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 *Abandoned Baby*

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

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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
fast	(integer). Period for the fast Moving Average (MA).
slow	(integer). Period for the slow Moving Average (MA).
ma	(list). The type of Moving Average (MA) used for the fast and slow MA. 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:

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_divergenc 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** `NA`s for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
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:

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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
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:

AroonDown double

AroonUp double

Author(s)

Serkan Korkmaz

See Also

Other Momentum Indicator: absolute_price_oscillator(), aroon_oscillator(), average_directional_movement_i
average_directional_movement_index_rating(), balance_of_power(), chande_momentum_oscillator(),
commodity_channel_index(), directional_movement_index(), extended_moving_average_convergence_divergenc
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

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.
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:

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_divergenc 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
 )
}
```

ATOM *Cosmos (ATOM)*

Description

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

Usage

```
ATOM
```

Format

A data.frame with 366 rows and 5 columns.

open Opening price for the trading day.

high Highest price reached during the trading day.

low Lowest price reached during the trading day.

close Closing price for the trading day.

volume Total trading volume for the day.

Source

Loaded using cryptoQuotes.

Examples

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

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

`average_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** `NA`s for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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`:

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_divergen 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

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.
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:

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_divergenc 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` *Average Price*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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:

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

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`.
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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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_ 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_avera
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`ma`	(list). The type of Moving Average (MA) used for the `MiddleBand`. SMA by default.
`sd`	(double). Deviation multiplier for the upper and lower band.
`sd_down`	(double). Optional. Deviation multiplier for lower band
`sd_up`	(double). Optional. Deviation multiplier for upper band
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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:

UpperBand double

MiddleBand double

LowerBand double

Author(s)

Serkan Korkmaz

See Also

Other Overlap Study: acceleration_bands(), double_exponential_moving_average(), exponential_moving_averag
extended_parabolic_stop_and_reverse(), kaufman_adaptive_moving_average(), mesa_adaptive_moving_averag
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

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 "<=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 ">= 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 "<= 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 *Bitcoin (BTC)*

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

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

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

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

Usage

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

Arguments

`x`	An OHLC-V series coercible to data.frame.
`cols`	(formula). An optional 4-variable formula selecting columns from `x` via model.frame. Defaults to `~high + low + close + volume`.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~high + low + close + volume.
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 ~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:

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_co 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

`x`	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.
`type`	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.
`idx`	An optional vector with the same length as the number of rows in `x`. Replaces the default x-axis labels (row names or integer index). Useful for custom date formatting or non-standard index types.
`title`	An optional character string for the chart title. If omitted, the title is inferred from the variable name passed to `x`.
`...`	Additional parameters passed to the backend chart constructor (e.g., `plotly::plot_ly()`).

Details

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

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

Two rendering backends are supported:

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

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

Options:

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

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

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

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

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

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

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

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

State and concurrency:

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

The practical consequences:

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

Value

A chart object whose class depends on the active backend:

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

When called without arguments, returns `NULL` invisibly.

Author(s)

Serkan Korkmaz

See Also

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

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

Examples

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

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

## OHLC bar chart
talib::chart(BTC, type = "ohlc")

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

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

---

chart_themes *Chart Themes*

Description

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

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

Details

Available Themes:

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

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

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

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

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

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

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

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

Colorblind-Friendly Themes:

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

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

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

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

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

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

Theme Properties:

Each theme sets the following color properties:

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

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

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

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

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

References

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

See Also

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

Examples

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

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

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

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

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

`closing_marubozu` *Closing Marubozu*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLCLOSINGMARUBOZU integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

`commodity_channel_index`

Commodity Channel Index

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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

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 "<=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 ">= 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 "<= 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

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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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:

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_co 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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.

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_averag 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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.

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_averag 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
fast	(list). Period and Moving Average (MA) type for the fast MA. EMA by default.
slow	(list). Period and Moving Average (MA) type for the slow MA. EMA by default.
signal	(list). Period and Moving Average (MA) type for the signal MA. EMA 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:

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

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.
init	(double). Start value and direction. 0 for Auto, >0 for Long, <0 for Short.
offset	(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 model.frame.

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~high + low + close.
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
signal	(integer). Period for the signal 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:

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

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 "<=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 ">= 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 "<= 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
 )
}
```

---

gravestone_doji *Gravestone Doji*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLGRAVESTONEDOJI integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

hammer *Hammer*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLHAMMER integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

hanging_man *Hanging Man*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLHANGINGMAN integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

harami *Harami*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLHARAMI integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

harami_cross *Harami Cross*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLHARAMICROSS integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

high_wave *High Wave*

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 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 "<=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 ">= 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 "<= 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
 )
}
```

---

hikakke *Hikkake*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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
 )
}
```

---

hikakke_mod *Hikkake Modified*

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

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

`na.bridge = TRUE` `NA` rows are stripped from the input before the C routine runs; the indicator is computed on the resulting dense series; results are then re-expanded to the original length with `NA` inserted at every position the input had `NA`. Output length always matches input length, so the result can be joined back to the source `data.frame` by row.
Consequence to understand before enabling: bridging causes the indicator to treat non-consecutive observations as consecutive. A 14-period RSI with `na.bridge = TRUE` over a series containing a month-long gap will compute using 14 observations that span several real-world months as if they were 14 adjacent trading days. For sparse missing values (e.g. a single missing tick) this is harmless; for clustered gaps (e.g. a delisted period, a weekend encoded as `NA`) the output is correctly aligned *by position* but economically meaningless across the gap. Inspect gap structure with `which(is.na(x))` before enabling on low-quality time series.

Usage

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

Arguments

`x`	An OHLC-V series coercible to data.frame.
`cols`	(formula). An optional 4-variable formula selecting columns from `x` via model.frame. Defaults to `~open + high + low + close`.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLHIKKAKEMOD integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

homing_pigeon *Homing Pigeon*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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
 )
}
```

---

indicator *Add Technical Indicators to a Chart*

Description

`indicator()` attaches one or more technical indicators to an existing `chart()`, or renders an indicator as a standalone chart. Each indicator is displayed in its own subchart panel below the main price chart.

If `chart()` has not been called beforehand, `indicator()` creates a standalone indicator chart — in this case, `data` must be provided explicitly.

See `vignette(topic = "charting", package = "talib")` for a comprehensive guide.

Usage

```
indicator(FUN, ...)
```

Arguments

`FUN`	An indicator function or an indicator call. In single indicator mode, pass the bare function (e.g., `RSI`); arguments for the indicator go in `...`. In multi-indicator mode, pass a call with parentheses (e.g., `RSI(n = 14)`); additional indicator calls go in `...`.
`...`	In single indicator mode: arguments passed to `FUN` (e.g., `n = 14`, `data = BTC`). In multi-indicator mode: additional indicator calls to merge onto the same panel (e.g., `RSI(n = 21)`, `MACD()`).

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

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 ~open + 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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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.

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~high + low.
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 model.frame.

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~high + low.
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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_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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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:

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 ~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:

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_average(), ultimate_oscillator(), williams_oscillator()

Examples

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

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

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

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

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

`money_flow_index` *Money Flow Index*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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_average(), 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
fast	(integer). Period for the fast Moving Average (MA).
slow	(integer). Period for the slow Moving Average (MA).
signal	(integer). Period for the signal 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:

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

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.
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:

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 *On-Balance Volume*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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

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_oscil`
`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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`fast`	(integer). Period for the fast Moving Average (MA).
`slow`	(integer). Period for the slow Moving Average (MA).
`ma`	(list). The type of Moving Average (MA) used for the `fast` and `slow` MA. SMA by default.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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`:

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
 )
}
```

---

phasor_components *Hilbert Transform - Phasor Components*

Description

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

`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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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:

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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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_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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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:

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` *Rate of Change*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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:

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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:

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(), 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** `NA`**s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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:

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

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 "<=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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLRICKSHAWMAN integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

`rise_fall_3_methods` *Rising/Falling Three Methods*

Description

`rise_fall_3_methods()` is a generic S3 function that preserves the input class: data.frame in, data.frame out; matrix in, matrix out.

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLRISEFALL3METHODS integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

## calculate the indicator
## for Bitcoin (BTC)
output <- talib::rise_fall_3_methods(BTC)

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

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

 ## chart indicator
 ## with default values
 talib::indicator(
     talib::rise_fall_3_methods
 )
}
```

---

rolling_beta *Rolling Beta*

Description

`rolling_beta()` is a generic S3 function that preserves the input class: double vector in, double vector out.

Handling of `NA` values:

Leading `NA`s are always produced for the initial lookback period where insufficient data is available. If the input itself contains `NA`s, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` **(default)** `NA`s 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 `NA`s are stripped, the statistic is computed on the dense series, and `NA`s 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

`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 `NA`s 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_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 `NA`s are always produced for the initial lookback period where insufficient data is available. If the input itself contains `NA`s, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` **(default)** `NA`s 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 `NA`s are stripped, the statistic is computed on the dense series, and `NA`s 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 `NA`s 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 `NA`s are always produced for the initial lookback period where insufficient data is available. If the input itself contains `NA`s, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` **(default)** `NA`s 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 `NA`s are stripped, the statistic is computed on the dense series, and `NA`s 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

`x`	(double). A double vector.
`n`	(integer). Lookback period (window size). A positive integer of length 1.
`k`	(double). Multiplier for the standard deviation.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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_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 `NA`s are always produced for the initial lookback period where insufficient data is available. If the input itself contains `NA`s, the behaviour depends on `na.bridge`:

`na.bridge = FALSE` **(default)** `NA`s 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 `NA`s are stripped, the statistic is computed on the dense series, and `NA`s 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

`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 `NA`s 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_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 *Rolling Standard Deviation*

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)
```

---

separating_lines *Separating Lines*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 5% of the average of the 5 previous candles high-low range."

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

Value

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

CDLSEPARATINGLINES integer

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

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

Author(s)

Serkan Korkmaz

See Also

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

Examples

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

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

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

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

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

---

set_theme *Set or List Chart Themes*

Description

Apply a named color theme to the charting system, override individual theme properties, or list all available theme names.

Theme changes take effect immediately and apply to all subsequent `chart()` and `indicator()` calls.

Usage

```
set_theme(name, ...)
```

Arguments

`name`	An optional character string matching one of the available theme names. Partial matching is supported. If omitted (and no `...` overrides are given), returns the available theme names instead.
`...`	Named color overrides applied after the base theme. Valid names include any theme property: `bearish_body`, `bearish_wick`, `bearish_border`, `bullish_body`, `bullish_wick`, `bullish_border`, `background_color`, `foreground_color`, `text_color`, `gridcolor`, `threshold_color`, and `colorway` (a character vector of up to 10 colors).

Details

`set_theme` supports three usage patterns:

`set_theme()` Returns a character vector of available theme names.

`set_theme("payout")` Applies the named theme.

`set_theme$payout` Applies the named theme via `$` syntax (supports tab-completion in interactive sessions).

Themes can be combined with individual color overrides. When both `name` and `...` are provided, the base theme is applied first, then the overrides are applied on top:

```
# Apply "payout" but with a custom background
set_theme("payout", background_color = "#000000")
```

It is also possible to override individual properties without selecting a theme:

```
# Change only the bearish candle color
set_theme(bearish_body = "#FF0000")
```

See chart_themes for a full description of all available themes and their color properties.

Value

When called without arguments, a character vector of available theme names. Otherwise, invisibly returns `NULL`; the theme is applied as a side effect to the internal chart-variables state.

See Also

chart_themes for descriptions of each theme, `chart()` for creating charts.

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

Examples

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

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

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

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

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

---

shooting_star *Shooting Star*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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** `NA`**s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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', 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

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.
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:

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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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.
fastk	(integer). Period for the fast-k line.
slowk	(list). Period and Moving Average (MA) type for the slow-k line. SMA by default.
slowd	(list). Period and Moving Average (MA) type for the slow-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:

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

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.
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(), 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~close.
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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` *Three-Line Strike*

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `~volume + open + close`.
`ma`	A list of MA specifications.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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`:

VOLUME double

SMA7 double

SMA15 double

Author(s)

Serkan Korkmaz

See Also

Other Volume Indicator: chaikin_accumulation_distribution_line(), chaikin_accumulation_distribution_oscil
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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`na.bridge`	(logical). A logical of length 1. FALSE by default. When FALSE, input `NA`s 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_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

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.
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_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

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.

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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.

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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

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 "<=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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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 ~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:

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)
```

---

ultimate_oscillator *Ultimate Oscillator*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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:

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_average(), williams_oscillator()

Examples

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

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

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

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

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

`unique_3_river` *Unique Three River*

Description

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

Handling of `NA` values:

Every indicator always emits **leading `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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 ">= 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 "<= 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 `NA`s** for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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 `NA`s 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:

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

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.

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** `NA`s for the initial lookback period - positions where there is not yet enough data to produce a result. This is separate from how `NA`s 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

`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`.
`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 `NA`s 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`:

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

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 "<=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 ">= 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 "<= 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