

Package ‘talib’

May 10, 2026

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

Version 0.9-2

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)

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>),
 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 2026-05-10 09:30:02 UTC

Contents

abandoned_baby	5
absolute_price_oscillator	8
acceleration_bands	10
advance_block	12
aroon	15
aroon_oscillator	17
ATOM	19
average_directional_movement_index	20
average_directional_movement_index_rating	22
average_price	25
average_true_range	26
balance_of_power	28
belt_hold	30
bollinger_bands	33
break_away	36
BTC	39
chaikin_accumulation_distribution_line	40
chaikin_accumulation_distribution_oscillator	42
chande_momentum_oscillator	44
chart	46
chart_themes	48
closing_marubozu	50
commodity_channel_index	53
concealing_baby_swallow	56
counter_attack	59
dark_cloud_cover	62
directional_movement_index	65
doji	67
doji_star	70
dominant_cycle_period	73
dominant_cycle_phase	75
double_exponential_moving_average	77
dragonfly_doji	79
engulfing	82
evening_doji_star	85
evening_star	88
exponential_moving_average	91

extended_moving_average_convergence_divergence	93
extended_parabolic_stop_and_reverse	95
fast_stochastic	98
fixed_moving_average_convergence_divergence	100
gaps_side_white	102
gravestone_doji	105
hammer	108
hanging_man	111
harami	114
harami_cross	117
high_wave	120
hikakke	123
hikakke_mod	126
homing_pigeon	129
indicator	132
intraday_movement_index	135
inverted_hammer	137
in_neck	140
kaufman_adaptive_moving_average	143
kicking	145
kicking_baby_length	148
ladder_bottom	151
long_legged_doji	154
long_line	157
marubozu	160
matching_low	163
mat_hold	166
median_price	169
mesa_adaptive_moving_average	170
midpoint_price	173
minus_directional_indicator	175
minus_directional_movement	177
momentum	179
money_flow_index	181
morning_doji_star	183
morning_star	186
moving_average_convergence_divergence	189
normalized_average_true_range	191
NVDA	193
on_balance_volume	194
on_neck	196
parabolic_stop_and_reverse	199
percentage_price_oscillator	201
phasor_components	203
piercing	206
plus_directional_indicator	209
plus_directional_movement	211
rate_of_change	213

ratio_of_change	215
relative_strength_index	217
rickshaw_man	219
rise_fall_3_methods	222
rolling_beta	225
rolling_correlation	227
rolling_max	228
rolling_min	229
rolling_standard_deviation	231
rolling_sum	232
rolling_variance	233
separating_lines	234
set_theme	237
shooting_star	239
short_line	242
simple_moving_average	245
sine_wave	247
spinning_top	249
SPY	252
stalled_pattern	253
stick_sandwich	256
stochastic	259
stochastic_relative_strength_index	261
t3_exponential_moving_average	264
takuri	266
tasuki_gap	269
three_black_crows	272
three_identical_crows	275
three_inside	278
three_line_strike	281
three_outside	284
three_stars_in_the_south	287
three_white_soldiers	290
thrusting	293
trading_volume	296
trendline	298
trend_cycle_mode	300
triangular_moving_average	302
triple_exponential_average	305
triple_exponential_moving_average	307
tristar	309
true_range	312
two_crows	314
typical_price	317
ultimate_oscillator	318
unique_3_river	321
upside_gap_2_crows	324
weighted_close_price	327

weighted_moving_average	328
williams_oscillator	330
xside_gap_3_methods	332

Index	336
--------------	------------

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.

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, eps = 0, 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> . Defaults to <code>~open + high + low + close</code> .

eps	(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 <code>model.frame</code> .

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.

`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,
  fast = 12,
  slow = 26,
  ma = SMA(n = 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 . Defaults to <code>~close</code> .
<code>fast</code>	(integer). Period for the fast Moving Average (MA).
<code>slow</code>	(integer). Period for the slow Moving Average (MA).
<code>ma</code>	(list). The type of Moving Average (MA) used for the fast and slow MA. SMA by default.
<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 .

Value

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

APO [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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(), rate_of_change(), 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::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 *Acceleration Bands*

Description

acceleration_bands() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 20, 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 . Defaults to <code>~high + low + close</code> .
<code>n</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 .

Value

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

UpperBand [double](#)

MiddleBand [double](#)

LowerBand [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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.

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 . 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 .

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.

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, n = 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 . Defaults to <code>~high + low</code> .
<code>n</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 .

Value

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

AroonDown [double](#)

AroonUp [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [absolute_price_oscillator\(\)](#), [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\(\)](#), [rate_of_change\(\)](#), [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::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.

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, n = 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 . Defaults to <code>~high + low</code> .
<code>n</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 .

Value

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

AROONOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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
  )
}
```

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_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.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</code>	(<i>integer</i>). Lookback period (window size). A positive <i>integer</i> of <i>length</i> 1.
<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> .

Value

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

ADX `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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.

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,
  n = 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 . Defaults to <code>~high + low + close</code> .
<code>n</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 .

Value

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

ADXR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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
  )
}
```

average_price	<i>Average Price</i>
---------------	----------------------

Description

average_price() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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> . 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> .

Value

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

AVGPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [median_price\(\)](#), [midpoint_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 *Average True Range*

Description

`average_true_range()` is a generic S3 function that preserves the input [class](#): [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out.

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, n = 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 . Defaults to <code>~high + low + close</code> .
<code>n</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 .

Value

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

ATR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicator: [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.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . 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 .

Value

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

BOP [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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_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\(\)](#), [rate_of_change\(\)](#), [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::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.

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 <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> . 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> .

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
  )
}

```

bollinger_bands

Bollinger Bands

Description

`bollinger_bands()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

`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,
  ma = SMA(n = 5),
  sd = 2,
  sd_down,
  sd_up,
  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 . Defaults to <code>~close</code> .
<code>ma</code>	(list). The type of Moving Average (MA) used for the MiddleBand. SMA by default.
<code>sd</code>	(double). Deviation multiplier for the upper and lower band.
<code>sd_down</code>	(double). Optional. Deviation multiplier for lower band
<code>sd_up</code>	(double). Optional. Deviation multiplier for upper band
<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 .

Value

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

UpperBand [double](#)

MiddleBand [double](#)

LowerBand [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: [acceleration_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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

Break Away

Description

break_away() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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.

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

<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> . Defaults to <code>~high + low + 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.

... Additional parameters passed into `model.frame`.

Value

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

AD `double`

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicator: `chaikin_accumulation_distribution_oscillator()`, `on_balance_volume()`, `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.

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,
  fast = 3,
  slow = 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 . Defaults to <code>~high + low + close + volume</code> .
fast	(integer). Period for the fast Moving Average (MA).
slow	(integer). Period for the slow Moving Average (MA).
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 .

Value

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

ADOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicator: [chaikin_accumulation_distribution_line\(\)](#), [on_balance_volume\(\)](#), [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
)
}
```

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.

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, n = 14, 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 . Defaults to <code>~close</code> .
n	(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 .

Value

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

CMO [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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")
```

Description

closing_marubozu() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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
  )
}
```

Description

`commodity_channel_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

CCI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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.

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 <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> . 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. |

... Additional parameters passed into `model.frame`.

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	Counter Attack
----------------	----------------

Description

counter_attack() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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.

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, eps = 0, 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> . Defaults to <code>~open + high + low + close</code> .
<code>eps</code>	(<code>double</code>). Penetration threshold for candlestick pattern recognition, expressed as a fraction of the candle body. A <code>double</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`.

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
  )
}
```

directional_movement_index

Directional Movement Index

Description

`directional_movement_index()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</code>	(<i>integer</i>). Lookback period (window size). A positive <i>integer</i> of length 1.
<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.

... Additional parameters passed into `model.frame`.

Value

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

DX `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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.

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`. 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`.

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.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . 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 .

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.

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 . Defaults to <code>~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 .

Value

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

HT_DCPERIOD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicator: [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.

`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

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 . Defaults to <code>~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 .

Value

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

HT_DCPHASE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicator: [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.

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, cols, n = 30, 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 . Defaults to <code>~close</code> .
n	(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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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
  )
}

```

dragonfly_doji

Dragonfly Doji

Description

dragonfly_doji() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x An OHLC-V series coercible to [data.frame](#).

cols ([formula](#)). An optional 4-variable [formula](#) selecting columns from x via [model.frame](#). 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](#).

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
  )
}
```

engulfing

Engulfing

Description

engulfing() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . 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 .

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.

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, eps = 0, 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 . Defaults to <code>~open + high + low + close</code> .
<code>eps</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 .

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:

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.

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, eps = 0, 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 . Defaults to ~open + high + low + close.
eps	(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 .

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.

`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, cols, n = 30, 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 . Defaults to <code>~close</code> .
n	(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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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

Moving Average Convergence Divergence (Extended)

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.

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,
  fast = SMA(n = 12),
  slow = SMA(n = 26),
  signal = SMA(n = 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 . Defaults to <code>~close</code> .
<code>fast</code>	(list). Period and Moving Average (MA) type for the fast MA. EMA by default.
<code>slow</code>	(list). Period and Moving Average (MA) type for the slow MA. EMA by default.
<code>signal</code>	(list). Period and Moving Average (MA) type for the signal MA. EMA by default.
<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 .

Value

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

MACD [double](#)

MACDSignal [double](#)

MACDHist [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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 Stop and Reverse (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.

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,
  init = 0,
  offset = 0,
  init_long = 0.02,
  long = 0.02,
  max_long = 0.2,
  init_short = 0.02,
  short = 0.02,
  max_short = 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 . Defaults to <code>~high + low</code> .
<code>init</code>	(double). Start value and direction. 0 for Auto, >0 for Long, <0 for Short.
<code>offset</code>	(double). Offset added/removed to initial stop on short/long reversal.

init_long	(double). Acceleration factor initial value for the Long direction.
long	(double). Acceleration factor for the Long direction.
max_long	(double). Acceleration factor maximum value for the Long direction.
init_short	(double). Acceleration factor initial value for the Short direction.
short	(double). Acceleration factor for the Short direction.
max_short	(double). Acceleration factor maximum value for the Short direction.
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> .

Value

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

SAREXT double

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: `acceleration_bands()`, `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `kaufman_adaptive_moving_average()`, `mesa_adaptive_moving_average()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_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

Fast Stochastic

Description

fast_stochastic() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, fastk = 5, fastd = SMA(n = 3), 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 . Defaults to <code>~high + low + close</code> .
fastk	(integer). Period for the fast-k line.
fastd	(list). Period and Moving Average (MA) type for the fast-d line. SMA by default.
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 .

Value

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

FastK [double](#)

FastD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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 (Fixed)

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.

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,
  signal = 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 . Defaults to <code>~close</code> .
<code>signal</code>	(integer). Period for the signal Moving Average (MA).
<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 .

Value

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

MACD [double](#)

MACDSignal [double](#)

MACDHist [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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.

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 . 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 .

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
  )
}
```

Description

`gravestone_doji()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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
  )
}
```

Description

hammer() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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
  )
}
```

Description

`hanging_man()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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
  )
}
```

Description

`harami()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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
  )
}
```

Description

`harami_cross()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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
  )
}
```


Description

high_wave() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x	An OHLC-V series coercible to <code>data.frame</code> .
cols	(<i>formula</i>). An optional 4-variable <i>formula</i> selecting columns from x via <code>model.frame</code> . Defaults to <code>~open + high + low + close</code> .
na.bridge	(<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.
...	Additional parameters passed into <code>model.frame</code> .

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.

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> . 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> .

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`:

CDLHIKKAKE *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.

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

<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> . 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> .

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
  )
}
```

Description

`homing_pigeon()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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
  )
}
```

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

FUN	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> .
...	In single indicator mode: arguments passed to FUN (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, n = 14)

## multiple indicators on the same panel
talib::chart(BTC)
talib::indicator(
  talib::RSI(n = 10),
  talib::RSI(n = 14),
  talib::RSI(n = 21)
)

## reset chart state
talib::chart()
```

intraday_movement_index

Intraday Movement 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.

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, n = 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 2-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~open + close</code> .
<code>n</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`.

Value

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

IMI `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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
)
}

```

inverted_hammer

Inverted Hammer

Description

`inverted_hammer()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

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

`cols` (`formula`). An optional 4-variable `formula` selecting columns from `x` via `model.frame`. 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`.

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

Description

in_neck() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . 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 .

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.

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, cols, n = 30, 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 . Defaults to <code>~close</code> .
n	(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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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.

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

x An OHLC-V series coercible to [data.frame](#).

cols ([formula](#)). An optional 4-variable [formula](#) selecting columns from x via [model.frame](#). 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](#).

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 *Kicking Baby Length*

Description

kicking_baby_length() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x An OHLC-V series coercible to [data.frame](#).

cols ([formula](#)). An optional 4-variable [formula](#) selecting columns from x via [model.frame](#). 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](#).

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.

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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . 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 .

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

Description

long_line() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . 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 .

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
)
}
```

marubozu

Marubozu

Description

marubozu() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 . 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 .

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.

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 . 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 .

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.

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, eps = 0, 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 . Defaults to ~open + high + low + close.
eps	(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 .

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.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 2-variable formula selecting columns from x via model.frame . Defaults to <code>~high + low</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 .

Value

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

MEDPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_price\(\)](#), [midpoint_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.

`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,
  cols,
  n = 30,
  fast = 0.5,
  slow = 0.05,
  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`. Defaults to `~close`.

n	(integer). Lookback period (window size). A positive integer of length 1.
fast	(double). Upper limit of the adaptive smoothing factor (alpha) used in the MESA algorithm. A double in [0.01, 0.99]. 0.5 by default.
slow	(double). Lower limit of the adaptive smoothing factor (alpha) used in the MESA algorithm. A double in [0.01, 0.99]. 0.05 by default.
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> .

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 Study: `acceleration_bands()`, `bollinger_bands()`, `double_exponential_moving_average()`, `exponential_moving_average()`, `extended_parabolic_stop_and_reverse()`, `kaufman_adaptive_moving_average()`, `parabolic_stop_and_reverse()`, `simple_moving_average()`, `t3_exponential_moving_average()`, `trendline()`, `triangular_moving_average()`, `triple_exponential_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_price

Midpoint Price

Description

midpoint_price() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 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 . Defaults to <code>~high + low</code> .
n	(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 .

Value

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

MIDPRICE [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::midpoint_price(BTC)

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

 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.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</code>	(<i>integer</i>). Lookback period (window size). A positive <i>integer</i> of length 1.
<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.

... Additional parameters passed into `model.frame`.

Value

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

MINUS_DI `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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.

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, n = 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> . Defaults to <code>~high + low</code> .
<code>n</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`.

Value

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

MINUS_DM double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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.

`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, n = 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 . Defaults to <code>~close</code> .
n	(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 .

Value

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

MOM [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_averaging_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	<i>Money Flow Index</i>
------------------	-------------------------

Description

money_flow_index() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 14, 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](#). Defaults to `~high + low + close + volume`.

n ([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](#).

Value

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

MFI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_averaging_oscillator\(\)](#), [ultimate_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 *Morning Doji Star*

Description

morning_doji_star() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, eps = 0, 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 . Defaults to ~open + high + low + close.
eps	(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 .

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:

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.

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, eps = 0, 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 . Defaults to <code>~open + high + low + close</code> .
eps	(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 .

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.

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,
  fast = 12,
  slow = 26,
  signal = 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 . Defaults to <code>~close</code> .
<code>fast</code>	(integer). Period for the fast Moving Average (MA).
<code>slow</code>	(integer). Period for the slow Moving Average (MA).
<code>signal</code>	(integer). Period for the signal Moving Average (MA).
<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 .

Value

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

MACD [double](#)

MACDSignal [double](#)

MACDHist [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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
  )
}
```

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.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</code>	(<i>integer</i>). Lookback period (window size). A positive <i>integer</i> of length 1.
<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> .

Value

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

NATR *double*

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicator: [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

NVIDIA Corporation (NVDA)

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.

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 data.frame .
<code>cols</code>	(formula). An optional 2-variable formula selecting columns from <code>x</code> via model.frame . 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 .

Value

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

OBV [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicator: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [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

Description

on_neck() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x An OHLC-V series coercible to [data.frame](#).

cols ([formula](#)). An optional 4-variable [formula](#) selecting columns from x via [model.frame](#). 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](#).

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 Stop and Reverse (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.

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,
  acceleration = 0.02,
  maximum = 0.2,
  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](#). Defaults to `~high + low`.

acceleration ([double](#)). Acceleration factor used up to the maximum value.

maximum ([double](#)). Acceleration factor maximum value.

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](#).

Value

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

SAR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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.

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,
  fast = 12,
  slow = 26,
  ma = SMA(n = 9),
  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>cols</code>	(<code>formula</code>). An optional 1-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~close</code> .
<code>fast</code>	(<code>integer</code>). Period for the fast Moving Average (MA).
<code>slow</code>	(<code>integer</code>). Period for the slow Moving Average (MA).
<code>ma</code>	(<code>list</code>). The type of Moving Average (MA) used for the fast and slow MA. <code>SMA</code> by default.
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

PPO `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [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::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
  )
}
```

Description

`phasor_components()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

`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 <code>data.frame</code> . Alternatively, <code>x</code> may also be supplied as a <code>double</code> vector. |
| <code>cols</code> | (<code>formula</code>). An optional 1-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~close</code> . |
| <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. |
| <code>...</code> | Additional parameters passed into <code>model.frame</code> . |

Value

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

InPhase [double](#)

Quadrature [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicator: [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***Description**

`piercing()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</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`.

Value

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

PLUS_DI `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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.

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, n = 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 2-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~high + low</code> .
<code>n</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> 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`.

Value

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

PLUS_DM double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `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::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
)
}

```

rate_of_change	<i>Rate of Change</i>
----------------	-----------------------

Description

rate_of_change() is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

rate_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

```
rate_of_change(x, cols, n = 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 . Defaults to <code>~close</code> .
n	(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 .

Value

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

ROC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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::rate_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::rate_of_change
  )
}

```

ratio_of_change

Ratio of Change

Description

ratio_of_change() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 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 . Defaults to <code>~close</code> .
<code>n</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 .

Value

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

ROCR [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_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::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.

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, n = 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 . Defaults to <code>~close</code> .
<code>n</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 .

Value

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

RSI [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [ratio_of_change\(\)](#), [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::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
  )
}
```

rickshaw_man

Rickshaw Man

Description

`rickshaw_man()` is a generic S3 function that preserves the input `class`: [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out.

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 data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . 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 .

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
  )
}
```

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.

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> . 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> .

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
  )
}
```

Description

rolling_beta() is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector 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, n = 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> .
<code>n</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.

Value

A `double` vector with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: `rolling_correlation()`, `rolling_max()`, `rolling_min()`, `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 *Rolling Correlation*

Description

rolling_correlation() is a generic S3 function that preserves the input `class: double` vector in, `double` vector 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, n = 30, na.bridge = FALSE)
```

Arguments

`x, y` ((`double`), (`double`)). A pair of `double` vectors of equal `length`.

`n` (`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.

Value

A `double` vector with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: [rolling_beta\(\)](#), [rolling_max\(\)](#), [rolling_min\(\)](#), [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)
```

rolling_max

Rolling Max

Description

`rolling_max()` is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector 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_max(x, n = 30, na.bridge = FALSE)
```

Arguments

`x` (`double`). A `double` vector.

`n` (`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.

Value

A double vector with the same length of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_min\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

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

## calculate the indicator
## Open
output <- talib::rolling_max(x = BTC[[1]])

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

rolling_min

Rolling Min

Description

rolling_min() is a generic S3 function that preserves the input class: double vector in, double vector 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_min(x, n = 30, na.bridge = FALSE)
```

Arguments

x (double). A double vector.

n (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.

Value

A double vector with the same length of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_max\(\)](#), [rolling_standard_deviation\(\)](#), [rolling_sum\(\)](#), [rolling_variance\(\)](#)

Examples

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

## calculate the indicator
## Open
output <- talib::rolling_min(x = BTC[[1]])

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

rolling_standard_deviation
Rolling Standard Deviation

Description

rolling_standard_deviation() is a generic S3 function that preserves the input `class: double` vector in, `double` vector 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, n = 5, k = 1, na.bridge = FALSE)
```

Arguments

<code>x</code>	(<code>double</code>). A <code>double</code> vector.
<code>n</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of <code>length 1</code> .
<code>k</code>	(<code>double</code>). Multiplier for the standard deviation.
<code>na.bridge</code>	(<code>logical</code>). A <code>logical</code> of <code>length 1</code> . <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.

Value

A `double` vector with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: `rolling_beta()`, `rolling_correlation()`, `rolling_max()`, `rolling_min()`, `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

Rolling Sum

Description

rolling_sum() is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector 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, n = 30, na.bridge = FALSE)
```

Arguments

<code>x</code>	(<code>double</code>). A <code>double</code> vector.
<code>n</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.

Value

A `double` vector with the same `length` of `x`

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: `rolling_beta()`, `rolling_correlation()`, `rolling_max()`, `rolling_min()`, `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>Rolling Standard Deviation</i>
------------------	-----------------------------------

Description

`rolling_variance()` is a generic S3 function that preserves the input `class`: `double` vector in, `double` vector 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, n = 5, k = 1, na.bridge = FALSE)
```

Arguments

x	(double). A double vector.
n	(integer). Lookback period (window size). A positive integer of length 1.
k	multiplier
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.

Value

A double vector with the same length of x

Author(s)

Serkan Korkmaz

See Also

Other Rolling Statistic: [rolling_beta\(\)](#), [rolling_correlation\(\)](#), [rolling_max\(\)](#), [rolling_min\(\)](#), [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)
```

Description

`separating_lines()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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
  )
}
```

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: <code>bearish_body</code> , <code>bearish_wick</code> , <code>bearish_border</code> , <code>bullish_body</code> , <code>bullish_wick</code> , <code>bullish_border</code> , <code>background_color</code> , <code>foreground_color</code> , <code>text_color</code> , <code>gridcolor</code> , <code>threshold_color</code> , and <code>colorway</code> (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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

`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, cols, n = 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 . Defaults to <code>~close</code> .
<code>n</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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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.

`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

<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 . 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 .

Value

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

Sine [double](#)

LeadSine [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicator: [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.

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 . 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 .

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

SPDR S&P 500 ETF (SPY)

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.

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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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,
  fastk = 5,
  slowk = SMA(n = 3),
  slowd = SMA(n = 3),
  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 . Defaults to <code>~high + low + close</code> .
<code>fastk</code>	(integer). Period for the fast-k line.
<code>slowk</code>	(list). Period and Moving Average (MA) type for the slow-k line. SMA by default.
<code>slowd</code>	(list). Period and Moving Average (MA) type for the slow-d line. SMA by default.
<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 .

Value

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

SlowK [double](#)

SlowD [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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(), rate_of_change(), ratio_of_change(),  
relative_strength_index(), 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::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.

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,
  n = 14,
  fastk = 5,
  fastd = SMA(n = 3),
  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>cols</code>	(<code>formula</code>). An optional 1-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~close</code> .
<code>n</code>	(<code>integer</code>). Lookback period (window size). A positive <code>integer</code> of length 1.
<code>fastk</code>	(<code>integer</code>). Period for the fast-k line.
<code>fastd</code>	(<code>list</code>). Period and Moving Average (MA) type for the fast-d line. <code>SMA</code> by default.
<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`.

Value

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

FastK `double`

FastD `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `triple_exponential_average()`, `ultimate_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.

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,
  cols,
```



```

n = 5,
vfactor = 0.7,
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 . Defaults to <code>~close</code> .
n	(integer). Lookback period (window size). A positive integer of length 1.
vfactor	(double). Volume Factor controlling the smoothing weight of the T3 curve. A double in $[0, 1]$: 0 collapses T3 to a standard triple EMA, larger values shift the curve closer to a DEMA. 0.7 by default, following Tillson (1998).
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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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

Description

`takuri()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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 *Three Black Crows*

Description

`three_black_crows()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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:

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

Description

three_inside() is a generic S3 function that preserves the input `class`: [data.frame](#) in, [data.frame](#) out; [matrix](#) in, [matrix](#) out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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

Description

`three_outside()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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 <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> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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
  )
}
```

three_white_soldiers *Three White Soldiers*

Description

`three_white_soldiers()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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

Description

`thrusting()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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>	(<code>formula</code>). An optional 4-variable <code>formula</code> selecting columns from <code>x</code> via <code>model.frame</code> . Defaults to <code>~open + high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

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.

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,
  ma = list(SMA(n = 7), SMA(n = 15)),
  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> . Defaults to <code>~volume + open + close</code> .
<code>ma</code>	A list of MA specifications.
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

VOLUME `double`

SMA7 `double`

SMA15 `double`

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicator: [chaikin_accumulation_distribution_line\(\)](#), [chaikin_accumulation_distribution_oscillator\(\)](#), [on_balance_volume\(\)](#)

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.

`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 . 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 .

Value

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

HT_TRENDLINE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_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.

`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 . 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 .

Value

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

HT_TRENDMODE [integer](#)

Author(s)

Serkan Korkmaz

See Also

Other Cycle Indicator: [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.

`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, cols, n = 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 . Defaults to <code>~close</code> .
<code>n</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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triple_exponential_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
      Triple Exponential Average
```

Description

`triple_exponential_average()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

`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, n = 30, 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`. Defaults to `~close`.
- `n` (`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`.

Value

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

TRIX double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: `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()`, `rate_of_change()`, `ratio_of_change()`, `relative_strength_index()`, `stochastic()`, `stochastic_relative_strength_index()`, `ultimate_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.

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, cols, n = 30, 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 . Defaults to <code>~close</code> .
n	(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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_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

Description

`tristar()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 4-variable formula selecting columns from x via model.frame . 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 .

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.

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](#). 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](#).

Value

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

TRANGE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Volatility Indicator: [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.

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 data.frame .
<code>cols</code>	(formula). An optional 4-variable formula selecting columns from <code>x</code> via model.frame . 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 .

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.

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

x	An OHLC-V series coercible to data.frame .
cols	(formula). An optional 3-variable formula selecting columns from x via model.frame . Defaults to <code>~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 .

Value

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

TYPPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_price\(\)](#), [median_price\(\)](#), [midpoint_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)
```

Description

ultimate_oscillator() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = c(7, 14, 28), 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> . Defaults to <code>~high + low + close</code> .
<code>n</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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

ULTOSC [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_averaging_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	<i>Unique Three River</i>
----------------	---------------------------

Description

`unique_3_river()` is a generic S3 function that preserves the input `class: data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . 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> .

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
  )
}
```

upside_gap_2_crows *Upside Gap Two Crows*

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.

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> . 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> .

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
  )
}
```

 weighted_close_price *Weighted Close Price*

Description

weighted_close_price() is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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 <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> . Defaults to <code>~high + low + close</code> .
<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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

WCLPRICE [double](#)

Author(s)

Serkan Korkmaz

See Also

Other Price Transform: [average_price\(\)](#), [median_price\(\)](#), [midpoint_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.

`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, cols, n = 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 . Defaults to <code>~close</code> .
<code>n</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 .

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 Study: [acceleration_bands\(\)](#), [bollinger_bands\(\)](#), [double_exponential_moving_average\(\)](#), [exponential_moving_average\(\)](#), [extended_parabolic_stop_and_reverse\(\)](#), [kaufman_adaptive_moving_average\(\)](#), [mesa_adaptive_moving_average\(\)](#), [parabolic_stop_and_reverse\(\)](#), [simple_moving_average\(\)](#), [t3_exponential_moving_average\(\)](#), [trendline\(\)](#), [triangular_moving_average\(\)](#), [triple_exponential_moving_av](#)

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 *Williams %R*

Description

`williams_oscillator()` is a generic S3 function that preserves the input `class`: `data.frame` in, `data.frame` out; `matrix` in, `matrix` out.

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, n = 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> . Defaults to <code>~high + low + close</code> .
<code>n</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.
<code>...</code>	Additional parameters passed into <code>model.frame</code> .

Value

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

WILLR `double`

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: [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\(\)](#), [rate_of_change\(\)](#), [ratio_of_change\(\)](#), [relative_strength_index\(\)](#), [stochastic\(\)](#), [stochastic_relative_strength_index\(\)](#), [triple_exponential_average\(\)](#), [ultimate_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.

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 . 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 .

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, 46
 - chart_themes, 48
 - indicator, 132
 - set_theme, 237
- * **Cycle Indicator**
 - dominant_cycle_period, 73
 - dominant_cycle_phase, 75
 - phasor_components, 203
 - sine_wave, 247
 - trend_cycle_mode, 300
- * **Financial Data**
 - ATOM, 19
 - BTC, 39
 - NVDA, 193
 - SPY, 252
- * **Momentum Indicator**
 - absolute_price_oscillator, 8
 - aroon, 15
 - aroon_oscillator, 17
 - average_directional_movement_index, 20
 - average_directional_movement_index_rating, 22
 - balance_of_power, 28
 - chande_momentum_oscillator, 44
 - commodity_channel_index, 53
 - directional_movement_index, 65
 - extended_moving_average_convergence_divergence, 93
 - fast_stochastic, 98
 - fixed_moving_average_convergence_divergence, 100
 - intraday_movement_index, 135
 - minus_directional_indicator, 175
 - minus_directional_movement, 177
 - momentum, 179
 - money_flow_index, 181
 - moving_average_convergence_divergence, 189
 - percentage_price_oscillator, 201
 - plus_directional_indicator, 209
 - plus_directional_movement, 211
 - rate_of_change, 213
 - ratio_of_change, 215
 - relative_strength_index, 217
 - stochastic, 259
 - stochastic_relative_strength_index, 261
 - triple_exponential_average, 305
 - ultimate_oscillator, 318
 - williams_oscillator, 330
- * **OHLCV**
 - ATOM, 19
 - BTC, 39
 - NVDA, 193
 - SPY, 252
- * **Overlap Study**
 - acceleration_bands, 10
 - bollinger_bands, 33
 - double_exponential_moving_average, 77
 - exponential_moving_average, 91
 - extended_parabolic_stop_and_reverse, 95
 - kaufman_adaptive_moving_average, 143
 - tesa_adaptive_moving_average, 170
 - parabolic_stop_and_reverse, 199
 - simple_moving_average, 245
 - t3_exponential_moving_average, 264
 - trendline, 298
 - triangular_moving_average, 302
 - triple_exponential_moving_average, 307
 - weighted_moving_average, 328
- * **Pattern Recognition**
 - abandoned_baby, 5

- advance_block, 12
- belt_hold, 30
- break_away, 36
- closing_marubozu, 50
- concealing_baby_swallow, 56
- counter_attack, 59
- dark_cloud_cover, 62
- doji, 67
- doji_star, 70
- dragonfly_doji, 79
- engulfing, 82
- evening_doji_star, 85
- evening_star, 88
- gaps_side_white, 102
- gravestone_doji, 105
- hammer, 108
- hanging_man, 111
- harami, 114
- harami_cross, 117
- high_wave, 120
- hikakke, 123
- hikakke_mod, 126
- homing_pigeon, 129
- in_neck, 140
- inverted_hammer, 137
- kicking, 145
- kicking_baby_length, 148
- ladder_bottom, 151
- long_legged_doji, 154
- long_line, 157
- marubozu, 160
- mat_hold, 166
- matching_low, 163
- morning_doji_star, 183
- morning_star, 186
- on_neck, 196
- piercing, 206
- rickshaw_man, 219
- rise_fall_3_methods, 222
- separating_lines, 234
- shooting_star, 239
- short_line, 242
- spinning_top, 249
- stalled_pattern, 253
- stick_sandwich, 256
- takuri, 266
- tasuki_gap, 269
- three_black_crows, 272
- three_identical_crows, 275
- three_inside, 278
- three_line_strike, 281
- three_outside, 284
- three_stars_in_the_south, 287
- three_white_soldiers, 290
- thrusting, 293
- tristar, 309
- two_crows, 314
- unique_3_river, 321
- upside_gap_2_crows, 324
- xside_gap_3_methods, 332
- * Price Transform**
 - average_price, 25
 - median_price, 169
 - midpoint_price, 173
 - typical_price, 317
 - weighted_close_price, 327
- * Rolling Statistic**
 - rolling_beta, 225
 - rolling_correlation, 227
 - rolling_max, 228
 - rolling_min, 229
 - rolling_standard_deviation, 231
 - rolling_sum, 232
 - rolling_variance, 233
- * Volatility Indicator**
 - average_true_range, 26
 - normalized_average_true_range, 191
 - true_range, 312
- * Volume Indicator**
 - chaikin_accumulation_distribution_line, 40
 - chaikin_accumulation_distribution_oscillator, 42
 - on_balance_volume, 194
 - trading_volume, 296
- * algorithmic trading**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 10
 - advance_block, 12
 - aroon, 15
 - aroon_oscillator, 17
 - average_directional_movement_index, 20
 - average_directional_movement_index_rating, 22

- average_price, 25
- average_true_range, 26
- balance_of_power, 28
- belt_hold, 30
- bollinger_bands, 33
- break_away, 36
- chaikin_accumulation_distribution_line, 40
- chaikin_accumulation_distribution_oscillator, 42
- chande_momentum_oscillator, 44
- closing_marubozu, 50
- commodity_channel_index, 53
- concealing_baby_swallow, 56
- counter_attack, 59
- dark_cloud_cover, 62
- directional_movement_index, 65
- doji, 67
- doji_star, 70
- dominant_cycle_period, 73
- dominant_cycle_phase, 75
- double_exponential_moving_average, 77
- dragonfly_doji, 79
- engulfing, 82
- evening_doji_star, 85
- evening_star, 88
- exponential_moving_average, 91
- extended_moving_average_convergence_divergence, 93
- extended_parabolic_stop_and_reverse, 95
- fast_stochastic, 98
- fixed_moving_average_convergence_divergence, 100
- gaps_side_white, 102
- gravestone_doji, 105
- hammer, 108
- hanging_man, 111
- harami, 114
- harami_cross, 117
- high_wave, 120
- hikakke, 123
- hikakke_mod, 126
- homing_pigeon, 129
- in_neck, 140
- intraday_movement_index, 135
- inverted_hammer, 137
- kaufman_adaptive_moving_average, 143
- kicking, 145
- kicking_baby_length, 148
- ladder_bottom, 151
- long_legged_doji, 154
- long_line, 157
- marubozu, 160
- mat_hold, 166
- matching_low, 163
- median_price, 169
- mesa_adaptive_moving_average, 170
- midpoint_price, 173
- minus_directional_indicator, 175
- minus_directional_movement, 177
- momentum, 179
- money_flow_index, 181
- morning_doji_star, 183
- morning_star, 186
- moving_average_convergence_divergence, 189
- normalized_average_true_range, 191
- on_balance_volume, 194
- on_neck, 196
- parabolic_stop_and_reverse, 199
- percentage_price_oscillator, 201
- phasor_components, 203
- piercing, 206
- plus_directional_indicator, 209
- plus_directional_movement, 211
- rate_of_change, 213
- ratio_of_change, 215
- relative_strength_index, 217
- rickshaw_man, 219
- rise_fall_3_methods, 222
- rolling_beta, 225
- rolling_correlation, 227
- rolling_max, 228
- rolling_min, 229
- rolling_standard_deviation, 231
- rolling_sum, 232
- rolling_variance, 233
- separating_lines, 234
- shooting_star, 239
- short_line, 242
- simple_moving_average, 245
- sine_wave, 247
- spinning_top, 249

- stalled_pattern, 253
- stick_sandwich, 256
- stochastic, 259
- stochastic_relative_strength_index, 261
- t3_exponential_moving_average, 264
- takuri, 266
- tasuki_gap, 269
- three_black_crows, 272
- three_identical_crows, 275
- three_inside, 278
- three_line_strike, 281
- three_outside, 284
- three_stars_in_the_south, 287
- three_white_soldiers, 290
- thrusting, 293
- trading_volume, 296
- trend_cycle_mode, 300
- trendline, 298
- triangular_moving_average, 302
- triple_exponential_average, 305
- triple_exponential_moving_average, 307
- tristar, 309
- true_range, 312
- two_crows, 314
- typical_price, 317
- ultimate_oscillator, 318
- unique_3_river, 321
- upside_gap_2_crows, 324
- weighted_close_price, 327
- weighted_moving_average, 328
- williams_oscillator, 330
- xside_gap_3_methods, 332
- * datasets**
 - ATOM, 19
 - BTC, 39
 - NVDA, 193
 - SPY, 252
- * finance**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 10
 - advance_block, 12
 - aroon, 15
 - aroon_oscillator, 17
 - average_directional_movement_index, 20
 - average_directional_movement_index_rating, 22
 - average_price, 25
 - average_true_range, 26
 - balance_of_power, 28
 - belt_hold, 30
 - bollinger_bands, 33
 - break_away, 36
 - chaikin_accumulation_distribution_line, 40
 - chaikin_accumulation_distribution_oscillator, 42
 - chande_momentum_oscillator, 44
 - closing_marubozu, 50
 - commodity_channel_index, 53
 - concealing_baby_swallow, 56
 - counter_attack, 59
 - dark_cloud_cover, 62
 - directional_movement_index, 65
 - doji, 67
 - doji_star, 70
 - dominant_cycle_period, 73
 - dominant_cycle_phase, 75
 - double_exponential_moving_average, 77
 - dragonfly_doji, 79
 - engulfing, 82
 - evening_doji_star, 85
 - evening_star, 88
 - exponential_moving_average, 91
 - extended_moving_average_convergence_divergence, 93
 - extended_parabolic_stop_and_reverse, 95
 - fast_stochastic, 98
 - fixed_moving_average_convergence_divergence, 100
 - gaps_side_white, 102
 - gravestone_doji, 105
 - hammer, 108
 - hanging_man, 111
 - harami, 114
 - harami_cross, 117
 - high_wave, 120
 - hikakke, 123
 - hikakke_mod, 126
 - homing_pigeon, 129
 - in_neck, 140

- intraday_movement_index, 135
- inverted_hammer, 137
- kaufman_adaptive_moving_average, 143
- kicking, 145
- kicking_baby_length, 148
- ladder_bottom, 151
- long_legged_doji, 154
- long_line, 157
- marubozu, 160
- mat_hold, 166
- matching_low, 163
- median_price, 169
- mesa_adaptive_moving_average, 170
- midpoint_price, 173
- minus_directional_indicator, 175
- minus_directional_movement, 177
- momentum, 179
- money_flow_index, 181
- morning_doji_star, 183
- morning_star, 186
- moving_average_convergence_divergence, 189
- normalized_average_true_range, 191
- on_balance_volume, 194
- on_neck, 196
- parabolic_stop_and_reverse, 199
- percentage_price_oscillator, 201
- phasor_components, 203
- piercing, 206
- plus_directional_indicator, 209
- plus_directional_movement, 211
- rate_of_change, 213
- ratio_of_change, 215
- relative_strength_index, 217
- rickshaw_man, 219
- rise_fall_3_methods, 222
- rolling_beta, 225
- rolling_correlation, 227
- rolling_max, 228
- rolling_min, 229
- rolling_standard_deviation, 231
- rolling_sum, 232
- rolling_variance, 233
- separating_lines, 234
- shooting_star, 239
- short_line, 242
- simple_moving_average, 245
- sine_wave, 247
- spinning_top, 249
- stalled_pattern, 253
- stick_sandwich, 256
- stochastic, 259
- stochastic_relative_strength_index, 261
- t3_exponential_moving_average, 264
- takuri, 266
- tasuki_gap, 269
- three_black_crows, 272
- three_identical_crows, 275
- three_inside, 278
- three_line_strike, 281
- three_outside, 284
- three_stars_in_the_south, 287
- three_white_soldiers, 290
- thrusting, 293
- trading_volume, 296
- trend_cycle_mode, 300
- trendline, 298
- triangular_moving_average, 302
- triple_exponential_average, 305
- triple_exponential_moving_average, 307
- tristar, 309
- true_range, 312
- two_crows, 314
- typical_price, 317
- ultimate_oscillator, 318
- unique_3_river, 321
- upside_gap_2_crows, 324
- weighted_close_price, 327
- weighted_moving_average, 328
- williams_oscillator, 330
- xside_gap_3_methods, 332
- * technical analysis**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 10
 - advance_block, 12
 - aroon, 15
 - aroon_oscillator, 17
 - average_directional_movement_index, 20
 - average_directional_movement_index_rating, 22
 - average_price, 25

- average_true_range, 26
- balance_of_power, 28
- belt_hold, 30
- bollinger_bands, 33
- break_away, 36
- chaikin_accumulation_distribution_line, 40
- chaikin_accumulation_distribution_oscillator, 42
- chande_momentum_oscillator, 44
- closing_marubozu, 50
- commodity_channel_index, 53
- concealing_baby_swallow, 56
- counter_attack, 59
- dark_cloud_cover, 62
- directional_movement_index, 65
- doji, 67
- doji_star, 70
- dominant_cycle_period, 73
- dominant_cycle_phase, 75
- double_exponential_moving_average, 77
- dragonfly_doji, 79
- engulfing, 82
- evening_doji_star, 85
- evening_star, 88
- exponential_moving_average, 91
- extended_moving_average_convergence_divergence, 93
- extended_parabolic_stop_and_reverse, 95
- fast_stochastic, 98
- fixed_moving_average_convergence_divergence, 100
- gaps_side_white, 102
- gravestone_doji, 105
- hammer, 108
- hanging_man, 111
- harami, 114
- harami_cross, 117
- high_wave, 120
- hikakke, 123
- hikakke_mod, 126
- homing_pigeon, 129
- in_neck, 140
- intraday_movement_index, 135
- inverted_hammer, 137
- kaufman_adaptive_moving_average, 143
- kicking, 145
- kicking_baby_length, 148
- ladder_bottom, 151
- long_legged_doji, 154
- long_line, 157
- marubozu, 160
- mat_hold, 166
- matching_low, 163
- median_price, 169
- mesa_adaptive_moving_average, 170
- midpoint_price, 173
- minus_directional_indicator, 175
- minus_directional_movement, 177
- momentum, 179
- money_flow_index, 181
- morning_doji_star, 183
- morning_star, 186
- moving_average_convergence_divergence, 189
- normalized_average_true_range, 191
- on_balance_volume, 194
- on_neck, 196
- parabolic_stop_and_reverse, 199
- percentage_price_oscillator, 201
- phasor_components, 203
- piercing, 206
- plus_directional_indicator, 209
- plus_directional_movement, 211
- rate_of_change, 213
- ratio_of_change, 215
- relative_strength_index, 217
- rickshaw_man, 219
- rise_fall_3_methods, 222
- rolling_beta, 225
- rolling_correlation, 227
- rolling_max, 228
- rolling_min, 229
- rolling_standard_deviation, 231
- rolling_sum, 232
- rolling_variance, 233
- separating_lines, 234
- shooting_star, 239
- short_line, 242
- simple_moving_average, 245
- sine_wave, 247
- spinning_top, 249
- stalled_pattern, 253

- stick_sandwich, 256
- stochastic, 259
- stochastic_relative_strength_index, 261
- t3_exponential_moving_average, 264
- takuri, 266
- tasuki_gap, 269
- three_black_crows, 272
- three_identical_crows, 275
- three_inside, 278
- three_line_strike, 281
- three_outside, 284
- three_stars_in_the_south, 287
- three_white_soldiers, 290
- thrusting, 293
- trading_volume, 296
- trend_cycle_mode, 300
- trendline, 298
- triangular_moving_average, 302
- triple_exponential_average, 305
- triple_exponential_moving_average, 307
- tristar, 309
- true_range, 312
- two_crows, 314
- typical_price, 317
- ultimate_oscillator, 318
- unique_3_river, 321
- upside_gap_2_crows, 324
- weighted_close_price, 327
- weighted_moving_average, 328
- williams_oscillator, 330
- xside_gap_3_methods, 332
- * trading**
 - abandoned_baby, 5
 - absolute_price_oscillator, 8
 - acceleration_bands, 10
 - advance_block, 12
 - aroon, 15
 - aroon_oscillator, 17
 - average_directional_movement_index, 20
 - average_directional_movement_index_rating, 22
 - average_price, 25
 - average_true_range, 26
 - balance_of_power, 28
 - belt_hold, 30
 - bollinger_bands, 33
 - break_away, 36
 - chaikin_accumulation_distribution_line, 40
 - chaikin_accumulation_distribution_oscillator, 42
 - chande_momentum_oscillator, 44
 - closing_marubozu, 50
 - commodity_channel_index, 53
 - concealing_baby_swallow, 56
 - counter_attack, 59
 - dark_cloud_cover, 62
 - directional_movement_index, 65
 - doji, 67
 - doji_star, 70
 - dominant_cycle_period, 73
 - dominant_cycle_phase, 75
 - double_exponential_moving_average, 77
 - dragonfly_doji, 79
 - engulfing, 82
 - evening_doji_star, 85
 - evening_star, 88
 - exponential_moving_average, 91
 - extended_moving_average_convergence_divergence, 93
 - extended_parabolic_stop_and_reverse, 95
 - fast_stochastic, 98
 - fixed_moving_average_convergence_divergence, 100
 - gaps_side_white, 102
 - gravestone_doji, 105
 - hammer, 108
 - hanging_man, 111
 - harami, 114
 - harami_cross, 117
 - high_wave, 120
 - hikakke, 123
 - hikakke_mod, 126
 - homing_pigeon, 129
 - in_neck, 140
 - intraday_movement_index, 135
 - inverted_hammer, 137
 - kaufman_adaptive_moving_average, 143
 - kicking, 145
 - kicking_baby_length, 148

- ladder_bottom, 151
- long_legged_doji, 154
- long_line, 157
- marubozu, 160
- mat_hold, 166
- matching_low, 163
- median_price, 169
- mesa_adaptive_moving_average, 170
- midpoint_price, 173
- minus_directional_indicator, 175
- minus_directional_movement, 177
- momentum, 179
- money_flow_index, 181
- morning_doji_star, 183
- morning_star, 186
- moving_average_convergence_divergence, 189
- normalized_average_true_range, 191
- on_balance_volume, 194
- on_neck, 196
- parabolic_stop_and_reverse, 199
- percentage_price_oscillator, 201
- phasor_components, 203
- piercing, 206
- plus_directional_indicator, 209
- plus_directional_movement, 211
- rate_of_change, 213
- ratio_of_change, 215
- relative_strength_index, 217
- rickshaw_man, 219
- rise_fall_3_methods, 222
- rolling_beta, 225
- rolling_correlation, 227
- rolling_max, 228
- rolling_min, 229
- rolling_standard_deviation, 231
- rolling_sum, 232
- rolling_variance, 233
- separating_lines, 234
- shooting_star, 239
- short_line, 242
- simple_moving_average, 245
- sine_wave, 247
- spinning_top, 249
- stalled_pattern, 253
- stick_sandwich, 256
- stochastic, 259
- stochastic_relative_strength_index, 261
- t3_exponential_moving_average, 264
- takuri, 266
- tasuki_gap, 269
- three_black_crows, 272
- three_identical_crows, 275
- three_inside, 278
- three_line_strike, 281
- three_outside, 284
- three_stars_in_the_south, 287
- three_white_soldiers, 290
- thrusting, 293
- trading_volume, 296
- trend_cycle_mode, 300
- trendline, 298
- triangular_moving_average, 302
- triple_exponential_average, 305
- triple_exponential_moving_average, 307
- tristar, 309
- true_range, 312
- two_crows, 314
- typical_price, 317
- ultimate_oscillator, 318
- unique_3_river, 321
- upside_gap_2_crows, 324
- weighted_close_price, 327
- weighted_moving_average, 328
- williams_oscillator, 330
- xside_gap_3_methods, 332
- abandoned_baby, 5
- abandoned_baby(), 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- absolute_price_oscillator, 8
- absolute_price_oscillator(), 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- ACCBANDS (acceleration_bands), 10
- acceleration_bands, 10

- acceleration_bands(), [35](#), [78](#), [92](#), [97](#), [144](#),
[172](#), [200](#), [247](#), [265](#), [300](#), [304](#), [308](#),
[330](#)
- AD
 (chaikin_accumulation_distribution_line),
[40](#)
- ADOSC
 (chaikin_accumulation_distribution_oscillator),
[42](#)
- advance_block, [12](#)
- advance_block(), [7](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#),
[69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#),
[114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#),
[139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#),
[162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#),
[222](#), [225](#), [237](#), [241](#), [244](#), [251](#), [255](#),
[258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#),
[286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#),
[326](#), [335](#)
- ADX
 (average_directional_movement_index),
[20](#)
- ADXR
 (average_directional_movement_index_rating),
[22](#)
- APO (absolute_price_oscillator), [8](#)
- AROON (aroon), [15](#)
- aroon, [15](#)
- aroon(), [9](#), [19](#), [22](#), [24](#), [29](#), [45](#), [55](#), [66](#), [95](#), [99](#),
[102](#), [136](#), [176](#), [178](#), [180](#), [182](#), [191](#),
[203](#), [210](#), [212](#), [214](#), [216](#), [219](#), [260](#),
[263](#), [306](#), [320](#), [332](#)
- aroon_oscillator, [17](#)
- aroon_oscillator(), [9](#), [17](#), [22](#), [24](#), [29](#), [45](#),
[55](#), [66](#), [95](#), [99](#), [102](#), [136](#), [176](#), [178](#),
[180](#), [182](#), [191](#), [203](#), [210](#), [212](#), [214](#),
[216](#), [219](#), [260](#), [263](#), [306](#), [320](#), [332](#)
- AROONOSC (aroon_oscillator), [17](#)
- ATOM, [19](#)
- ATR (average_true_range), [26](#)
- average_directional_movement_index, [20](#)
- average_directional_movement_index(),
[9](#), [17](#), [19](#), [24](#), [29](#), [45](#), [55](#), [66](#), [95](#), [99](#),
[102](#), [136](#), [176](#), [178](#), [180](#), [182](#), [191](#),
[203](#), [210](#), [212](#), [214](#), [216](#), [219](#), [260](#),
[263](#), [306](#), [320](#), [332](#)
- average_directional_movement_index_rating,
[22](#)
- average_directional_movement_index_rating(),
[9](#), [17](#), [19](#), [22](#), [29](#), [45](#), [55](#), [66](#), [95](#), [99](#),
[102](#), [136](#), [176](#), [178](#), [180](#), [182](#), [191](#),
[203](#), [210](#), [212](#), [214](#), [216](#), [219](#), [260](#),
[263](#), [306](#), [320](#), [332](#)
- average_price, [25](#)
- average_price(), [170](#), [174](#), [318](#), [328](#)
- average_true_range, [26](#)
- average_true_range(), [193](#), [313](#)
- AVGPRICE (average_price), [25](#)
- balance_of_power, [28](#)
- balance_of_power(), [9](#), [17](#), [19](#), [22](#), [24](#), [45](#),
[55](#), [66](#), [95](#), [99](#), [102](#), [136](#), [176](#), [178](#),
[180](#), [182](#), [191](#), [203](#), [210](#), [212](#), [214](#),
[216](#), [219](#), [260](#), [263](#), [306](#), [320](#), [332](#)
- BBANDS (bollinger_bands), [33](#)
- belt_hold, [30](#)
- belt_hold(), [7](#), [14](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#),
[81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#),
[117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#),
[142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#),
[165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#),
[225](#), [237](#), [241](#), [244](#), [251](#), [255](#), [258](#),
[268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#),
[289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#),
[335](#)
- BETA (rolling_beta), [225](#)
- bollinger_bands, [33](#)
- bollinger_bands(), [12](#), [78](#), [92](#), [97](#), [144](#), [172](#),
[200](#), [247](#), [265](#), [300](#), [304](#), [308](#), [330](#)
- BOP (balance_of_power), [28](#)
- break_away, [36](#)
- break_away(), [7](#), [14](#), [32](#), [53](#), [58](#), [61](#), [64](#), [69](#),
[72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#),
[114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#),
[139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#),
[162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#),
[222](#), [225](#), [237](#), [241](#), [244](#), [251](#), [255](#),
[258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#),
[286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#),
[326](#), [335](#)
- BTC, [39](#)
- CCI (commodity_channel_index), [53](#)
- CDL2CROWS (two_crows), [314](#)
- CDL3BLACKCROWS (three_black_crows), [272](#)
- CDL3INSIDE (three_inside), [278](#)
- CDL3LINESTRIKE (three_line_strike), [281](#)

- CDL3OUTSIDE (three_outside), 284
- CDL3STARSINSOUTH
 - (three_stars_in_the_south), 287
- CDL3WHITESOLDIERS
 - (three_white_soldiers), 290
- CDLABANDONEDBABY (abandoned_baby), 5
- CDLADVANCEBLOCK (advance_block), 12
- CDLBELTHOLD (belt_hold), 30
- CDLBREAKAWAY (break_away), 36
- CDLCLOSINGMARUBOZU (closing_marubozu), 50
- CDLCONCEALBABYSWALL
 - (concealing_baby_swallow), 56
- CDLCOUNTERATTACK (counter_attack), 59
- CDLDARKCLOUDCOVER (dark_cloud_cover), 62
- CDLDOJI (doji), 67
- CDLDOJISTAR (doji_star), 70
- CDLDRAGONFLYDOJI (dragonfly_doji), 79
- CDLENGULFING (engulfing), 82
- CDLEVENINGDOJISTAR (evening_doji_star), 85
- CDLEVENINGSTAR (evening_star), 88
- CDLGAPSIDESIDEWHITE (gaps_side_white), 102
- CDLGRAVESTONEDOJI (gravestone_doji), 105
- CDLHAMMER (hammer), 108
- CDLHANGINGMAN (hanging_man), 111
- CDLHARAMI (harami), 114
- CDLHARAMICROSS (harami_cross), 117
- CDLHIGHWAVE (high_wave), 120
- CDLHIKKAKE (hikakke), 123
- CDLHIKKAKEMOD (hikakke_mod), 126
- CDLHOMINGPIGEON (homing_pigeon), 129
- CDLIDENTICAL3CROWS
 - (three_identical_crows), 275
- CDLINNECK (in_neck), 140
- CDLINVERTEDHAMMER (inverted_hammer), 137
- CDLKICKING (kicking), 145
- CDLKICKINGBYLENGTH
 - (kicking_baby_length), 148
- CDLLADDERBOTTOM (ladder_bottom), 151
- CDLLONGLEGGEDOJI (long_legged_doji), 154
- CDLLONGLINE (long_line), 157
- CDLMARUBOZU (marubozu), 160
- CDLMATCHINGLOW (matching_low), 163
- CDLMATHOLD (mat_hold), 166
- CDLMORNINGDOJISTAR (morning_doji_star), 183
- CDLMORNINGSTAR (morning_star), 186
- CDLONNECK (on_neck), 196
- CDLPIERCING (piercing), 206
- CDLRICKSHAWMAN (rickshaw_man), 219
- CDLRISEFALL3METHODS
 - (rise_fall_3_methods), 222
- CDLSEPARATINGLINES (separating_lines), 234
- CDLSHOOTINGSTAR (shooting_star), 239
- CDLSHORTLINE (short_line), 242
- CDLSPINNINGTOP (spinning_top), 249
- CDLSTALLEDPATTERN (stalled_pattern), 253
- CDLSTICKSANDWICH (stick_sandwich), 256
- CDLTAKURI (takuri), 266
- CDLTASUKIGAP (tasuki_gap), 269
- CDLTHRUSTING (thrusting), 293
- CDLTRISTAR (tristar), 309
- CDLUNIQUE3RIVER (unique_3_river), 321
- CDLUPSIDEGAP2CROWS
 - (upside_gap_2_crows), 324
- CDLXSIDEGAP3METHODS
 - (xside_gap_3_methods), 332
- chaikin_accumulation_distribution_line, 40
- chaikin_accumulation_distribution_line(), 43, 195, 298
- chaikin_accumulation_distribution_oscillator, 42
- chaikin_accumulation_distribution_oscillator(), 41, 195, 298
- chande_momentum_oscillator, 44
- chande_momentum_oscillator(), 9, 17, 19, 22, 24, 29, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- character, 46, 238
- chart, 46
- chart(), 50, 133, 134, 238, 239
- chart_themes, 48, 48, 134, 238, 239
- class, 5, 7–12, 14–18, 20–22, 24–30, 32, 33, 35–37, 40–45, 51, 52, 54–57, 59, 60, 62, 63, 65–67, 69, 70, 72–79, 81, 82, 84, 85, 87, 88, 90–95, 97–102, 104, 106, 107, 109, 110, 112, 113, 115, 116, 118, 119, 121, 122, 124, 125, 127, 128, 130, 131, 135–137, 139,

- 140, 142–145, 147, 148, 150, 151, 153, 154, 156, 157, 159, 160, 162, 163, 165, 166, 168–183, 185, 186, 188–192, 194–196, 198–202, 204–207, 209–219, 221, 223, 224, 226–229, 231–233, 235, 236, 239, 241, 242, 244–249, 251, 253, 255, 256, 258–261, 263–266, 268, 269, 271, 272, 274, 275, 277, 278, 280, 281, 283, 284, 286, 287, 289, 290, 292, 293, 295–302, 304–309, 311–314, 316–322, 324, 325, 327–332, 334
- `closing_marubozu`, 50
- `closing_marubozu()`, 7, 14, 32, 38, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- CMO (`chande_momentum_oscillator`), 44
- `commodity_channel_index`, 53
- `commodity_channel_index()`, 9, 17, 19, 22, 24, 29, 45, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- `concealing_baby_swallow`, 56
- `concealing_baby_swallow()`, 7, 14, 32, 38, 53, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- CORREL (`rolling_correlation`), 227
- `counter_attack`, 59
- `counter_attack()`, 7, 14, 32, 38, 53, 58, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- `dark_cloud_cover`, 62
- `dark_cloud_cover()`, 7, 14, 32, 38, 53, 58, 61, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- `data.frame`, 5, 8–13, 15–18, 20–23, 25–31, 33, 34, 36, 39, 40, 42, 44–46, 51, 54, 56, 59, 62, 65, 67, 70, 71, 73–80, 82, 83, 85, 86, 88, 89, 91–96, 98–103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 135, 137, 140, 141, 143–146, 148, 149, 151, 152, 154, 155, 157, 158, 160, 161, 163, 164, 166, 167, 169–171, 173–175, 177, 179–184, 186, 187, 189–192, 194–197, 199–202, 204, 206, 209, 211, 213–220, 223, 235, 239, 240, 242, 243, 245–250, 253, 254, 256, 257, 259–262, 264–267, 269, 270, 272, 273, 275, 276, 278, 279, 281, 282, 284, 285, 287, 288, 290, 291, 293, 294, 296–303, 305, 307–310, 312–315, 317–319, 321, 324, 327–333
- DEMA
(`double_exponential_moving_average`), 77
- `directional_movement_index`, 65
- `directional_movement_index()`, 9, 17, 19, 22, 24, 29, 45, 55, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- `doji`, 67
- `doji()`, 7, 14, 32, 38, 53, 58, 61, 64, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268,

- 271, 274, 277, 280, 283, 286, 289,
292, 295, 311, 316, 323, 326, 335
- doji_star, 70
- doji_star(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
81, 84, 87, 90, 105, 108, 111, 114,
117, 120, 123, 126, 129, 132, 139,
142, 147, 150, 153, 156, 159, 162,
165, 168, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- dominant_cycle_period, 73
- dominant_cycle_period(), 76, 205, 249,
302
- dominant_cycle_phase, 75
- dominant_cycle_phase(), 74, 205, 249, 302
- double, 6, 8, 9, 11, 13, 16, 18, 21, 24, 26, 27,
29, 31, 33–35, 37, 41, 43–45, 52, 55,
57, 60, 62, 63, 66, 68, 71, 73–78, 80,
83, 86, 89, 91–94, 96, 97, 99–101,
103, 107, 110, 113, 116, 119, 122,
125, 128, 131, 136, 138, 141, 143,
144, 146, 149, 152, 155, 158, 161,
164, 167, 170–172, 174, 176,
178–180, 182, 184, 187, 189, 190,
192, 195, 197, 200–202, 204, 205,
207, 210, 212–218, 220, 224,
226–234, 236, 240, 243, 245–248,
250, 254, 257, 260–265, 267, 270,
273, 276, 279, 282, 285, 288, 291,
294, 297–308, 310, 313, 315, 318,
320, 322, 325, 328, 329, 331, 333
- double_exponential_moving_average, 77,
78
- double_exponential_moving_average(),
12, 35, 92, 97, 144, 172, 200, 247,
265, 300, 304, 308, 330
- dragonfly_doji, 79
- dragonfly_doji(), 7, 14, 32, 38, 53, 58, 61,
64, 69, 72, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 198, 208,
222, 225, 237, 241, 244, 251, 255,
258, 268, 271, 274, 277, 280, 283,
286, 289, 292, 295, 311, 316, 323,
326, 335
- DX (directional_movement_index), 65
- EMA, 94
- EMA (exponential_moving_average), 91
- engulfing, 82
- engulfing(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 87, 90, 105, 108, 111, 114,
117, 120, 123, 126, 129, 132, 139,
142, 147, 150, 153, 156, 159, 162,
165, 168, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- evening_doji_star, 85
- evening_doji_star(), 7, 14, 32, 38, 53, 58,
61, 64, 69, 72, 81, 84, 90, 105, 108,
111, 114, 117, 120, 123, 126, 129,
132, 139, 142, 147, 150, 153, 156,
159, 162, 165, 168, 185, 188, 198,
208, 222, 225, 237, 241, 244, 251,
255, 258, 268, 271, 274, 277, 280,
283, 286, 289, 292, 295, 311, 316,
323, 326, 335
- evening_star, 88
- evening_star(), 7, 14, 32, 38, 53, 58, 61, 64,
69, 72, 81, 84, 87, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 198, 208,
222, 225, 237, 241, 244, 251, 255,
258, 268, 271, 274, 277, 280, 283,
286, 289, 292, 295, 311, 316, 323,
326, 335
- exponential_moving_average, 91, 92
- exponential_moving_average(), 12, 35, 78,
97, 144, 172, 200, 247, 265, 300,
304, 308, 330
- extended_moving_average_convergence_divergence,
93
- extended_moving_average_convergence_divergence(),
9, 17, 19, 22, 24, 29, 45, 55, 66, 99,
102, 136, 176, 178, 180, 182, 191,
203, 210, 212, 214, 216, 219, 260,
263, 306, 320, 332
- extended_parabolic_stop_and_reverse,
95
- extended_parabolic_stop_and_reverse(),
12, 35, 78, 92, 144, 172, 200, 247,

- 265, 300, 304, 308, 330
- FALSE, 6, 9, 11, 13, 16, 18, 21, 23, 25, 27, 29, 31, 34, 36, 40, 43, 45, 51, 54, 56, 59, 62, 65, 68, 71, 74, 76, 78, 80, 83, 86, 89, 92, 94, 97, 99, 101, 103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 135, 138, 141, 144, 146, 149, 152, 155, 158, 161, 164, 167, 170, 172, 174, 175, 178, 180, 182, 184, 187, 190, 192, 195, 197, 200, 202, 204, 206, 209, 212, 214, 216, 218, 220, 223, 226, 227, 229–232, 234, 235, 240, 243, 246, 248, 250, 254, 257, 260, 262, 265, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 299, 301, 303, 306, 308, 310, 313, 315, 318, 319, 321, 324, 327, 329, 331, 333
- fast_stochastic, 98
- fast_stochastic(), 9, 17, 19, 22, 24, 29, 45, 55, 66, 95, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- fixed_moving_average_convergence_divergence, 100
- fixed_moving_average_convergence_divergence(), 9, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- formula, 5, 9, 11, 13, 16, 18, 21, 23, 25, 27, 29, 31, 34, 36, 40, 43, 45, 51, 54, 56, 59, 62, 65, 67, 71, 74, 76, 78, 80, 83, 86, 89, 92, 94, 96, 99, 101, 103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 135, 137, 141, 144, 146, 149, 152, 155, 158, 161, 164, 167, 170, 171, 174, 175, 177, 180, 182, 184, 187, 190, 192, 195, 197, 200, 202, 204, 206, 209, 211, 214, 216, 218, 220, 223, 235, 240, 243, 246, 248, 250, 254, 257, 260, 262, 265, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 299, 301, 303, 305, 308, 310, 313, 315, 318, 319, 321, 324, 327, 329, 331, 333
- gaps_side_white, 102
- gaps_side_white(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- gravestone_doji, 105
- gravestone_doji(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- hammer, 108
- hammer(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- hanging_man, 111
- hanging_man(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- harami, 114
- harami(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- harami_cross, 117

- harami_cross(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- high_wave, 120
- high_wave(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- hikakke, 123
- hikakke(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- hikakke_mod, 126
- hikakke_mod(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- homing_pigeon, 129
- homing_pigeon(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- HT_DCPERIOD (dominant_cycle_period), 73
- HT_DCPHASE (dominant_cycle_phase), 75
- HT_PHASOR (phasor_components), 203
- HT_SINE (sine_wave), 247
- HT_TRENDLINE (trendline), 298
- HT_TRENDMODE (trend_cycle_mode), 300
- IMI (intraday_movement_index), 135
- in_neck, 140
- in_neck(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- indicator, 132
- indicator(), 46–48, 50, 238, 239
- integer, 6, 7, 9, 11, 13, 14, 16, 18, 21, 23, 27, 31, 32, 37, 43, 45, 52, 54, 57, 60, 63, 65, 68, 69, 71, 72, 78, 80, 81, 83, 84, 86, 87, 89, 90, 92, 99, 101, 103, 104, 107, 110, 113, 116, 119, 122, 125, 128, 131, 135, 138, 139, 141, 142, 144, 146, 147, 149, 150, 152, 153, 155, 156, 158, 159, 161, 162, 164, 165, 167, 168, 172, 174, 175, 177, 180, 182, 184, 185, 187, 188, 190, 192, 197, 198, 202, 207, 209, 211, 214, 216, 218, 220, 221, 224, 226–228, 230–232, 234, 236, 240, 241, 243, 244, 246, 250, 251, 254, 255, 257, 258, 260, 262, 265, 267, 268, 270, 271, 273, 274, 276, 277, 279, 280, 282, 283, 285, 286, 288, 289, 291, 292, 294, 295, 301, 303, 305, 308, 310, 311, 315, 316, 319, 322, 325, 329, 331, 333, 334
- intraday_movement_index, 135
- intraday_movement_index(), 9, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 260, 263, 306, 320, 332
- inverted_hammer, 137
- inverted_hammer(), 7, 14, 32, 38, 53, 58, 61, 326, 335

- 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- KAMA (kaufman_adaptive_moving_average), 143
- kaufman_adaptive_moving_average, 143, 144
- kaufman_adaptive_moving_average(), 12, 35, 78, 92, 97, 172, 200, 247, 265, 300, 304, 308, 330
- kicking, 145
- kicking(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- kicking_baby_length, 148
- kicking_baby_length(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- ladder_bottom, 151
- ladder_bottom(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- length, 6, 7, 9, 11, 13, 14, 16, 18, 21, 23–27, 29, 31, 32, 34–37, 40, 41, 43, 45, 46, 51, 52, 54–57, 59, 60, 62, 63, 65, 66, 68, 69, 71, 72, 74, 76, 78, 80, 81, 83, 84, 86, 87, 89, 90, 92, 94, 97, 99, 101, 103, 104, 106, 107, 109, 110, 112, 113, 115, 116, 118, 119, 121, 122, 124, 125, 127, 128, 130, 131, 135, 136, 138, 139, 141, 142, 144, 146, 147, 149, 150, 152, 153, 155, 156, 158, 159, 161, 162, 164, 165, 167, 168, 170, 172, 174–178, 180, 182, 184, 185, 187, 188, 190, 192, 195, 197, 198, 200, 202, 204–207, 209–212, 214, 216, 218, 220, 221, 223, 224, 226–236, 240, 241, 243, 244, 246, 248, 250, 251, 254, 255, 257, 258, 260, 262, 263, 265, 267, 268, 270, 271, 273, 274, 276, 277, 279, 280, 282, 283, 285, 286, 288, 289, 291, 292, 294, 295, 297, 299, 301, 303–306, 308, 310, 311, 313, 315, 316, 318–322, 324, 325, 327–329, 331, 333, 334
- lines(), 47
- list, 9, 34, 94, 99, 202, 260, 262
- logical, 6, 9, 11, 13, 16, 18, 21, 23, 25, 27, 29, 31, 34, 36, 40, 43, 45, 51, 54, 56, 59, 62, 65, 68, 71, 74, 76, 78, 80, 83, 86, 89, 92, 94, 97, 99, 101, 103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 135, 138, 141, 144, 146, 149, 152, 155, 158, 161, 164, 167, 170, 172, 174, 175, 178, 180, 182, 184, 187, 190, 192, 195, 197, 200, 202, 204, 206, 209, 212, 214, 216, 218, 220, 223, 226, 227, 229–232, 234, 235, 240, 243, 246, 248, 250, 254, 257, 260, 262, 265, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 299, 301, 303, 306, 308, 310, 313, 315, 318, 319, 321, 324, 327, 329, 331, 333
- long_legged_doji, 154
- long_legged_doji(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 251,

- 255, 258, 268, 271, 274, 277, 280,
283, 286, 289, 292, 295, 311, 316,
323, 326, 335
- long_line, 157
- long_line(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 162,
165, 168, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- MACD
(moving_average_convergence_divergence),
189
- MACDEXT
(extended_moving_average_convergence_divergence),
93
- MACDFIX
(fixed_moving_average_convergence_divergence),
100
- MAMA (mesa_adaptive_moving_average), 170
- marubozu, 160
- marubozu(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
165, 168, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- mat_hold, 166
- mat_hold(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- matching_low, 163
- matching_low(), 7, 14, 32, 38, 53, 58, 61, 64,
69, 72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 168, 185, 188, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- matrix, 5, 8, 10, 12, 15, 17, 20, 22, 25, 26, 28,
30, 33, 36, 40, 42, 44, 51, 54, 56, 59,
62, 65, 67, 70, 73, 75, 77, 79, 82, 85,
88, 91, 93, 95, 98, 100, 102, 106,
109, 112, 115, 118, 121, 124, 127,
130, 135, 137, 140, 143, 145, 148,
151, 154, 157, 160, 163, 166, 169,
171, 173, 175, 177, 179, 181, 183,
186, 189, 191, 194, 196, 199, 201,
204, 206, 209, 211, 213, 215, 217,
219, 223, 235, 239, 242, 245, 247,
249, 253, 256, 259, 261, 264, 266,
269, 272, 275, 278, 281, 284, 287,
290, 293, 296, 298, 300, 302, 305,
307, 309, 312, 314, 317, 319, 321,
324, 327, 328, 330, 332
- MAX (rolling_max), 228
- median_price, 169
- median_price(), 26, 174, 318, 328
- MEDPRICE (median_price), 169
- mesa_adaptive_moving_average, 170, 172
- mesa_adaptive_moving_average(), 12, 35,
78, 92, 97, 144, 200, 247, 265, 300,
304, 308, 330
- MFI (money_flow_index), 181
- midpoint_price, 173
- midpoint_price(), 26, 170, 318, 328
- MIDPRICE (midpoint_price), 173
- MIN (rolling_min), 229
- MINUS_DI (minus_directional_indicator),
175
- minus_directional_indicator, 175
- minus_directional_indicator(), 9, 17, 19,
22, 24, 29, 45, 55, 66, 95, 99, 102,
136, 178, 180, 182, 191, 203, 210,
212, 214, 216, 219, 260, 263, 306,
320, 332
- minus_directional_movement, 177
- minus_directional_movement(), 9, 17, 19,
22, 24, 29, 45, 55, 66, 95, 99, 102,
136, 176, 180, 182, 191, 203, 210,
212, 214, 216, 219, 260, 263, 306,
320, 332

- MINUS_DM (minus_directional_movement),
177
- model.frame, 5, 6, 9, 11, 13, 16, 18, 21, 23,
25, 27, 29, 31, 34, 36, 40, 41, 43, 45,
51, 54, 56, 57, 59, 62, 63, 65–68, 71,
74, 76, 78, 80, 83, 86, 89, 92, 94, 96,
97, 99, 101, 103, 106, 109, 112, 115,
118, 121, 124, 127, 130, 135–138,
141, 144, 146, 149, 152, 155, 158,
161, 164, 167, 170–172, 174–178,
180, 182, 184, 187, 190, 192, 195,
197, 200, 202, 204, 206, 209–212,
214, 216, 218, 220, 223, 235, 240,
243, 246, 248, 250, 254, 257, 260,
262, 263, 265, 267, 270, 273, 276,
279, 282, 285, 288, 291, 294, 297,
299, 301, 303, 305, 306, 308, 310,
313, 315, 318, 319, 321, 324, 327,
329, 331, 333
- MOM (momentum), 179
- momentum, 179
- momentum(), 9, 17, 19, 22, 24, 29, 45, 55, 66,
95, 99, 102, 136, 176, 178, 182, 191,
203, 210, 212, 214, 216, 219, 261,
263, 306, 320, 332
- money_flow_index, 181
- money_flow_index(), 9, 17, 19, 22, 24, 29,
45, 55, 66, 95, 99, 102, 136, 176,
178, 180, 191, 203, 210, 212, 214,
216, 219, 261, 263, 306, 320, 332
- morning_doji_star, 183
- morning_doji_star(), 7, 14, 32, 38, 53, 58,
61, 64, 69, 72, 81, 84, 87, 90, 105,
108, 111, 114, 117, 120, 123, 126,
129, 132, 139, 142, 147, 150, 153,
156, 159, 162, 165, 168, 188, 198,
208, 222, 225, 237, 241, 244, 251,
255, 258, 268, 271, 274, 277, 280,
283, 286, 289, 292, 295, 311, 316,
323, 326, 335
- morning_star, 186
- morning_star(), 7, 14, 32, 38, 53, 58, 61, 64,
69, 72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 198, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- moving_average_convergence_divergence,
189
- moving_average_convergence_divergence(),
9, 17, 19, 22, 24, 29, 45, 55, 66, 95,
99, 102, 136, 176, 178, 180, 182,
203, 210, 212, 214, 216, 219, 261,
263, 306, 320, 332
- NATR (normalized_average_true_range),
191
- normalized_average_true_range, 191
- normalized_average_true_range(), 27,
313
- NVDA, 193
- OBV (on_balance_volume), 194
- on_balance_volume, 194
- on_balance_volume(), 41, 43, 298
- on_neck, 196
- on_neck(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 208, 222,
225, 237, 241, 244, 251, 255, 258,
268, 271, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- options(), 47
- parabolic_stop_and_reverse, 199
- parabolic_stop_and_reverse(), 12, 35, 78,
92, 97, 144, 172, 247, 265, 300, 304,
308, 330
- percentage_price_oscillator, 201
- percentage_price_oscillator(), 9, 17, 19,
22, 24, 29, 45, 55, 66, 95, 99, 102,
136, 176, 178, 180, 182, 191, 210,
212, 214, 216, 219, 261, 263, 306,
320, 332
- phasor_components, 203
- phasor_components(), 74, 76, 249, 302
- piercing, 206
- piercing(), 7, 14, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,

- 162, 165, 168, 185, 188, 198, 222, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- plot(), 47
- plotly::plot_ly(), 46
- PLUS_DI (plus_directional_indicator), 209
- plus_directional_indicator, 209
- plus_directional_indicator(), 9, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 212, 214, 216, 219, 261, 263, 306, 320, 332
- plus_directional_movement, 211
- plus_directional_movement(), 10, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 214, 216, 219, 261, 263, 306, 320, 332
- PLUS_DM (plus_directional_movement), 211
- PPO (percentage_price_oscillator), 201
- rate_of_change, 213
- rate_of_change(), 10, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 216, 219, 261, 263, 306, 320, 332
- ratio_of_change, 215
- ratio_of_change(), 10, 17, 19, 22, 24, 29, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 219, 261, 263, 306, 320, 332
- relative_strength_index, 217
- relative_strength_index(), 10, 17, 19, 22, 24, 30, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 261, 263, 306, 320, 332
- rickshaw_man, 219
- rickshaw_man(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 225, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- rise_fall_3_methods, 222
- rise_fall_3_methods(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 237, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- ROC (rate_of_change), 213
- ROCR (ratio_of_change), 215
- rolling_beta, 225
- rolling_beta(), 228–231, 233, 234
- rolling_correlation, 227
- rolling_correlation(), 226, 229–231, 233, 234
- rolling_max, 228
- rolling_max(), 226, 228, 230, 231, 233, 234
- rolling_min, 229
- rolling_min(), 226, 228, 229, 231, 233, 234
- rolling_standard_deviation, 231
- rolling_standard_deviation(), 226, 228–230, 233, 234
- rolling_sum, 232
- rolling_sum(), 226, 228–231, 234
- rolling_variance, 233
- rolling_variance(), 226, 228–231, 233
- RSI (relative_strength_index), 217
- SAR (parabolic_stop_and_reverse), 199
- SAREXT (extended_parabolic_stop_and_reverse), 95
- separating_lines, 234
- separating_lines(), 7, 14, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 241, 244, 251, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 326, 335
- set_theme, 237
- set_theme(), 47, 48, 50, 134
- shooting_star, 239

- shooting_star(), [7](#), [14](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [244](#), [251](#), [255](#), [258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- short_line, [242](#)
- short_line(), [7](#), [14](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [241](#), [252](#), [255](#), [258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- simple_moving_average, [245](#), [246](#)
- simple_moving_average(), [12](#), [35](#), [78](#), [92](#), [97](#), [144](#), [172](#), [200](#), [265](#), [300](#), [304](#), [308](#), [330](#)
- sine_wave, [247](#)
- sine_wave(), [74](#), [76](#), [205](#), [302](#)
- SMA, [9](#), [34](#), [99](#), [202](#), [260](#), [262](#)
- SMA (simple_moving_average), [245](#)
- spinning_top, [249](#)
- spinning_top(), [7](#), [14](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [241](#), [244](#), [255](#), [258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- SPY, [252](#)
- stalled_pattern, [253](#)
- stalled_pattern(), [7](#), [14](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [241](#), [244](#), [252](#), [255](#), [258](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- STDDEV (rolling_standard_deviation), [231](#)
- stick_sandwich, [256](#)
- stick_sandwich(), [7](#), [15](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [241](#), [244](#), [252](#), [255](#), [268](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- STOCH (stochastic), [259](#)
- stochastic, [78](#), [92](#), [144](#), [172](#), [246](#), [259](#), [265](#), [303](#), [308](#), [329](#)
- stochastic(), [10](#), [17](#), [19](#), [22](#), [24](#), [30](#), [45](#), [55](#), [66](#), [95](#), [99](#), [102](#), [136](#), [176](#), [178](#), [180](#), [182](#), [191](#), [203](#), [210](#), [212](#), [214](#), [216](#), [219](#), [263](#), [306](#), [320](#), [332](#)
- stochastic_relative_strength_index, [261](#)
- stochastic_relative_strength_index(), [10](#), [17](#), [19](#), [22](#), [24](#), [30](#), [45](#), [55](#), [66](#), [95](#), [99](#), [102](#), [136](#), [176](#), [178](#), [180](#), [182](#), [191](#), [203](#), [210](#), [212](#), [214](#), [216](#), [219](#), [261](#), [306](#), [320](#), [332](#)
- STOCHF (fast_stochastic), [98](#)
- STOCHRSI (stochastic_relative_strength_index), [261](#)
- SUM (rolling_sum), [232](#)
- T3 (t3_exponential_moving_average), [264](#)
- t3_exponential_moving_average, [264](#), [265](#)
- t3_exponential_moving_average(), [12](#), [35](#), [78](#), [92](#), [97](#), [144](#), [172](#), [200](#), [247](#), [300](#), [304](#), [308](#), [330](#)
- takuri, [266](#)
- takuri(), [7](#), [15](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [139](#), [142](#), [147](#), [150](#), [153](#), [156](#), [159](#), [162](#), [165](#), [168](#), [185](#), [188](#), [198](#), [208](#), [222](#), [225](#), [237](#), [241](#), [244](#), [252](#), [255](#), [258](#), [271](#), [274](#), [277](#), [280](#), [283](#), [286](#), [289](#), [292](#), [295](#), [311](#), [316](#), [323](#), [326](#), [335](#)
- tasuki_gap, [269](#)
- tasuki_gap(), [7](#), [15](#), [32](#), [38](#), [53](#), [58](#), [61](#), [64](#), [69](#), [72](#), [81](#), [84](#), [87](#), [90](#), [105](#), [108](#), [111](#), [114](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#),

- 139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 198, 208,
222, 225, 237, 241, 244, 252, 255,
258, 268, 274, 277, 280, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- TEMA
(triple_exponential_moving_average),
307
- three_black_crows, 272
- three_black_crows(), 7, 15, 32, 38, 53, 58,
61, 64, 69, 72, 81, 84, 87, 90, 105,
108, 111, 114, 117, 120, 123, 126,
129, 132, 139, 142, 147, 150, 153,
156, 159, 162, 165, 168, 185, 188,
198, 208, 222, 225, 237, 241, 244,
252, 255, 258, 268, 271, 277, 280,
283, 286, 289, 292, 295, 311, 316,
323, 326, 335
- three_identical_crows, 275
- three_identical_crows(), 7, 15, 32, 38, 53,
58, 61, 64, 69, 72, 81, 84, 87, 90,
105, 108, 111, 114, 117, 120, 123,
126, 129, 132, 139, 142, 147, 150,
153, 156, 159, 162, 165, 168, 185,
188, 198, 208, 222, 225, 237, 241,
244, 252, 255, 258, 268, 271, 274,
280, 283, 286, 289, 292, 295, 311,
316, 323, 326, 335
- three_inside, 278
- three_inside(), 7, 15, 32, 38, 53, 58, 61, 64,
69, 72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 198, 208,
222, 225, 237, 241, 244, 252, 255,
258, 268, 271, 274, 277, 283, 286,
289, 292, 295, 311, 316, 323, 326,
335
- three_line_strike, 281
- three_line_strike(), 7, 15, 32, 38, 53, 58,
61, 64, 69, 72, 81, 84, 87, 90, 105,
108, 111, 114, 117, 120, 123, 126,
129, 132, 139, 142, 147, 150, 153,
156, 159, 162, 165, 168, 185, 188,
198, 208, 222, 225, 237, 241, 244,
252, 255, 258, 268, 271, 274, 277,
280, 286, 289, 292, 295, 311, 316,
323, 326, 335
- three_outside, 284
- three_outside(), 7, 15, 32, 38, 53, 58, 61,
64, 69, 72, 81, 84, 87, 90, 105, 108,
111, 114, 117, 120, 123, 126, 129,
132, 139, 142, 147, 150, 153, 156,
159, 162, 165, 168, 185, 188, 198,
208, 222, 225, 237, 241, 244, 252,
255, 258, 268, 271, 274, 277, 280,
283, 289, 292, 295, 311, 316, 323,
326, 335
- three_stars_in_the_south, 287
- three_stars_in_the_south(), 7, 15, 32, 38,
53, 58, 61, 64, 69, 72, 81, 84, 87, 90,
105, 108, 111, 114, 117, 120, 123,
126, 129, 132, 139, 142, 147, 150,
153, 156, 159, 162, 165, 168, 185,
188, 198, 208, 222, 225, 237, 241,
244, 252, 255, 258, 268, 271, 274,
277, 280, 283, 286, 292, 295, 311,
316, 323, 326, 335
- three_white_soldiers, 290
- three_white_soldiers(), 7, 15, 32, 38, 53,
58, 61, 64, 69, 72, 81, 84, 87, 90,
105, 108, 111, 114, 117, 120, 123,
126, 129, 132, 139, 142, 147, 150,
153, 156, 159, 162, 165, 168, 185,
188, 198, 208, 222, 225, 237, 241,
244, 252, 255, 258, 268, 271, 274,
277, 280, 283, 286, 289, 295, 311,
316, 323, 326, 335
- thrusting, 293
- thrusting(), 7, 15, 32, 38, 53, 58, 61, 64, 69,
72, 81, 84, 87, 90, 105, 108, 111,
114, 117, 120, 123, 126, 129, 132,
139, 142, 147, 150, 153, 156, 159,
162, 165, 168, 185, 188, 198, 208,
222, 225, 237, 241, 244, 252, 255,
258, 268, 271, 274, 277, 280, 283,
286, 289, 292, 311, 316, 323, 326,
335
- trading_volume, 296
- trading_volume(), 41, 43, 195
- TRANGE (true_range), 312
- trend_cycle_mode, 300
- trend_cycle_mode(), 74, 76, 205, 249
- trendline, 298
- trendline(), 12, 35, 78, 92, 97, 144, 172,

- [200, 247, 265, 304, 308, 330](#)
- [triangular_moving_average, 302, 303](#)
- [triangular_moving_average\(\), 12, 35, 78, 92, 97, 144, 172, 200, 247, 265, 300, 308, 330](#)
- [TRIMA \(triangular_moving_average\), 302](#)
- [triple_exponential_average, 305](#)
- [triple_exponential_average\(\), 10, 17, 19, 22, 24, 30, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 261, 263, 320, 332](#)
- [triple_exponential_moving_average, 307, 308](#)
- [triple_exponential_moving_average\(\), 12, 35, 78, 92, 97, 144, 172, 200, 247, 265, 300, 304, 330](#)
- [tristar, 309](#)
- [tristar\(\), 7, 15, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 252, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 316, 323, 326, 335](#)
- [TRIX \(triple_exponential_average\), 305](#)
- [TRUE, 6, 9, 11, 13, 16, 18, 21, 23, 25, 27, 29, 31, 34, 36, 40, 43, 45, 51, 54, 56, 59, 62, 65, 68, 71, 74, 76, 78, 80, 83, 86, 89, 92, 94, 97, 99, 101, 103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 135, 138, 141, 144, 146, 149, 152, 155, 158, 161, 164, 167, 170, 172, 174, 175, 178, 180, 182, 184, 187, 190, 192, 195, 197, 200, 202, 204, 206, 209, 212, 214, 216, 218, 220, 223, 226, 227, 229–232, 234, 235, 240, 243, 246, 248, 250, 254, 257, 260, 262, 265, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 299, 301, 303, 306, 308, 310, 313, 315, 318, 319, 321, 324, 327, 329, 331, 333](#)
- [true_range, 312](#)
- [true_range\(\), 27, 193](#)
- [two_crows, 314](#)
- [two_crows\(\), 7, 15, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 252, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 323, 326, 335](#)
- [typical_price, 317](#)
- [typical_price\(\), 26, 170, 174, 328](#)
- [TYPPRICE \(typical_price\), 317](#)
- [ultimate_oscillator, 318](#)
- [ultimate_oscillator\(\), 10, 17, 19, 22, 24, 30, 45, 55, 66, 95, 99, 102, 136, 176, 178, 180, 182, 191, 203, 210, 212, 214, 216, 219, 261, 263, 306, 332](#)
- [ULTOSC \(ultimate_oscillator\), 318](#)
- [unique_3_river, 321](#)
- [unique_3_river\(\), 7, 15, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 252, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 326, 335](#)
- [upside_gap_2_crows, 324](#)
- [upside_gap_2_crows\(\), 7, 15, 32, 38, 53, 58, 61, 64, 69, 72, 81, 84, 87, 90, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 139, 142, 147, 150, 153, 156, 159, 162, 165, 168, 185, 188, 198, 208, 222, 225, 237, 241, 244, 252, 255, 258, 268, 271, 274, 277, 280, 283, 286, 289, 292, 295, 311, 316, 323, 335](#)
- [VAR \(rolling_variance\), 233](#)
- [vector, 46](#)
- [VOLUME \(trading_volume\), 296](#)
- [WCLPRICE \(weighted_close_price\), 327](#)
- [weighted_close_price, 327](#)
- [weighted_close_price\(\), 26, 170, 174, 318](#)
- [weighted_moving_average, 328, 329](#)

`weighted_moving_average()`, 12, 35, 78, 92,
97, 144, 172, 200, 247, 265, 300,
304, 308

`williams_oscillator`, 330

`williams_oscillator()`, 10, 17, 19, 22, 24,
30, 45, 55, 66, 95, 99, 102, 136, 176,
178, 180, 182, 191, 203, 210, 212,
214, 216, 219, 261, 263, 306, 320

WILLR (`williams_oscillator`), 330

WMA (`weighted_moving_average`), 328

`xside_gap_3_methods`, 332

`xside_gap_3_methods()`, 7, 15, 32, 38, 53,
58, 61, 64, 69, 72, 81, 84, 87, 90,
105, 108, 111, 114, 117, 120, 123,
126, 129, 132, 139, 142, 147, 150,
153, 156, 159, 162, 165, 168, 185,
188, 198, 208, 222, 225, 237, 241,
244, 252, 255, 258, 268, 271, 274,
277, 280, 283, 286, 289, 292, 295,
311, 316, 323, 326