GOOGL, [114](#)
 - NVDA, [215](#)
 - SPY, [279](#)
- * **Momentum Indicators**
 - absolute_price_oscillator, [8](#)
 - aroon, [16](#)
 - aroon_oscillator, [18](#)
 - average_directional_movement_index, [23](#)
 - average_directional_movement_index_rating, [25](#)
 - balance_of_power, [31](#)
 - chande_momentum_oscillator, [49](#)
 - commodity_channel_index, [59](#)
 - directional_movement_index, [70](#)
 - extended_moving_average_convergence_divergence, [100](#)
 - fast_stochastic, [106](#)
 - fixed_moving_average_convergence_divergence, [108](#)
 - intraday_movement_index, [147](#)
 - minus_directional_indicator, [193](#)
 - minus_directional_movement, [195](#)
 - momentum, [197](#)
 - money_flow_index, [199](#)
 - moving_average_convergence_divergence, [208](#)
 - percentage_price_oscillator, [223](#)
 - plus_directional_indicator, [233](#)
 - plus_directional_movement, [236](#)
 - ratio_of_change, [240](#)
 - relative_strength_index, [242](#)
 - stochastic, [286](#)
 - stochastic_relative_strength_index, [288](#)
 - triple_exponential_average, [334](#)
 - ultimate_oscillator, [349](#)
 - unsmoothed_chande_momentum_oscillator, [354](#)
 - williams_oscillator, [369](#)
- * **OHLCV**
 - ATOM, [20](#)
 - BTC, [42](#)
 - GOOGL, [114](#)
 - NVDA, [215](#)
 - SPY, [279](#)
- * **Overlap Studies**
 - acceleration_bands, [11](#)
 - bollinger_bands, [36](#)
 - double_exponential_moving_average, [83](#)
 - exponential_moving_average, [98](#)
 - extended_parabolic_stop_and_reverse, [103](#)
 - hull_moving_average, [143](#)
 - kaufman_adaptive_moving_average, [156](#)
 - mesa_adaptive_moving_average, [186](#)
 - midpoint_period, [188](#)
 - midpoint_price, [191](#)
 - parabolic_stop_and_reverse, [221](#)
 - simple_moving_average, [271](#)
 - t3_exponential_moving_average, [291](#)
 - trendline, [327](#)

- triangular_moving_average, 331
- triple_exponential_moving_average, 336
- variable_moving_average_period, 360
- volume_weighted_moving_average, 362
- weighted_moving_average, 367
- * Pattern Recognition**
 - abandoned_baby, 5
 - advance_block, 13
 - belt_hold, 33
 - break_away, 39
 - closing_marubozu, 56
 - concealing_baby_swallow, 61
 - counter_attack, 64
 - dark_cloud_cover, 67
 - doji, 73
 - doji_star, 76
 - dragonfly_doji, 85
 - engulfing, 89
 - evening_doji_star, 92
 - evening_star, 95
 - gaps_side_white, 111
 - gravestone_doji, 115
 - hammer, 118
 - hanging_man, 121
 - harami, 124
 - harami_cross, 127
 - high_wave, 130
 - hikakke, 133
 - hikakke_mod, 137
 - homing_pigeon, 140
 - in_neck, 153
 - inverted_hammer, 149
 - kicking, 158
 - kicking_baby_length, 161
 - ladder_bottom, 165
 - long_legged_doji, 168
 - long_line, 171
 - marubozu, 175
 - mat_hold, 181
 - matching_low, 178
 - morning_doji_star, 202
 - morning_star, 205
 - on_neck, 218
 - piercing, 230
 - rickshaw_man, 244
 - rise_fall_3_methods, 248
 - separating_lines, 260
 - shooting_star, 265
 - short_line, 268
 - spinning_top, 276
 - stalled_pattern, 280
 - stick_sandwich, 283
 - takuri, 294
 - tasuki_gap, 297
 - three_black_crows, 300
 - three_identical_crows, 303
 - three_inside, 306
 - three_line_strike, 309
 - three_outside, 312
 - three_stars_in_the_south, 315
 - three_white_soldiers, 318
 - thrusting, 322
 - tristar, 339
 - two_crows, 344
 - unique_3_river, 351
 - upside_gap_2_crows, 356
 - xside_gap_3_methods, 371
- * Price Transform**
 - average_deviation, 21
 - average_price, 27
 - median_price, 184
 - typical_price, 347
 - weighted_close_price, 365
- * Rolling Statistics**
 - rolling_beta, 251
 - rolling_correlation, 252
 - rolling_maximum, 253
 - rolling_minimum, 255
 - rolling_standard_deviation, 256
 - rolling_sum, 258
 - rolling_variance, 259
- * Utility**
 - lookback, 174
- * Volatility Indicators**
 - average_true_range, 29
 - normalized_average_true_range, 213
 - true_range, 342
- * Volume Indicators**
 - chaikin_accumulation_distribution_line, 43
 - chaikin_accumulation_distribution_oscillator, 45
 - chaikin_money_flow, 47

- negative_volume_index, 210
- on_balance_volume, 215
- percentage_volume_oscillator, 226
- positive_volume_index, 238
- trading_volume, 325
- * algorithmic trading**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 11
 - advance_block, 13
 - aroon, 16
 - aroon_oscillator, 18
 - average_deviation, 21
 - average_directional_movement_index, 23
 - average_directional_movement_index_rating, 25
 - average_price, 27
 - average_true_range, 29
 - balance_of_power, 31
 - belt_hold, 33
 - bollinger_bands, 36
 - break_away, 39
 - chaikin_accumulation_distribution_line, 43
 - chaikin_accumulation_distribution_oscillator, 45
 - chaikin_money_flow, 47
 - chande_momentum_oscillator, 49
 - closing_marubozu, 56
 - commodity_channel_index, 59
 - concealing_baby_swallow, 61
 - counter_attack, 64
 - dark_cloud_cover, 67
 - directional_movement_index, 70
 - doji, 73
 - doji_star, 76
 - dominant_cycle_period, 79
 - dominant_cycle_phase, 81
 - double_exponential_moving_average, 83
 - dragonfly_doji, 85
 - engulfing, 89
 - evening_doji_star, 92
 - evening_star, 95
 - exponential_moving_average, 98
 - extended_moving_average_convergence_divergence, 100
 - extended_parabolic_stop_and_reverse, 103
 - fast_stochastic, 106
 - fixed_moving_average_convergence_divergence, 108
 - gaps_side_white, 111
 - gravestone_doji, 115
 - hammer, 118
 - hanging_man, 121
 - harami, 124
 - harami_cross, 127
 - high_wave, 130
 - hikakke, 133
 - hikakke_mod, 137
 - homing_pigeon, 140
 - hull_moving_average, 143
 - in_neck, 153
 - intraday_movement_index, 147
 - inverted_hammer, 149
 - kaufman_adaptive_moving_average, 156
 - kicking, 158
 - kicking_baby_length, 161
 - ladder_bottom, 165
 - long_legged_doji, 168
 - long_line, 171
 - lookback, 174
 - marubozu, 175
 - mat_hold, 181
 - matching_low, 178
 - median_price, 184
 - mesa_adaptive_moving_average, 186
 - midpoint_period, 188
 - midpoint_price, 191
 - minus_directional_indicator, 193
 - minus_directional_movement, 195
 - momentum, 197
 - money_flow_index, 199
 - morning_doji_star, 202
 - morning_star, 205
 - moving_average_convergence_divergence, 208
 - negative_volume_index, 210
 - normalized_average_true_range, 213
 - on_balance_volume, 215
 - on_neck, 218
 - parabolic_stop_and_reverse, 221
 - percentage_price_oscillator, 223

- percentage_volume_oscillator, 226
- phasor_components, 228
- piercing, 230
- plus_directional_indicator, 233
- plus_directional_movement, 236
- positive_volume_index, 238
- ratio_of_change, 240
- relative_strength_index, 242
- rickshaw_man, 244
- rise_fall_3_methods, 248
- rolling_beta, 251
- rolling_correlation, 252
- rolling_maximum, 253
- rolling_minimum, 255
- rolling_standard_deviation, 256
- rolling_sum, 258
- rolling_variance, 259
- separating_lines, 260
- shooting_star, 265
- short_line, 268
- simple_moving_average, 271
- sine_wave, 274
- spinning_top, 276
- stalled_pattern, 280
- stick_sandwich, 283
- stochastic, 286
- stochastic_relative_strength_index, 288
- t3_exponential_moving_average, 291
- takuri, 294
- tasuki_gap, 297
- three_black_crows, 300
- three_identical_crows, 303
- three_inside, 306
- three_line_strike, 309
- three_outside, 312
- three_stars_in_the_south, 315
- three_white_soldiers, 318
- thrusting, 322
- trading_volume, 325
- trend_cycle_mode, 329
- trendline, 327
- triangular_moving_average, 331
- triple_exponential_average, 334
- triple_exponential_moving_average, 336
- tristar, 339
- true_range, 342
- two_crows, 344
- typical_price, 347
- ultimate_oscillator, 349
- unique_3_river, 351
- unsmoothed_chande_momentum_oscillator, 354
- upside_gap_2_crows, 356
- variable_moving_average_period, 360
- volume_weighted_moving_average, 362
- weighted_close_price, 365
- weighted_moving_average, 367
- williams_oscillator, 369
- xside_gap_3_methods, 371
- * datasets**
 - ATOM, 20
 - BTC, 42
 - GOOGL, 114
 - NVDA, 215
 - SPY, 279
- * finance**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 11
 - advance_block, 13
 - aroon, 16
 - aroon_oscillator, 18
 - average_deviation, 21
 - average_directional_movement_index, 23
 - average_directional_movement_index_rating, 25
 - average_price, 27
 - average_true_range, 29
 - balance_of_power, 31
 - belt_hold, 33
 - bollinger_bands, 36
 - break_away, 39
 - chaikin_accumulation_distribution_line, 43
 - chaikin_accumulation_distribution_oscillator, 45
 - chaikin_money_flow, 47
 - chande_momentum_oscillator, 49
 - closing_marubozu, 56
 - commodity_channel_index, 59
 - concealing_baby_swallow, 61

- counter_attack, 64
- dark_cloud_cover, 67
- directional_movement_index, 70
- doji, 73
- doji_star, 76
- dominant_cycle_period, 79
- dominant_cycle_phase, 81
- double_exponential_moving_average, 83
- dragonfly_doji, 85
- engulfing, 89
- evening_doji_star, 92
- evening_star, 95
- exponential_moving_average, 98
- extended_moving_average_convergence_divergence, 100
- extended_parabolic_stop_and_reverse, 103
- fast_stochastic, 106
- fixed_moving_average_convergence_divergence, 108
- gaps_side_white, 111
- gravestone_doji, 115
- hammer, 118
- hanging_man, 121
- harami, 124
- harami_cross, 127
- high_wave, 130
- hikakke, 133
- hikakke_mod, 137
- homing_pigeon, 140
- hull_moving_average, 143
- in_neck, 153
- intraday_movement_index, 147
- inverted_hammer, 149
- kaufman_adaptive_moving_average, 156
- kicking, 158
- kicking_baby_length, 161
- ladder_bottom, 165
- long_legged_doji, 168
- long_line, 171
- lookback, 174
- marubozu, 175
- mat_hold, 181
- matching_low, 178
- median_price, 184
- mesa_adaptive_moving_average, 186
- midpoint_period, 188
- midpoint_price, 191
- minus_directional_indicator, 193
- minus_directional_movement, 195
- momentum, 197
- money_flow_index, 199
- morning_doji_star, 202
- morning_star, 205
- moving_average_convergence_divergence, 208
- negative_volume_index, 210
- normalized_average_true_range, 213
- on_balance_volume, 215
- on_neck, 218
- parabolic_stop_and_reverse, 221
- percentage_price_oscillator, 223
- percentage_volume_oscillator, 226
- phasor_components, 228
- piercing, 230
- plus_directional_indicator, 233
- plus_directional_movement, 236
- positive_volume_index, 238
- ratio_of_change, 240
- relative_strength_index, 242
- rickshaw_man, 244
- rise_fall_3_methods, 248
- rolling_beta, 251
- rolling_correlation, 252
- rolling_maximum, 253
- rolling_minimum, 255
- rolling_standard_deviation, 256
- rolling_sum, 258
- rolling_variance, 259
- separating_lines, 260
- shooting_star, 265
- short_line, 268
- simple_moving_average, 271
- sine_wave, 274
- spinning_top, 276
- stalled_pattern, 280
- stick_sandwich, 283
- stochastic, 286
- stochastic_relative_strength_index, 288
- t3_exponential_moving_average, 291
- takuri, 294
- tasuki_gap, 297
- three_black_crows, 300

- three_identical_crows, 303
- three_inside, 306
- three_line_strike, 309
- three_outside, 312
- three_stars_in_the_south, 315
- three_white_soldiers, 318
- thrusting, 322
- trading_volume, 325
- trend_cycle_mode, 329
- trendline, 327
- triangular_moving_average, 331
- triple_exponential_average, 334
- triple_exponential_moving_average, 336
- tristar, 339
- true_range, 342
- two_crows, 344
- typical_price, 347
- ultimate_oscillator, 349
- unique_3_river, 351
- unsmoothed_chande_momentum_oscillator, 354
- upside_gap_2_crows, 356
- variable_moving_average_period, 360
- volume_weighted_moving_average, 362
- weighted_close_price, 365
- weighted_moving_average, 367
- williams_oscillator, 369
- xside_gap_3_methods, 371
- * **technical analysis**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 11
 - advance_block, 13
 - aroon, 16
 - aroon_oscillator, 18
 - average_deviation, 21
 - average_directional_movement_index, 23
 - average_directional_movement_index_rating, 25
 - average_price, 27
 - average_true_range, 29
 - balance_of_power, 31
 - belt_hold, 33
 - bollinger_bands, 36
 - break_away, 39
 - chaikin_accumulation_distribution_line, 43
 - chaikin_accumulation_distribution_oscillator, 45
 - chaikin_money_flow, 47
 - chande_momentum_oscillator, 49
 - closing_marubozu, 56
 - commodity_channel_index, 59
 - concealing_baby_swallow, 61
 - counter_attack, 64
 - dark_cloud_cover, 67
 - directional_movement_index, 70
 - doji, 73
 - doji_star, 76
 - dominant_cycle_period, 79
 - dominant_cycle_phase, 81
 - double_exponential_moving_average, 83
 - dragonfly_doji, 85
 - engulfing, 89
 - evening_doji_star, 92
 - evening_star, 95
 - exponential_moving_average, 98
 - extended_moving_average_convergence_divergence, 100
 - extended_parabolic_stop_and_reverse, 103
 - fast_stochastic, 106
 - fixed_moving_average_convergence_divergence, 108
 - gaps_side_white, 111
 - gravestone_doji, 115
 - hammer, 118
 - hanging_man, 121
 - harami, 124
 - harami_cross, 127
 - high_wave, 130
 - hikakke, 133
 - hikakke_mod, 137
 - homing_pigeon, 140
 - hull_moving_average, 143
 - in_neck, 153
 - intraday_movement_index, 147
 - inverted_hammer, 149
 - kaufman_adaptive_moving_average, 156
 - kicking, 158

- kicking_baby_length, 161
- ladder_bottom, 165
- long_legged_doji, 168
- long_line, 171
- lookback, 174
- marubozu, 175
- mat_hold, 181
- matching_low, 178
- median_price, 184
- mesa_adaptive_moving_average, 186
- midpoint_period, 188
- midpoint_price, 191
- minus_directional_indicator, 193
- minus_directional_movement, 195
- momentum, 197
- money_flow_index, 199
- morning_doji_star, 202
- morning_star, 205
- moving_average_convergence_divergence, 208
- negative_volume_index, 210
- normalized_average_true_range, 213
- on_balance_volume, 215
- on_neck, 218
- parabolic_stop_and_reverse, 221
- percentage_price_oscillator, 223
- percentage_volume_oscillator, 226
- phasor_components, 228
- piercing, 230
- plus_directional_indicator, 233
- plus_directional_movement, 236
- positive_volume_index, 238
- ratio_of_change, 240
- relative_strength_index, 242
- rickshaw_man, 244
- rise_fall_3_methods, 248
- rolling_beta, 251
- rolling_correlation, 252
- rolling_maximum, 253
- rolling_minimum, 255
- rolling_standard_deviation, 256
- rolling_sum, 258
- rolling_variance, 259
- separating_lines, 260
- shooting_star, 265
- short_line, 268
- simple_moving_average, 271
- sine_wave, 274
- spinning_top, 276
- stalled_pattern, 280
- stick_sandwich, 283
- stochastic, 286
- stochastic_relative_strength_index, 288
- t3_exponential_moving_average, 291
- takuri, 294
- tasuki_gap, 297
- three_black_crows, 300
- three_identical_crows, 303
- three_inside, 306
- three_line_strike, 309
- three_outside, 312
- three_stars_in_the_south, 315
- three_white_soldiers, 318
- thrusting, 322
- trading_volume, 325
- trend_cycle_mode, 329
- trendline, 327
- triangular_moving_average, 331
- triple_exponential_average, 334
- triple_exponential_moving_average, 336
- tristar, 339
- true_range, 342
- two_crows, 344
- typical_price, 347
- ultimate_oscillator, 349
- unique_3_river, 351
- unsmoothed_chande_momentum_oscillator, 354
- upside_gap_2_crows, 356
- variable_moving_average_period, 360
- volume_weighted_moving_average, 362
- weighted_close_price, 365
- weighted_moving_average, 367
- williams_oscillator, 369
- xside_gap_3_methods, 371
- * trading**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 11
 - advance_block, 13
 - aroon, 16
 - aroon_oscillator, 18

- average_deviation, 21
- average_directional_movement_index, 23
- average_directional_movement_index_rating, 25
- average_price, 27
- average_true_range, 29
- balance_of_power, 31
- belt_hold, 33
- bollinger_bands, 36
- break_away, 39
- chaikin_accumulation_distribution_line, 43
- chaikin_accumulation_distribution_oscillator, 45
- chaikin_money_flow, 47
- chande_momentum_oscillator, 49
- closing_marubozu, 56
- commodity_channel_index, 59
- concealing_baby_swallow, 61
- counter_attack, 64
- dark_cloud_cover, 67
- directional_movement_index, 70
- doji, 73
- doji_star, 76
- dominant_cycle_period, 79
- dominant_cycle_phase, 81
- double_exponential_moving_average, 83
- dragonfly_doji, 85
- engulfing, 89
- evening_doji_star, 92
- evening_star, 95
- exponential_moving_average, 98
- extended_moving_average_convergence_divergence, 100
- extended_parabolic_stop_and_reverse, 103
- fast_stochastic, 106
- fixed_moving_average_convergence_divergence, 108
- gaps_side_white, 111
- gravestone_doji, 115
- hammer, 118
- hanging_man, 121
- harami, 124
- harami_cross, 127
- high_wave, 130
- hikakke, 133
- hikakke_mod, 137
- homing_pigeon, 140
- hull_moving_average, 143
- in_neck, 153
- intraday_movement_index, 147
- inverted_hammer, 149
- kaufman_adaptive_moving_average, 156
- kicking, 158
- kicking_baby_length, 161
- ladder_bottom, 165
- long_legged_doji, 168
- long_line, 171
- lookback, 174
- marubozu, 175
- mat_hold, 181
- matching_low, 178
- median_price, 184
- mesa_adaptive_moving_average, 186
- midpoint_period, 188
- midpoint_price, 191
- minus_directional_indicator, 193
- minus_directional_movement, 195
- momentum, 197
- money_flow_index, 199
- morning_doji_star, 202
- morning_star, 205
- moving_average_convergence_divergence, 208
- negative_volume_index, 210
- normalized_average_true_range, 213
- on_balance_volume, 215
- on_neck, 218
- parabolic_stop_and_reverse, 221
- percentage_price_oscillator, 223
- percentage_volume_oscillator, 226
- phasor_components, 228
- piercing, 230
- plus_directional_indicator, 233
- plus_directional_movement, 236
- positive_volume_index, 238
- ratio_of_change, 240
- relative_strength_index, 242
- rickshaw_man, 244
- rise_fall_3_methods, 248
- rolling_beta, 251
- rolling_correlation, 252

- rolling_maximum, 253
- rolling_minimum, 255
- rolling_standard_deviation, 256
- rolling_sum, 258
- rolling_variance, 259
- separating_lines, 260
- shooting_star, 265
- short_line, 268
- simple_moving_average, 271
- sine_wave, 274
- spinning_top, 276
- stalled_pattern, 280
- stick_sandwich, 283
- stochastic, 286
- stochastic_relative_strength_index, 288
- t3_exponential_moving_average, 291
- takuri, 294
- tasuki_gap, 297
- three_black_crows, 300
- three_identical_crows, 303
- three_inside, 306
- three_line_strike, 309
- three_outside, 312
- three_stars_in_the_south, 315
- three_white_soldiers, 318
- thrusting, 322
- trading_volume, 325
- trend_cycle_mode, 329
- trendline, 327
- triangular_moving_average, 331
- triple_exponential_average, 334
- triple_exponential_moving_average, 336
- tristar, 339
- true_range, 342
- two_crows, 344
- typical_price, 347
- ultimate_oscillator, 349
- unique_3_river, 351
- unsmoothed_chande_momentum_oscillator, 354
- upside_gap_2_crows, 356
- variable_moving_average_period, 360
- volume_weighted_moving_average, 362
- weighted_close_price, 365
- weighted_moving_average, 367
- williams_oscillator, 369
- xside_gap_3_methods, 371
- abandoned_baby, 5
- abandoned_baby(), 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- abandonedBaby (abandoned_baby), 5
- absolute_price_oscillator, 8
- absolute_price_oscillator(), 17, 20, 24, 27, 32, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- absolutePriceOscillator (absolute_price_oscillator), 8
- ACCBANDS (acceleration_bands), 11
- acceleration_bands, 11
- acceleration_bands(), 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- accelerationBands (acceleration_bands), 11
- AD (chaikin_accumulation_distribution_line), 43
- ADOSC (chaikin_accumulation_distribution_oscillator), 45
- advance_block, 13
- advance_block(), 7, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- advanceBlock (advance_block), 13
- ADX (average_directional_movement_index),

- 23
- ADX (average_directional_movement_index_rating), 23
- ADXR (average_directional_movement_index_rating), 23
- APO (absolute_price_oscillator), 8
- AROON (aroon), 16
- aroon(), 10, 20, 24, 27, 32, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- aroon_oscillator, 18
- aroon_oscillator(), 10, 17, 24, 27, 32, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- AROONOSC (aroon_oscillator), 18
- aroonOscillator (aroon_oscillator), 18
- ATOM, 20
- ATR (average_true_range), 29
- average_deviation, 21
- average_deviation(), 29, 185, 348, 366
- average_directional_movement_index, 23
- average_directional_movement_index(), 10, 17, 20, 27, 32, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- average_directional_movement_index_rating, 23
- average_directional_movement_index_rating(), 10, 17, 20, 24, 32, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- average_price, 27
- average_price(), 22, 185, 348, 366
- average_true_range, 29
- average_true_range(), 214, 343
- averageDeviation (average_deviation), 21
- averageDirectionalMovementIndex (average_directional_movement_index), 23
- averageDirectionalMovementIndexRating (average_directional_movement_index_rating), 23
- averagePrice (average_price), 27
- averageTrueRange (average_true_range), 29
- AVGDEV (average_deviation), 21
- AVGPRICE (average_price), 27
- balance_of_power, 31
- balance_of_power(), 10, 17, 20, 24, 27, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- balanceOfPower (balance_of_power), 31
- BBANDS (bollinger_bands), 36
- belt_hold, 33
- belt_hold(), 7, 15, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- beltHold (belt_hold), 33
- BETA (rolling_beta), 251
- bollinger_bands, 36
- bollinger_bands(), 12, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- bollingerBands (bollinger_bands), 36
- BOP (balance_of_power), 31
- break_away, 39
- break_away(), 7, 15, 36, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- breakAway (break_away), 39
- BTC, 42
- call, 174
- CCI (commodity_channel_index), 59
- CDL2CROWS (two_crows), 344
- CDL3BLACKCROWS (three_black_crows), 300
- CDL3INSIDE (three_inside), 306
- CDL3LINESTRIKE (three_line_strike), 309
- CDL3OUTSIDE (three_outside), 312

- CDL3STARSINSOUTH
 - (three_stars_in_the_south), [315](#)
- CDL3WHITESOLDIERS
 - (three_white_soldiers), [318](#)
- CDLABANDONEDBABY (abandoned_baby), [5](#)
- CDLADVANCEBLOCK (advance_block), [13](#)
- CDLBELTHOLD (belt_hold), [33](#)
- CDLBREAKAWAY (break_away), [39](#)
- CDLCLOSINGMARUBOZU (closing_marubozu), [56](#)
- CDLCONCEALBABYSWALL
 - (concealing_baby_swallow), [61](#)
- CDLCOUNTERATTACK (counter_attack), [64](#)
- CDLDARKCLOUDCOVER (dark_cloud_cover), [67](#)
- CDLDOJI (doji), [73](#)
- CDLDOJISTAR (doji_star), [76](#)
- CDLDRAGONFLYDOJI (dragonfly_doji), [85](#)
- CDLENGULFING (engulfing), [89](#)
- CDLEVENINGDOJISTAR (evening_doji_star), [92](#)
- CDLEVENINGSTAR (evening_star), [95](#)
- CDLGAPSIDESIDEWHITE (gaps_side_white), [111](#)
- CDLGRAVESTONEDOJI (gravestone_doji), [115](#)
- CDLHAMMER (hammer), [118](#)
- CDLHANGINGMAN (hanging_man), [121](#)
- CDLHARAMI (harami), [124](#)
- CDLHARAMICROSS (harami_cross), [127](#)
- CDLHIGHWAVE (high_wave), [130](#)
- CDLHIKKAKE (hikakke), [133](#)
- CDLHIKKAKEMOD (hikakke_mod), [137](#)
- CDLHOMINGPIGEON (homing_pigeon), [140](#)
- CDLIDENTICAL3CROWS
 - (three_identical_crows), [303](#)
- CDLINNECK (in_neck), [153](#)
- CDLINVERTEDHAMMER (inverted_hammer), [149](#)
- CDLKICKING (kicking), [158](#)
- CDLKICKINGBYLENGTH
 - (kicking_baby_length), [161](#)
- CDLLADDERBOTTOM (ladder_bottom), [165](#)
- CDLLONGLEGGEDOJI (long_legged_doji), [168](#)
- CDLLONGLINE (long_line), [171](#)
- CDLMARUBOZU (marubozu), [175](#)
- CDLMATCHINGLOW (matching_low), [178](#)
- CDLMATHOLD (mat_hold), [181](#)
- CDLMORNINGDOJISTAR (morning_doji_star), [202](#)
- CDLMORNINGSTAR (morning_star), [205](#)
- CDLONNECK (on_neck), [218](#)
- CDLPIERCING (piercing), [230](#)
- CDLRICKSHAWMAN (rickshaw_man), [244](#)
- CDLRISEFALL3METHODS
 - (rise_fall_3_methods), [248](#)
- CDLSEPARATINGLINES (separating_lines), [260](#)
- CDLSHOOTINGSTAR (shooting_star), [265](#)
- CDLSHORTLINE (short_line), [268](#)
- CDLSPINNINGTOP (spinning_top), [276](#)
- CDLSTALLEDPATTERN (stalled_pattern), [280](#)
- CDLSTICKSANDWICH (stick_sandwich), [283](#)
- CDLTAKURI (takuri), [294](#)
- CDLTASUKIGAP (tasuki_gap), [297](#)
- CDLTHRUSTING (thrusting), [322](#)
- CDLTRISTAR (tristar), [339](#)
- CDLUNIQUE3RIVER (unique_3_river), [351](#)
- CDLUPSIDEGAP2CROWS
 - (upside_gap_2_crows), [356](#)
- CDLXSIDEGAP3METHODS
 - (xside_gap_3_methods), [371](#)
- chaikin_accumulation_distribution_line, [43](#)
- chaikin_accumulation_distribution_line(), [46, 48, 212, 217, 227, 239, 326](#)
- chaikin_accumulation_distribution_oscillator, [45](#)
- chaikin_accumulation_distribution_oscillator(), [44, 48, 212, 217, 227, 239, 326](#)
- chaikin_money_flow, [47](#)
- chaikin_money_flow(), [44, 46, 212, 217, 227, 239, 326](#)
- chaikinAccumulationDistributionLine
 - (chaikin_accumulation_distribution_line), [43](#)
- chaikinAccumulationDistributionOscillator
 - (chaikin_accumulation_distribution_oscillator), [45](#)
- chaikinMoneyFlow (chaikin_money_flow), [47](#)
- chande_momentum_oscillator, [49](#)
- chande_momentum_oscillator(), [10, 17, 20, 24, 27, 32, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370](#)
- chandeMomentumOscillator

- (chande_momentum_oscillator),
49
- character, 52, 264
- chart, 51
- chart(), 55, 145–147, 263, 264
- chart_themes, 53, 54, 147, 264
- class, 5, 7, 8, 10–13, 15–19, 21–26, 28–33,
35, 37–39, 41, 43–50, 56, 58–61, 63,
64, 66, 67, 69, 71–73, 75, 76, 78–84,
86, 87, 89, 91, 92, 94, 95, 97–100,
102, 103, 105–108, 110, 111, 113,
115, 117, 118, 120, 121, 123, 124,
126, 127, 129, 130, 132, 134, 135,
137, 139, 140, 142–144, 147, 148,
150, 151, 153, 155–158, 160, 162,
163, 165, 167, 168, 170, 171, 173,
175, 177, 178, 180, 181, 183–188,
190–200, 202, 204, 205, 207–209,
211–214, 216–218, 220–223,
225–230, 232, 234–243, 245, 246,
248, 250–252, 254–256, 258–260,
262, 265, 267, 268, 270, 271,
273–276, 278, 280, 282, 283,
285–287, 289–291, 293, 294, 296,
297, 299, 300, 302, 303, 305, 306,
308, 309, 311, 312, 314, 315, 317,
319, 320, 322, 324–331, 333–336,
338–340, 342–344, 346–351,
353–355, 357, 358, 360–362,
364–371, 373
- closing_marubozu, 56
- closing_marubozu(), 7, 15, 36, 41, 63, 66,
70, 75, 78, 88, 91, 94, 97, 113, 117,
120, 123, 126, 129, 133, 136, 139,
142, 152, 155, 161, 164, 167, 170,
173, 177, 180, 183, 204, 207, 220,
233, 247, 250, 262, 267, 270, 278,
282, 285, 296, 299, 302, 305, 308,
311, 314, 318, 321, 324, 341, 346,
353, 359, 373
- closingMarubozu(closing_marubozu), 56
- CMF (chaikin_money_flow), 47
- CMO (chande_momentum_oscillator), 49
- CMOU
(unsmoothed_chande_momentum_oscillator),
354
- commodity_channel_index, 59
- commodity_channel_index(), 10, 17, 20, 24,
27, 32, 51, 72, 102, 108, 110, 149,
194, 196, 199, 201, 210, 225, 235,
237, 241, 244, 288, 290, 335, 350,
356, 370
- commodityChannelIndex
(commodity_channel_index), 59
- concealing_baby_swallow, 61
- concealing_baby_swallow(), 7, 15, 36, 41,
58, 66, 70, 75, 78, 88, 91, 94, 97,
113, 117, 120, 123, 126, 129, 133,
136, 139, 142, 152, 155, 161, 164,
167, 170, 173, 177, 180, 183, 204,
207, 220, 233, 247, 250, 262, 267,
270, 278, 282, 285, 296, 299, 302,
305, 308, 311, 314, 318, 321, 324,
341, 346, 353, 359, 373
- concealingBabySwallow
(concealing_baby_swallow), 61
- CORREL (rolling_correlation), 252
- counter_attack, 64
- counter_attack(), 7, 15, 36, 41, 58, 63, 70,
75, 78, 88, 91, 94, 97, 113, 117, 120,
123, 126, 129, 133, 136, 139, 142,
152, 155, 161, 164, 167, 170, 173,
177, 180, 183, 204, 207, 220, 233,
247, 250, 262, 267, 270, 278, 282,
285, 296, 299, 302, 305, 308, 311,
314, 318, 321, 324, 341, 346, 353,
359, 373
- counterAttack(counter_attack), 64
- dark_cloud_cover, 67
- dark_cloud_cover(), 7, 15, 36, 41, 58, 63,
66, 75, 78, 88, 91, 94, 97, 113, 117,
120, 123, 126, 129, 133, 136, 139,
142, 152, 155, 161, 164, 167, 170,
173, 177, 180, 183, 204, 207, 220,
233, 247, 250, 262, 267, 270, 278,
282, 285, 296, 299, 302, 305, 308,
311, 314, 318, 321, 324, 341, 346,
353, 359, 373
- darkCloudCover(dark_cloud_cover), 67
- data.frame, 5, 6, 8, 9, 11, 13, 14, 16–19,
21–26, 28–34, 37, 39, 40, 42–50, 52,
56, 57, 59–62, 64, 65, 67, 68, 71, 73,
76, 77, 79–84, 86, 89, 92, 93, 95, 96,
98–101, 103, 104, 106–109, 111,
112, 115, 116, 118, 119, 121, 122,
124, 125, 127, 128, 130, 131, 134,

- 137, 140, 141, 143, 144, 147, 148,
150, 153, 156–159, 162, 165, 168,
169, 171, 172, 175, 176, 178, 179,
181, 182, 184–189, 191, 193–200,
202, 205, 206, 208, 209, 211, 213,
216, 218, 221–224, 226–231, 234,
236, 238–243, 245, 248, 260, 261,
265, 266, 268, 269, 271, 272,
274–277, 280, 283, 284, 286, 287,
289, 291, 292, 294, 297, 298, 300,
301, 303, 304, 306, 307, 309, 310,
312, 313, 315, 316, 319, 322,
325–332, 334–337, 339, 342, 344,
347–349, 351, 352, 354, 355, 357,
360–363, 365–367, 369–372
- DEMA
(double_exponential_moving_average),
83
- directional_movement_index, 70
- directional_movement_index(), 10, 17, 20,
24, 27, 32, 51, 60, 102, 108, 110,
149, 194, 196, 199, 201, 210, 225,
235, 237, 241, 244, 288, 290, 335,
350, 356, 370
- directionalMovementIndex
(directional_movement_index),
70
- doji, 73
- doji(), 7, 15, 36, 41, 58, 63, 66, 70, 78, 88,
91, 94, 97, 113, 117, 120, 123, 126,
129, 133, 136, 139, 142, 152, 155,
161, 164, 167, 170, 173, 177, 180,
183, 204, 207, 220, 233, 247, 250,
262, 267, 270, 278, 282, 285, 296,
299, 302, 305, 308, 311, 314, 318,
321, 324, 341, 346, 353, 359, 373
- doji_star, 76
- doji_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75,
88, 91, 94, 97, 113, 117, 120, 123,
126, 129, 133, 136, 139, 142, 152,
155, 161, 164, 167, 170, 173, 177,
180, 183, 204, 207, 220, 233, 247,
250, 262, 267, 270, 278, 282, 285,
296, 299, 302, 305, 308, 311, 314,
318, 321, 324, 341, 346, 353, 359,
373
- dojiStar(doji_star), 76
- dominant_cycle_period, 79
- dominant_cycle_period(), 82, 230, 275,
331
- dominant_cycle_phase, 81
- dominant_cycle_phase(), 80, 230, 275, 331
- dominantCyclePeriod
(dominant_cycle_period), 79
- dominantCyclePhase
(dominant_cycle_phase), 81
- double, 6, 8–10, 12, 14, 17, 19, 21, 22, 24, 26,
29, 30, 32, 34, 37, 38, 40, 44, 46,
48–50, 57, 60, 62, 65, 68, 72, 74, 77,
79–85, 87, 90, 93, 96, 98, 99, 101,
102, 104, 105, 107, 109, 110, 112,
116, 119, 122, 125, 128, 131, 135,
138, 141, 143, 144, 148, 151, 154,
156, 157, 160, 163, 166, 169, 172,
176, 179, 182, 185–190, 192, 194,
196–198, 201–203, 206, 208, 209,
212, 214, 217, 219, 222–229, 231,
235, 237, 239–243, 246, 249,
251–261, 266, 269, 271–275, 277,
281, 284, 287, 289–293, 295, 298,
301, 304, 307, 310, 313, 316, 320,
323, 326–338, 340, 343, 345, 348,
350, 352, 354, 355, 358, 360, 361,
364, 366–368, 370, 372
- double_exponential_moving_average, 83,
84
- double_exponential_moving_average(),
12, 38, 100, 105, 144, 158, 188, 190,
192, 222, 273, 293, 328, 333, 338,
361, 364, 368
- doubleExponentialMovingAverage
(double_exponential_moving_average),
83
- dragonfly_doji, 85
- dragonfly_doji(), 7, 15, 36, 41, 58, 63, 66,
70, 75, 78, 91, 94, 97, 113, 117, 120,
123, 126, 129, 133, 136, 139, 142,
152, 155, 161, 164, 167, 170, 173,
177, 180, 183, 204, 207, 220, 233,
247, 250, 262, 267, 270, 278, 282,
285, 296, 299, 302, 305, 308, 311,
314, 318, 321, 324, 341, 346, 353,
359, 373
- dragonflyDoji(dragonfly_doji), 85
- DX(directional_movement_index), 70
- EMA, 9, 224, 227

- EMA (exponential_moving_average), 98
- engulfing, 89
- engulfing(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- evening_doji_star, 92
- evening_doji_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- evening_star, 95
- evening_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- eveningDojiStar (evening_doji_star), 92
- eveningStar (evening_star), 95
- exponential_moving_average, 98, 99
- exponential_moving_average(), 12, 38, 85, 105, 144, 158, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- exponentialMovingAverage (exponential_moving_average), 98
- extended_moving_average_convergence_divergence, 100
- extended_moving_average_convergence_divergence(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- extendedMovingAverageConvergenceDivergence (extended_moving_average_convergence_divergence), 100
- extendedParabolicStopAndReverse (extended_parabolic_stop_and_reverse), 103
- extended_parabolic_stop_and_reverse, 103
- extended_parabolic_stop_and_reverse(), 12, 38, 85, 100, 144, 158, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- FALSE, 6, 9, 12, 14, 17, 19, 22, 24, 26, 28, 30, 32, 34, 38, 40, 44, 46, 48, 50, 57, 60, 62, 65, 68, 71, 74, 77, 80, 82, 84, 86, 90, 93, 96, 99, 102, 105, 107, 110, 112, 116, 119, 122, 125, 128, 131, 134, 138, 141, 144, 148, 150, 154, 157, 159, 162, 166, 169, 172, 176, 179, 182, 185, 187, 189, 192, 194, 196, 198, 200, 203, 206, 209, 211, 214, 216, 219, 222, 224, 227, 229, 231, 234, 237, 239, 241, 243, 245, 249, 251, 253–255, 257–259, 261, 266, 269, 272, 275, 277, 281, 284, 287, 290, 292, 295, 298, 301, 304, 307, 310, 313, 316, 319, 323, 326, 328, 330, 332, 335, 337, 339, 343, 345, 348, 350, 352, 355, 357, 361, 363, 366, 368, 370, 372
- fast_stochastic, 106
- fast_stochastic(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- fastStochastic (fast_stochastic), 106
- fixed_moving_average_convergence_divergence, 108
- fixed_moving_average_convergence_divergence(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- fixedMovingAverageConvergenceDivergence (fixed_moving_average_convergence_divergence), 108
- formula, 6, 9, 11, 14, 17, 19, 22, 24, 26, 28, 30, 32, 34, 38, 40, 44, 46, 48, 50, 57,

- 60, 62, 65, 68, 71, 73, 77, 80, 82, 84, 86, 89, 93, 96, 99, 101, 104, 107, 109, 112, 116, 119, 122, 125, 128, 131, 134, 137, 141, 144, 148, 150, 153, 157, 159, 162, 165, 169, 172, 176, 179, 182, 185, 187, 189, 191, 194, 196, 198, 200, 202, 206, 209, 211, 213, 216, 218, 222, 224, 227, 229, 231, 234, 236, 239, 241, 243, 245, 248, 261, 266, 269, 272, 275, 277, 280, 284, 287, 290, 292, 294, 298, 301, 304, 307, 310, 313, 316, 319, 322, 326, 328, 330, 332, 335, 337, 339, 342, 344, 348, 350, 352, 355, 357, 361, 363, 366, 368, 370, 372
- function, 174
- gaps_side_white, 111
- gaps_side_white(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- gapsSideWhite (gaps_side_white), 111
- GOOGL, 114
- gravestone_doji, 115
- gravestone_doji(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- gravestoneDoji (gravestone_doji), 115
- hammer, 118
- hammer(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- hanging_man, 121
- hanging_man(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- hangingMan (hanging_man), 121
- harami, 124
- harami(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- harami_cross, 127
- harami_cross(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- haramiCross (harami_cross), 127
- high_wave, 130
- high_wave(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- highWave (high_wave), 130
- hikakke, 133
- hikakke(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 139, 142, 152,

- 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373*
- `hikakke_mod`, 137
- `hikakke_mod()`, *7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373*
- `hikakkeMod (hikakke_mod)`, 137
- `HMA (hull_moving_average)`, 143
- `homing_pigeon`, 140
- `homing_pigeon()`, *7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373*
- `homingPigeon (homing_pigeon)`, 140
- `HT_DCPERIOD (dominant_cycle_period)`, 79
- `HT_DCPHASE (dominant_cycle_phase)`, 81
- `HT_PHASOR (phasor_components)`, 228
- `HT_SINE (sine_wave)`, 274
- `HT_TRENDLINE (trendline)`, 327
- `HT_TRENDMODE (trend_cycle_mode)`, 329
- `hull_moving_average`, 143, 144
- `hull_moving_average()`, *12, 38, 85, 100, 105, 158, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368*
- `hullMovingAverage (hull_moving_average)`, 143
- `IMI (intraday_movement_index)`, 147
- `in_neck`, 153
- `in_neck()`, *7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373*
- `indicator`, 145
- `indicator()`, *51–53, 55, 263, 264*
- `inNeck (in_neck)`, 153
- `integer`, *6, 7, 9, 12, 14, 15, 17, 19, 22, 24, 26, 30, 34, 35, 38, 40, 41, 46, 48, 50, 57, 58, 60, 62, 63, 65, 66, 68, 69, 71, 74, 75, 77, 78, 84, 87, 90, 91, 93, 94, 96, 97, 99, 102, 107, 109, 112, 113, 116, 117, 119, 120, 122, 123, 125, 126, 128, 129, 131, 132, 135, 138, 139, 141, 142, 144, 148, 151, 154, 155, 157, 159, 160, 163, 166, 167, 169, 170, 172–174, 176, 177, 179, 180, 182, 183, 187, 189, 191, 194, 196, 198, 200, 203, 204, 206, 207, 209, 213, 219, 220, 224, 227, 231, 232, 234, 237, 241, 243, 246, 249–251, 253–255, 257–259, 261, 262, 266, 267, 269, 270, 272, 277, 278, 281, 282, 284, 285, 287, 290, 292, 295, 296, 298, 299, 301, 302, 304, 305, 307, 308, 310, 311, 313, 314, 316, 317, 320, 323, 324, 330, 332, 335, 337, 340, 345, 346, 350, 352, 353, 355, 358, 361, 363, 367, 370, 372, 373*
- `intraday_movement_index`, 147
- `intraday_movement_index()`, *10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370*
- `intradayMovementIndex (intraday_movement_index)`, 147
- `inverted_hammer`, 149
- `inverted_hammer()`, *7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373*

- `invertedHammer (inverted_hammer)`, 149
- `KAMA (kaufman_adaptive_moving_average)`, 156
- `kaufman_adaptive_moving_average`, 156, 157
- `kaufman_adaptive_moving_average()`, 12, 38, 85, 100, 105, 144, 188, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- `kaufmanAdaptiveMovingAverage (kaufman_adaptive_moving_average)`, 156
- `kicking`, 158
- `kicking()`, 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- `kicking_baby_length`, 161
- `kicking_baby_length()`, 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- `kickingBabyLength (kicking_baby_length)`, 161
- `ladder_bottom`, 165
- `ladder_bottom()`, 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- `ladderBottom (ladder_bottom)`, 165
- `length`, 6, 7, 9, 10, 12, 14, 15, 17, 19, 22, 24, 26, 28–30, 32, 34, 35, 38, 40, 41, 44, 46, 48, 50, 52, 57, 58, 60, 62, 63, 65, 66, 68, 69, 71, 72, 74, 75, 77, 78, 80, 82, 84, 86, 87, 90, 91, 93, 94, 96, 97, 99, 102, 105, 107, 110, 112, 113, 116, 117, 119, 120, 122, 123, 125, 126, 128, 129, 131, 132, 134, 135, 138, 139, 141, 142, 144, 148, 150, 151, 154, 155, 157, 159, 160, 162, 163, 166, 167, 169, 170, 172–174, 176, 177, 179, 180, 182, 183, 185, 187, 189–192, 194, 196, 198, 200, 202–204, 206, 207, 209, 211–214, 216, 217, 219, 220, 222, 224, 225, 227, 229, 231, 232, 234, 235, 237, 239, 241, 243, 245, 246, 249–251, 253–262, 266, 267, 269, 270, 272, 273, 275, 277, 278, 281, 282, 284, 285, 287, 290, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 307, 308, 310, 311, 313, 314, 316, 317, 319, 320, 323, 324, 326, 328, 330, 332, 333, 335, 337–340, 343, 345, 346, 348, 350, 352, 353, 355, 357, 358, 361, 363, 364, 366–368, 370, 372, 373
- `lines()`, 53
- `list`, 326
- `logical`, 6, 9, 12, 14, 17, 19, 22, 24, 26, 28, 30, 32, 34, 38, 40, 44, 46, 48, 50, 57, 60, 62, 65, 68, 71, 74, 77, 80, 82, 84, 86, 90, 93, 96, 99, 102, 105, 107, 110, 112, 116, 119, 122, 125, 128, 131, 134, 138, 141, 144, 148, 150, 154, 157, 159, 162, 166, 169, 172, 176, 179, 182, 185, 187, 189, 192, 194, 196, 198, 200, 203, 206, 209, 211, 214, 216, 219, 222, 224, 227, 229, 231, 234, 237, 239, 241, 243, 245, 249, 251, 253–255, 257–259, 261, 266, 269, 272, 275, 277, 281, 284, 287, 290, 292, 295, 298, 301, 304, 307, 310, 313, 316, 319, 323, 326, 328, 330, 332, 335, 337, 339, 343, 345, 348, 350, 352, 355, 357, 361, 363, 366, 368, 370, 372
- `long_legged_doji`, 168
- `long_legged_doji()`, 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113,

- [117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- `long_line`, [171](#)
- `long_line()`, [7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- `longLeggedDoji` (`long_legged_doji`), [168](#)
- `longLine` (`long_line`), [171](#)
- `lookback`, [174](#)
- MACD
 - (`moving_average_convergence_divergence`), [208](#)
- MACDEXT
 - (`extended_moving_average_convergence_divergence`), [100](#)
- MACDFIX
 - (`fixed_moving_average_convergence_divergence`), [108](#)
- MAMA (`mesa_adaptive_moving_average`), [186](#)
- `marubozu`, [175](#)
- `marubozu()`, [7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- `mat_hold`, [181](#)
- `mat_hold()`, [7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- `matching_low`, [178](#)
- `matching_low()`, [7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 183, 204, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- `matchingLow` (`matching_low`), [178](#)
- `matHold` (`mat_hold`), [181](#)
- `matrix`, [5, 8, 11, 13, 16, 18, 21, 23, 25, 28, 29, 31, 33, 37, 39, 43, 45, 47, 49, 56, 59, 61, 64, 67, 71, 73, 76, 79, 81, 83, 86, 89, 92, 95, 98, 100, 101, 103, 106, 108, 109, 111, 115, 118, 121, 124, 127, 130, 134, 137, 140, 143, 147, 150, 153, 156, 158, 162, 165, 168, 171, 175, 178, 181, 184, 186, 188, 191, 193, 195, 197, 199, 202, 205, 208, 211, 213, 215, 216, 218, 221, 223, 226, 228, 230, 234, 236, 238, 240, 242, 245, 248, 260, 265, 268, 271, 274, 276, 279, 280, 283, 286, 289, 291, 294, 297, 300, 303, 306, 309, 312, 315, 319, 322, 325, 327, 329, 331, 334, 336, 339, 342, 344, 347, 349, 351, 354, 357, 360, 362, 365, 367, 369, 371](#)
- MAVP (`variable_moving_average_period`), [360](#)
- MAX (`rolling_maximum`), [253](#)
- `median_price`, [184](#)
- `median_price()`, [22, 29, 348, 366](#)
- `medianPrice` (`median_price`), [184](#)
- MEDPRICE (`median_price`), [184](#)
- `mesa_adaptive_moving_average`, [186, 187](#)
- `mesa_adaptive_moving_average()`, [12, 38, 85, 100, 105, 144, 158, 190, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368](#)
- `mesaAdaptiveMovingAverage` (`mesa_adaptive_moving_average`), [186](#)
- MFI (`money_flow_index`), [199](#)

- MIDPOINT (midpoint_period), 188
- midpoint_period, 188
- midpoint_period(), 12, 38, 85, 100, 105, 144, 158, 188, 192, 222, 273, 293, 328, 333, 338, 361, 364, 368
- midpoint_price, 191
- midpoint_price(), 12, 38, 85, 100, 105, 144, 158, 188, 190, 222, 273, 293, 328, 333, 338, 361, 364, 368
- midpointPeriod (midpoint_period), 188
- midpointPrice (midpoint_price), 191
- MIDPRICE (midpoint_price), 191
- MIN (rolling_minimum), 255
- MINUS_DI (minus_directional_indicator), 193
- minus_directional_indicator, 193
- minus_directional_indicator(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- minus_directional_movement, 195
- minus_directional_movement(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- MINUS_DM (minus_directional_movement), 195
- minusDirectionalIndicator (minus_directional_indicator), 193
- minusDirectionalMovement (minus_directional_movement), 195
- model.frame, 6, 9–12, 14, 17, 19, 22, 24, 26, 28, 30, 32, 34, 38, 40, 44, 46, 48, 50, 57, 60, 62, 65, 68, 71–74, 77, 80, 82, 84, 86, 89, 90, 93, 96, 99, 101, 102, 104, 105, 107, 109, 110, 112, 116, 119, 122, 125, 128, 131, 134, 137, 138, 141, 144, 148, 150, 153, 154, 157, 159, 162, 165, 166, 169, 172, 176, 179, 182, 185, 187, 189, 191, 192, 194, 196, 198, 200, 202, 203, 206, 209, 211, 213, 214, 216, 218, 219, 222, 224, 225, 227, 229, 231, 234–237, 239, 241, 243, 245, 248, 249, 261, 266, 269, 272, 275, 277, 280, 281, 284, 287, 290, 292, 294, 295, 298, 301, 304, 307, 310, 313, 316, 319, 322, 323, 326, 328, 330, 332, 335, 337, 339, 340, 342–345, 348, 350, 352, 355, 357, 361, 363, 364, 366, 368, 370, 372
- MOM (momentum), 197
- momentum, 197
- momentum(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- money_flow_index, 199
- money_flow_index(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- moneyFlowIndex (money_flow_index), 199
- morning_doji_star, 202
- morning_doji_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 207, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- morning_star, 205
- morning_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 220, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- morningDojiStar (morning_doji_star), 202
- morningStar (morning_star), 205
- moving_average_convergence_divergence, 208
- moving_average_convergence_divergence(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370

- movingAverageConvergenceDivergence
 - (moving_average_convergence_divergence), 208
- NA, 174
- NATR (normalized_average_true_range), 213
- negative_volume_index, 210
- negative_volume_index(), 44, 46, 48, 217, 227, 239, 326
- negativeVolumeIndex
 - (negative_volume_index), 210
- normalized_average_true_range, 213
- normalized_average_true_range(), 30, 343
- normalizedAverageTrueRange
 - (normalized_average_true_range), 213
- numeric, 361, 363
- NVDA, 215
- NVI (negative_volume_index), 210
- OBV (on_balance_volume), 215
- on_balance_volume, 215
- on_balance_volume(), 44, 46, 48, 212, 227, 239, 326
- on_neck, 218
- on_neck(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 233, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- onBalanceVolume (on_balance_volume), 215
- onNeck (on_neck), 218
- options(), 52
- parabolic_stop_and_reverse, 221
- parabolic_stop_and_reverse(), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 273, 293, 328, 333, 338, 361, 364, 368
- parabolicStopAndReverse
 - (parabolic_stop_and_reverse), 221
- percentage_price_oscillator, 223
 - percentage_price_oscillator(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 235, 237, 241, 244, 288, 290, 335, 350, 356, 370
- percentage_volume_oscillator, 226
 - percentage_volume_oscillator(), 44, 46, 48, 212, 217, 239, 326
- percentagePriceOscillator
 - (percentage_price_oscillator), 223
- percentageVolumeOscillator
 - (percentage_volume_oscillator), 226
- phasor_components, 228
- phasor_components(), 80, 82, 275, 331
- phasorComponents (phasor_components), 228
- piercing, 230
- piercing(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 247, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- plot(), 53
- plotly::plot_ly(), 52
- PLUS_DI (plus_directional_indicator), 233
- plus_directional_indicator, 233
- plus_directional_indicator(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 237, 241, 244, 288, 290, 335, 350, 356, 370
- plus_directional_movement, 236
- plus_directional_movement(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 241, 244, 288, 290, 335, 350, 356, 370
- PLUS_DM (plus_directional_movement), 236
- plusDirectionalIndicator
 - (plus_directional_indicator), 233

- plusDirectionalMovement
 (plus_directional_movement), 236
- positive_volume_index, 238
- positive_volume_index(), 44, 46, 48, 212, 217, 227, 326
- positiveVolumeIndex
 (positive_volume_index), 238
- PPO (percentage_price_oscillator), 223
- PVI (positive_volume_index), 238
- PVO (percentage_volume_oscillator), 226
- ratio_of_change, 240
- ratio_of_change(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 244, 288, 290, 335, 350, 356, 370
- ratioOfChange (ratio_of_change), 240
- relative_strength_index, 242
- relative_strength_index(), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 288, 290, 335, 350, 356, 370
- relativeStrengthIndex
 (relative_strength_index), 242
- rickshaw_man, 244
- rickshaw_man(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 250, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- rickshawMan (rickshaw_man), 244
- rise_fall_3_methods, 248
- rise_fall_3_methods(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 262, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- riseFall3Methods (rise_fall_3_methods), 248
- ROCR (ratio_of_change), 240
- rolling_beta, 251
- rolling_beta(), 253, 254, 256–258, 260
- rolling_correlation, 252
- rolling_correlation(), 252, 254, 256–258, 260
- rolling_maximum, 253
- rolling_maximum(), 252, 253, 256–258, 260
- rolling_minimum, 255
- rolling_minimum(), 252–254, 257, 258, 260
- rolling_standard_deviation, 256
- rolling_standard_deviation(), 252–254, 256, 258, 260
- rolling_sum, 258
- rolling_sum(), 252–254, 256, 257, 260
- rolling_variance, 259
- rolling_variance(), 252–254, 256–258
- rollingBeta (rolling_beta), 251
- rollingCorrelation
 (rolling_correlation), 252
- rollingMaximum (rolling_maximum), 253
- rollingMinimum (rolling_minimum), 255
- rollingStandardDeviation
 (rolling_standard_deviation), 256
- rollingSum (rolling_sum), 258
- rollingVariance (rolling_variance), 259
- RSI (relative_strength_index), 242
- SAR (parabolic_stop_and_reverse), 221
- SAREXT
 (extended_parabolic_stop_and_reverse), 103
- separating_lines, 260
- separating_lines(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 267, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373
- separatingLines (separating_lines), 260
- set_theme, 263
- set_theme(), 53–55, 147
- shooting_star, 265
- shooting_star(), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117,

- [120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 270, 278, 282, 285, 296, 299, 302, 305, 308, 311, 314, 318, 321, 324, 341, 346, 353, 359, 373](#)
- [shootingStar \(shooting_star\), 265](#)
- [short_line, 268](#)
- [short_line\(\), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 129, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374](#)
- [shortLine \(short_line\), 268](#)
- [simple_moving_average, 271, 273](#)
- [simple_moving_average\(\), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 293, 328, 333, 338, 361, 364, 368](#)
- [simpleMovingAverage \(simple_moving_average\), 271](#)
- [sine_wave, 274](#)
- [sine_wave\(\), 80, 82, 230, 331](#)
- [sineWave \(sine_wave\), 274](#)
- [SMA, 38, 102, 107, 287, 290, 361](#)
- [SMA \(simple_moving_average\), 271](#)
- [spinning_top, 276](#)
- [spinning_top\(\), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 262, 267, 271, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374](#)
- [spinningTop \(spinning_top\), 276](#)
- [SPY, 279](#)
- [stalled_pattern, 280](#)
- [stalled_pattern\(\), 7, 15, 36, 41, 58, 63, 66, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374](#)
- [stalledPattern \(stalled_pattern\), 280](#)
- [STDDEV \(rolling_standard_deviation\), 256](#)
- [stick_sandwich, 283](#)
- [stick_sandwich\(\), 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374](#)
- [stickSandwich \(stick_sandwich\), 283](#)
- [STOCH \(stochastic\), 286](#)
- [stochastic, 84, 99, 144, 157, 187, 273, 286, 292, 333, 337, 364, 368](#)
- [stochastic\(\), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 290, 335, 350, 356, 370](#)
- [stochastic_relative_strength_index, 288](#)
- [stochastic_relative_strength_index\(\), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 335, 350, 356, 370](#)
- [stochasticRelativeStrengthIndex \(stochastic_relative_strength_index\), 288](#)
- [STOCHF \(fast_stochastic\), 106](#)
- [STOCHRSI \(stochastic_relative_strength_index\), 288](#)
- [SUM \(rolling_sum\), 258](#)
- [T3 \(t3_exponential_moving_average\), 291](#)
- [t3_exponential_moving_average, 291, 292](#)
- [t3_exponential_moving_average\(\), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 328, 333, 338, 361, 364, 368](#)
- [t3ExponentialMovingAverage \(t3_exponential_moving_average\), 291](#)

- `takuri`, 294
- `takuri()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `tasuki_gap`, 297
- `tasuki_gap()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `tasukiGap(tasuki_gap)`, 297
- TEMA
 - (`triple_exponential_moving_average`), 336
- `three_black_crows`, 300
- `three_black_crows()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `three_identical_crows`, 303
- `three_identical_crows()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `three_inside`, 306
- `three_inside()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `three_line_strike`, 309
- `three_line_strike()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 315, 318, 321, 324, 341, 346, 353, 359, 374
- `three_outside`, 312
- `three_outside()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 318, 321, 324, 341, 346, 353, 359, 374
- `three_stars_in_the_south`, 315
- `three_stars_in_the_south()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 321, 324, 341, 346, 353, 359, 374
- `three_white_soldiers`, 318
- `three_white_soldiers()`, 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 324, 341, 346, 353, 359, 374
- `threeBlackCrows(three_black_crows)`, 300
- `threeIdenticalCrows(three_identical_crows)`, 303

- threeInside (three_inside), 306
- threeLineStrike (three_line_strike), 309
- threeOutside (three_outside), 312
- threeStarsInTheSouth
(three_stars_in_the_south), 315
- threeWhiteSoldiers
(three_white_soldiers), 318
- thrusting, 322
- thrusting(), 7, 15, 36, 41, 58, 63, 67, 70, 75,
78, 88, 91, 94, 97, 113, 117, 120,
123, 126, 130, 133, 136, 139, 142,
152, 155, 161, 164, 167, 170, 173,
177, 180, 183, 204, 207, 220, 233,
247, 250, 263, 267, 271, 278, 282,
285, 296, 299, 302, 305, 308, 311,
315, 318, 321, 341, 346, 353, 359,
374
- trading_volume, 325
- trading_volume(), 44, 46, 48, 212, 217, 227,
239
- tradingVolume (trading_volume), 325
- TRANGE (true_range), 342
- trend_cycle_mode, 329
- trend_cycle_mode(), 80, 82, 230, 275
- trendCycleMode (trend_cycle_mode), 329
- trendline, 327
- trendline(), 12, 38, 85, 100, 105, 144, 158,
188, 190, 192, 222, 273, 293, 333,
338, 361, 364, 368
- triangular_moving_average, 331, 333
- triangular_moving_average(), 12, 38, 85,
100, 105, 144, 158, 188, 190, 192,
222, 273, 293, 329, 338, 361, 364,
368
- triangularMovingAverage
(triangular_moving_average),
331
- TRIMA (triangular_moving_average), 331
- triple_exponential_average, 334
- triple_exponential_average(), 10, 17, 20,
24, 27, 33, 51, 60, 72, 102, 108, 110,
149, 194, 196, 199, 201, 210, 225,
235, 237, 241, 244, 288, 290, 350,
356, 370
- triple_exponential_moving_average, 336,
337
- triple_exponential_moving_average(),
12, 38, 85, 100, 105, 144, 158, 188,
190, 192, 222, 273, 293, 333,
361, 364, 368
- tripleExponentialAverage
(triple_exponential_average),
334
- tripleExponentialMovingAverage
(triple_exponential_moving_average),
336
- tristar, 339
- tristar(), 7, 15, 36, 41, 58, 63, 67, 70, 75,
78, 88, 91, 94, 97, 113, 117, 120,
123, 126, 130, 133, 136, 139, 142,
152, 155, 161, 164, 167, 170, 173,
177, 180, 183, 204, 207, 220, 233,
247, 250, 263, 267, 271, 278, 282,
285, 296, 299, 302, 305, 308, 311,
315, 318, 321, 324, 346, 353, 359,
374
- TRIX (triple_exponential_average), 334
- TRUE, 6, 9, 12, 14, 17, 19, 22, 24, 26, 28, 30,
32, 34, 38, 40, 44, 46, 48, 50, 57, 60,
62, 65, 68, 71, 74, 77, 80, 82, 84, 86,
90, 93, 96, 99, 102, 105, 107, 110,
112, 116, 119, 122, 125, 128, 131,
134, 138, 141, 144, 148, 150, 154,
157, 159, 162, 166, 169, 172, 176,
179, 182, 185, 187, 189, 192, 194,
196, 198, 200, 203, 206, 209, 211,
214, 216, 219, 222, 224, 227, 229,
231, 234, 237, 239, 241, 243, 245,
249, 251, 253–255, 257–259, 261,
266, 269, 272, 275, 277, 281, 284,
287, 290, 292, 295, 298, 301, 304,
307, 310, 313, 316, 319, 323, 326,
328, 330, 332, 335, 337, 340, 343,
345, 348, 350, 352, 355, 357, 361,
364, 366, 368, 370, 372
- true_range, 342
- true_range(), 30, 214
- trueRange (true_range), 342
- two_crows, 344
- two_crows(), 7, 15, 36, 41, 58, 63, 67, 70, 75,
78, 88, 91, 94, 97, 113, 117, 120,
123, 126, 130, 133, 136, 139, 142,
152, 155, 161, 164, 167, 170, 173,
177, 180, 183, 204, 207, 220, 233,
247, 250, 263, 267, 271, 278, 282,
285, 296, 299, 302, 305, 308, 311,

- [315, 318, 321, 324, 341, 353, 359, 374](#)
- [twoCrows \(two_crows\), 344](#)
- [typical_price, 347](#)
- [typical_price\(\), 22, 29, 185, 366](#)
- [typicalPrice \(typical_price\), 347](#)
- [TYPPRICE \(typical_price\), 347](#)
- [ultimate_oscillator, 349](#)
- [ultimate_oscillator\(\), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 356, 370](#)
- [ultimateOscillator \(ultimate_oscillator\), 349](#)
- [ULTOSC \(ultimate_oscillator\), 349](#)
- [unique3River \(unique_3_river\), 351](#)
- [unique_3_river, 351](#)
- [unique_3_river\(\), 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 359, 374](#)
- [unsmoothed_chande_momentum_oscillator, 354](#)
- [unsmoothed_chande_momentum_oscillator\(\), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 370](#)
- [unsmoothedChandeMomentumOscillator \(unsmoothed_chande_momentum_oscillator\), 354](#)
- [upside_gap_2_crows, 356](#)
- [upside_gap_2_crows\(\), 7, 15, 36, 41, 58, 63, 67, 70, 75, 78, 88, 91, 94, 97, 113, 117, 120, 123, 126, 130, 133, 136, 139, 142, 152, 155, 161, 164, 167, 170, 173, 177, 180, 183, 204, 207, 220, 233, 247, 250, 263, 267, 271, 278, 282, 285, 296, 299, 302, 305, 308, 311, 315, 318, 321, 324, 341, 346, 353, 374](#)
- [upsideGap2Crows \(upside_gap_2_crows\), 356](#)
- [VAR \(rolling_variance\), 259](#)
- [variable_moving_average_period, 360](#)
- [variable_moving_average_period\(\), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 293, 329, 333, 338, 364, 368](#)
- [variableMovingAveragePeriod \(variable_moving_average_period\), 360](#)
- [vector, 52](#)
- [VOLUME \(trading_volume\), 325](#)
- [volume_weighted_moving_average, 362, 364](#)
- [volume_weighted_moving_average\(\), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 293, 329, 333, 338, 361, 368](#)
- [volumeWeightedMovingAverage \(volume_weighted_moving_average\), 362](#)
- [VWMA \(volume_weighted_moving_average\), 362](#)
- [WCLPRICE \(weighted_close_price\), 365](#)
- [weighted_close_price, 365](#)
- [weighted_close_price\(\), 22, 29, 185, 348](#)
- [weighted_moving_average, 367, 368](#)
- [weighted_moving_average\(\), 12, 38, 85, 100, 105, 144, 158, 188, 190, 192, 222, 273, 293, 329, 333, 338, 361, 364](#)
- [weightedClosePrice \(weighted_close_price\), 365](#)
- [weightedMovingAverage \(weighted_moving_average\), 367](#)
- [williams_oscillator, 369](#)
- [williams_oscillator\(\), 10, 17, 20, 24, 27, 33, 51, 60, 72, 102, 108, 110, 149, 194, 196, 199, 201, 210, 225, 235, 237, 241, 244, 288, 290, 335, 350, 356](#)
- [williamsOscillator \(williams_oscillator\), 369](#)
- [WILLR \(williams_oscillator\), 369](#)
- [WMA \(weighted_moving_average\), 367](#)

xside_gap_3_methods, [371](#)
xside_gap_3_methods(), [7](#), [15](#), [36](#), [41](#), [58](#),
 [63](#), [67](#), [70](#), [75](#), [78](#), [88](#), [91](#), [94](#), [97](#),
 [113](#), [117](#), [120](#), [123](#), [126](#), [130](#), [133](#),
 [136](#), [139](#), [142](#), [152](#), [155](#), [161](#), [164](#),
 [167](#), [170](#), [173](#), [177](#), [180](#), [183](#), [204](#),
 [207](#), [220](#), [233](#), [247](#), [250](#), [263](#), [267](#),
 [271](#), [278](#), [282](#), [285](#), [296](#), [299](#), [302](#),
 [305](#), [308](#), [311](#), [315](#), [318](#), [321](#), [324](#),
 [341](#), [346](#), [353](#), [359](#)
xsideGap3Methods (xside_gap_3_methods),
 [371](#)