qufilab

Anton Normelius

Sep 07, 2020

CONTENTS

CHAPTER

ONE

ABOUT

Something about the project.

CHAPTER

TWO

INSTALLATION

The easiest way to install qufilab is to use a package manager like pip

```
$ pip3 install qufilab
```

Alternatively, one can clone the package and install from source

```
$ git clone https://github.com/normelius/qufilab.git
$ python setup.py install
```

CHAPTER

THREE

INDICATORS

Warning: All of qufilab's technical indicators are implemented in c++ and a big part of the speed performance comes from the fact that no type conversion exist between python and c++. In order for this to work, numpy arrays of type `numpy.dtype.float64` or `numpy.dtype.float32` are preferably used. Observe that all other types of numpy arrays are accepted, however the returned numpy array will be converted into the type `numpy.dtype.float64`.

3.1 Trend

3.1.1 Double Exponential Moving Average

`qufilab.`**`dema`**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated double exponential moving average values.

Notes

$$dema_K = 2 \cdot ema_K - ema(ema_K)$$

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> dema = ql.ema(df['close'], periods = 10)
>>> print(dema)
[nan nan nan ... 210.18599259 211.72078484 212.32702478]
```

3.1.2 Exponential Moving Average

qufilab.**ema**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated exponential moving average values.

Notes

The calculation of ema can be written as a power series:

$$ema_K = \alpha[price_K + (1-\alpha)price_{K-1} + \ (1-\alpha)^2 price_{K-2} + ... + (1-\alpha)^{K-(n-1)} price_{K-(n-1)},$$

which also can be written as:

$$ema_K = ema_{K-1} + \alpha[price_K - ema_{K-1}],$$

where $\alpha = \frac{2}{n+1}$ is the multiplier and depends on the period n. Observe that for the first ema value, a simple moving average is used.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> ema = ql.ema(df['close'], periods = 10)
>>> print(ema)
[nan nan nan ... 209.63323895 210.53265005 210.98489549]
```

3.1.3 Linear Weighted Moving Average

qufilab.**lwma**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated linear weighted moving average values.

Notes

The linear weighted moving average is calculated similar to the simple moving average, except that the values aren't equally weighted, but linearly weighted with the highest weight going first and then decreasing linearly. This implementation use the number of periods as the highest weight, and thereafter decreasing down to one.

$$lwma_K = \frac{price_K \cdot w_n + price_{K-1} \cdot w_{n-1} \ + ... + price_{K-(n-1)} \cdot w_1}{\sum_{i=1}^{n} w_i},$$

where $w_1 = 1, w_2 = 2, ...w_n = n$

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> lwma = ql.lwma(df['close'], periods = 10)
>>> print(lwma)
[nan nan nan ... 209.50327273 210.35927273 210.96381818]
```

3.1.4 Simple Moving Average

qufilab.**sma**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated simple moving average values.

Notes

The calculation of sma is a equally weighted mean for the last *n* days.

$$sma_K = \frac{price_K + price_{K-1} + ... + price_{K-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} price_{K-i}$$

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> sma = ql.sma(df['close'], periods = 10)
>>> print(sma)
[nan nan nan ... 209.872 209.695 209.749]
```

3.1.5 Smoothed Moving Average

qufilab.**smma**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated smoothed moving average values.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> smma = ql.smma(df['close'], periods = 10)
>>> print(smma)
[nan nan nan ... 208.85810754 209.43029679 209.78926711]
```

3.1.6 T3 Moving Average

qufilab.**t3**(*data*, *periods*, *volume_factor=0.7*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

volume_factor [*float*, optional] What volume factor to be used when calculating the constants. See *Notes* below for implementation.

Returns

ndarray An array containing calculated t3 moving average values.

Notes

The t3 moving average indicator utilize many different exponential moving averages and is calculated with:

$$
\begin{aligned}
t3 &= c_1 e_6 + c_2 e_5 + c_3 e_4 + c_4 e_3 \\
e_1 &= ema(data) \\
e_2 &= ema(e_1) \\
e_3 &= ema(e_2) \\
e_4 &= ema(e_3) \\
e_5 &= ema(e_4) \\
e_6 &= ema(e_5) \\
c_1 &= -a^3 \\
c_2 &= 3a^2 + 3a^3 \\
c_3 &= \breve{}6a^2\breve{}3a\breve{}3a^3 \\
c_4 &= 1 + 3a + a^3 + 3a^2
\end{aligned}
$$

where *a* is the volume factor and is typically set to *0.7*.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> t3 = ql.t3(df['close'], periods = 10)
>>> print(t3)
[nan nan nan ... 210.08668472 210.2348457 210.47802463]
```

3.1.7 Triangular Moving Average

qufilab.**tma**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing calculated triangular moving average values.

Notes

The triangular moving average is calculated by taking an sma of an sma.

$$tma = sma(sma(price, n_1), n_2)$$

As seen above, two different periods are used in this implementation. If the parameter *periods* is even

$$n_1 = \frac{periods}{2}$$
$$n_2 = \frac{periods}{2} + 1$$

otherwise they are rounded up after the following calculation

$$n_1 = n_2 = \frac{periods + 1}{2}$$

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> tma = ql.tma(df['close'], periods = 10)
>>> print(tma)
[nan nan nan ... 208.93833333 209.115 209.684]
```

3.1.8 Weighted Close

qufilab.**wc**(*high*, *low*, *close*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

close [*ndarray*] An array containing closing prices.

Returns

ndarray An array containing calculated weighted close values.

Notes

The weighted close is defined as:

$$wc_K = \frac{2 \cdot close_K + high_K + low_K}{4}$$

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> wc = ql.wc(df['high'], df['low'], df['close'])
>>> print(wc)
[153.1975 154.09 154.3075 ... 210.1875 213.2675 213.785]
```

3.2 Momentum

3.2.1 Absolute Price Oscillator

qufilab.**apo**(*price*, *period_slow=26*, *period_fast=12*, *ma='sma'*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing price values.

period_slow [*int*, optional] Number of periods for the slow moving average. Default to 26.

period_fast [*int*, optional] Number of fast periods for the fast moving average. Default to 12.

ma [{'sma', 'ema'}, optional] Type of moving average to be used. Default to 'sma'

Returns

ndarray Array of type float64 or float32 containing the calculated absolute price oscillator values.

3.2.2 Aroon Indicator

qufilab.**aroon**(*high*, *low*, *period=20*)

Parameters

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

period [*int*, optional] Number of periods to be used. Defaults to 20.

Returns

ndarray Array of type float64 or float32 containing the calculated aroon indicator values.

3.2.3 Balance of Power

qufilab.**bop**(*high*, *low*, *open_*, *close*)

Parameters

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

open_ [*ndarray*] Array of type float64 or float32 containing open prices.

close [*ndarray*] Array of type float64 or float32 containing closing prices.

Returns

ndarray Array of type float64 or float32 containing the calculated balance of power values.

3.2.4 Commodity Channel Index

qufilab.**cci**(*close*, *high*, *low*, *period=20*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing closing prices.

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

period [*int*, optional] Number of periods to be used. Defaults to 20.

Returns

ndarray Array of type float64 or float32 containing the calculated commodity channel index values.

3.2.5 Chande Momentum Indicator

qufilab.**cmo**(*close*, *period*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing closing prices.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the calculated chande momentum values.

3.2.6 MACD

qufilab.**macd**(*price*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing the prices to calculate macd from.

Returns

macd [*ndarray*] Array of type float64 or float32 containing the macd values.

signal [*ndarray*] Array of type float64 or float32 containing the signal values.

3.2.7 Money Flow Index

qufilab.**mfi**(*high*, *low*, *close*, *volume*, *period*)

Parameters

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

close [*ndarray*] Array of type float64 or float32 containing closing prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the calculated money flow index values.

3.2.8 Momentum Indicator

qufilab.**mi**(*price*, *period*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing price values.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the calculated momentum indicator values.

3.2.9 Percentage Price Oscillator

qufilab.**ppo**(*price*, *period_fast=12*, *period_slow=26*, *ma='ema'*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing price values.

period_fast [*int*, optional] Number of fast periods for the fast moving average. Default to 12.

period_slow [*int*, optional] Number of periods for the slow moving average. Default to 26.

ma [{'ema', 'sma'}, optional] Type of moving average to be used. Default to 'ema'

Returns

ndarray Array of type float64 or float32 containing the calculated percentage price values.

3.2.10 Relative Strength Index

qufilab.**rsi**(*price*, *period*, *rsi_type='smoothed'*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing the data to calculate rsi from.

period [*int*] Number of periods to be used.

rsi_type [{'smoothed', 'standard'}, optional] Specify what kind of averaging should be used for calculating the average gain/ average loss. Standard is the Wilder's smoothing.

Returns

ndarray Returns a numpy ndarray with type float64 or float32.

3.2.11 Price Rate of Change

qufilab.**roc**(*price*, *period*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing price values.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the calculated rate of change values.

3.2.12 Volume Price Trend

qufilab.**vpt**(*price*, *volume*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing price values.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated volume price trend values.

3.2.13 William's R

qufilab.**willr**(*close*, *high*, *low*, *period*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing the closing prices.

high [*ndarray*] Array of type float64 or float32 containing the high prices.

low [*ndarray*] Array of type float64 or float32 containing the low prices.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the william's r values.

3.3 Volatility

3.3.1 Average True Range

qufilab.**atr**(*close*, *high*, *low*, *period*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing the closing prices.

high [*ndarray*] Array of type float64 or float32 containing the high prices.

low [*ndarray*] Array of type float64 or float32 containing the low prices.

period [*int*] Number of periods to be used.

Returns

ndarray Array of type float64 or float32 containing the calculated average true range values.

3.3.2 Bollinger Bands

qufilab.**bbands**(*price*, *period*, *deviation=2*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing the prices.

period [*int*] Number of periods to be used.

deviation [*int*, optional] Number of standard deviations from the mean. Defaults to 20.

Returns

upper [*ndarray*] Upper bollinger band.

middle [*ndarray*] middle bollinger band.

lower [*ndarray*] lower bollinger band.

3.3.3 Chaikin Volatility

qufilab.**cv**(*high*, *low*, *period=10*, *smoothing_period=10*)

Parameters

high [*ndarray*] Array of type float64 or float32 containing the high prices.

low [*ndarray*] Array of type float64 or float32 containing the low prices.

period [*int*, optional] Number of periods to be used. Defaults to 10.

smooting_period [*int*, optional] Number of periods to be used for smoothing chaikin volatility values. Defaults to 10.

Returns

ndarray Array of type float64 or float32 containing the calculated chaikin volatility values.

3.3.4 Keltner Channels

qufilab.**kc**(*close*, *high*, *low*, *period=20*, *period_atr=20*, *deviation=2*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing the closing prices.

high [*ndarray*] Array of type float64 or float32 containing the high prices.

low [*ndarray*] Array of type float64 or float32 containing the low prices.

period [*int*, optional] Number of periods to be used. Defaults to 20.

period_atr [*int*, optional] Number of periods to be used for the average true range calculations. Defaults to 20.

deviation [*int*, optional] Number of deviations from the mean. Defaults to 2.

Returns

upper [*ndarray*] Upper keltner band.

middle [*ndarray*] middle keltner band.

lower [*ndarray*] lower keltner band.

3.4 Volume

3.4.1 Accumulation Distribution

qufilab.**acdi**(*close*, *high*, *low*, *volume*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing closing prices.

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated accumulation distribution values.

3.4.2 Chaikin Indicator

qufilab.**ci**(*close*, *high*, *low*, *volume*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing closing prices.

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated chaikin indicator values.

3.4.3 Chaikin Money Flow

qufilab.**cmf**(*close*, *high*, *low*, *volume*, *period=21*)

Parameters

close [*ndarray*] Array of type float64 or float32 containing closing prices.

high [*ndarray*] Array of type float64 or float32 containing high prices.

low [*ndarray*] Array of type float64 or float32 containing low prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

period [*int*, optional] Number of periods to use. Defaults to 21.

Returns

ndarray Array of type float64 or float32 containing the calculated chaikin money flow values.

3.4.4 Negative Volume Index

qufilab.**nvi**(*price*, *volume*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated negative volume index values.

3.4.5 On Balance Volume

qufilab.**obv**(*price*, *volume*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated on balance volume values.

3.4.6 Positive Volume Index

qufilab.**pvi**(*price*, *volume*)

Parameters

price [*ndarray*] Array of type float64 or float32 containing prices.

volume [*ndarray*] Array of type float64 or float32 containing volume values.

Returns

ndarray Array of type float64 or float32 containing the calculated positive volume index values.

3.5 Statistics

3.5.1 Beta

qufilab.**beta**(*data*, *market*, *periods*, *normalize=False*)

Parameters

data [*ndarray*] An array containing values.

market [*ndarray*] An array containing market values to be used as the comparison when calculating beta.

periods [*int*] Number of periods to be used.

normalize [*bool*, optional] Specify whether to normalize the standard deviation calculation within the beta calculation with n - 1 instead of n. Defaults to False.

Returns

ndarray An array containing beta values.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> df_market = ql.load_sample('DJI')
>>> beta = ql.beta(df['close'], df_market['close'], periods = 10)
>>> print(beta)
[nan nan nan ... 0.67027616 0.45641977 0.3169785]
```

3.5.2 Covariance

qufilab.**cov**(*data*, *market*, *periods*, *normalize=True*)

Parameters

data [*ndarray*] An array containing values.

market [*ndarray*] An array containing market values to be used as the comparison when calculating beta.

periods [*int*] Number of periods to be used.

normalize [*bool*, optional] Specify whether to normalize covariance with n - 1 instead of n. Defaults to *True*.

Returns

ndarray An array containing covariance values.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> df_market = ql.load_sample('DJI')
>>> cov = ql.cov(df['close'], df_market['close'], periods = 10)
>>> print(cov)
[nan nan nan ... -360.37842558  -99.1077715 60.84627274]
```

3.5.3 Percentage Change

qufilab.**pct_change**(*data*, *periods*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

Returns

ndarray An array containing percentage change for the specified periods.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> pct_change = ql.pct_change(df['close'], periods = 10)
>>> print(pct_change)
[nan nan nan ... -1.52155537 -0.81811879 0.25414157]
```

3.5.4 Standard Deviation

qufilab.**std**(*data*, *periods*, *normalize=True*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

normalize [*bool*, optional] Specify whether to normalize the standard deviation with n - 1 instead of n. Defaults to True.

Returns

ndarray An array containing standard deviation values for the specified periods.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> sma = ql.std(df['close'], periods = 10)
>>> print(sma)
[nan nan nan ... 3.31897842 2.9632574  3.02394683]
```

3.5.5 Variance

qufilab.**var**(*data*, *periods*, *normalize=True*)

Parameters

data [*ndarray*] An array containing values.

periods [*int*] Number of periods to be used.

normalize [*bool*, optional] Specify whether to normalize the standard deviation with n - 1 instead of n. Defaults to *True*.

Returns

ndarray An array containing variance values.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> # Load a sample dataframe.
>>> df = ql.load_sample('MSFT')
>>> print(df['close'].dtype)
float64
>>> var = ql.var(df['close'], periods = 10)
```

(continues on next page)

(continued from previous page)

```
>>> print(var)
[nan nan nan ... 11.01561778  8.78089444 9.14425444]
```

CHAPTER

FOUR

CANDLESTICK PATTERNS

4.1 Abandoned Baby - Bear

qufilab.**abandoned_baby_bear**(*high*, *low*, *open_*, *close*, *periods=10*)

Abandoned Baby Bear

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

pattern [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.2 Abandoned Baby - Bull

qufilab.**abandoned_baby_bull**(*high*, *low*, *open_*, *close*, *periods=10*)

Abandoned Baby Bull

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

pattern [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.3 Belt Hold - Bear

qufilab.**belthold_bear**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)
Belt Hold Bear

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

Returns

pattern [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.4 Belt Hold - Bull

qufilab.**belthold_bull**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)
Belt Hold Bull

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

Returns

pattern [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.5 Doji

qufilab.**doji**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

doji [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> df = ql.load_sample('MSFT')
>>> doji = ql.doji(df['high'], df['low'], df['open'], df['close'])
>>> print(doji)
[False False False ... False False False]
```

4.6 Dragonfly Doji

qufilab.**dragonfly_doji**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

dragonfly_doji [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> df = ql.load_sample('MSFT')
>>> dragonfly_doji = ql.dragonfly_doji(df['high'], df['low'], df['open'], df[
↪'close'])
>>> print(dragonfly_doji)
[False False False ... False False False]
```

4.7 Engulfing - Bear

qufilab.**engulfing_bear**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

engulfing [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.8 Engulfing - Bull

qufilab.**engulfing_bull**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

engulfing [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.9 Hammer

qufilab.**hammer**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

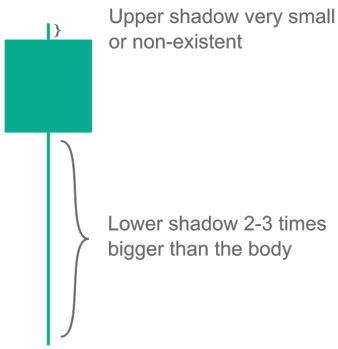
shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using i.e. 5%, upper shadow can be as long as 5% of the candlestick body size. This exist to allow some margin and not exclude the shadows entirely.

Returns

hammer [*ndarray*] A numpy array of type bool specifying true whether a pattern has been found or false otherwise.

Notes

Observe that the lower shadow shall be bigger than 2x the body, but lower than 3x the body.



Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> df = ql.load_sample('MSFT')
>>> hammer = ql.hammer(df['high'], df['low'], df['open'], df['close'])
>>> print(hammer)
[False False False ... False False False]
```

4.10 Harami - Bear

qufilab.**harami_bear**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

harami_bear [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.11 Harami - Bull

qufilab.**harami_bull**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

harami_bull [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.12 Inverted Hammer

qufilab.**inverted_hammer**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using i.e. 5%, lower shadow can be as long as 5% of the candlestick body size. This exist to allow some margin and not exclude the shadows entirely.

Returns

inverted_hammer [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> df = ql.load_sample('MSFT')
>>> inverted_hammer = ql.inverted_hammer(df['high'], df['low'], df['open'], df[
↪'close'])
>>> print(inverted_hammer)
[False False False ... False False False]
```

4.13 Kicking - Bear

qufilab.**kicking_bear**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

Returns

kicking_bear [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.14 Kicking - Bull

qufilab.**kicking_bull**(*high*, *low*, *open_*, *close*, *periods=10*, *shadow_margin=5.0*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

Returns

kicking_bull [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.15 Marubozu White

qufilab.**marubozu_white**(*high*, *low*, *open_*, *close*, *shadow_margin=5.0*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

marubozu_white [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

Examples

```
>>> import qufilab as ql
>>> import numpy as np
...
>>> df = ql.load_sample('MSFT')
>>> marubozu_white = ql.marubozu_white(df['high'], df['low'], df['open'], df[
↪'close'])
>>> print(marubozu_white)
[False False False ... False False False]
```

4.16 Marubozu Black

qufilab.**marubozu_black**(*high*, *low*, *open_*, *close*, *shadow_margin=5.0*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

shadow_margin [*float*, optional] Specify what margin should be allowed for the shadows. By using for example 5%, both the lower and upper shadow can be as high as 5% of the candlestick body size. This exist to allow some margin (not restrict to no shadow).

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

marubozu_black [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.17 Piercing

qufilab.**piercing**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

piercing [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.18 Spinning Top White

qufilab.**spinning_top_white**(*high*, *low*, *open_*, *close*, *periods=10*)

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

spinning_top_white [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

4.19 Three White Soldiers

qufilab.**tws**(*high*, *low*, *open_*, *close*, *periods=10*)
Three White Soldiers

Parameters

high [*ndarray*] An array containing high prices.

low [*ndarray*] An array containing low prices.

open_ [*ndarray*] An array containing open prices.

close [*ndarray*] An array containing close prices.

periods [*int*, optional] Specifying number of periods for trend identification.

Returns

tws [*ndarray*] A numpy ndarray of type bool specifying true whether a pattern has been found or false otherwise.

CHAPTER

FIVE

FOR MAINTAINERS

5.1 Adding new indicators

The process of adding a new indicator to the library consists of several steps and is documented here for future reference.

1. When creating a new indicator, start by declaring what type of indicator it is. For example, the rsi indicator is a momentum indicator, and hence should be implemented in the *momentum.cc* file. The first line of the docstring in the *.cc* file should be **Implementation of INDICATOR**, i.e. **Implementation of SMA**. This is needed for the linking of source code to github, since *docs/source/conf.py* manually retrieves the source code from github and searches for that line to get the correct line number. If the indicator have dependencies from other functions in other script, be sure to include the header files and update *qufilab/indicators/CMakeLists.txt*. This is needed when building manually, since the linking won't work otherwise.
2. When the implementation is done, the documentation for the indicator needs to be created. Start by adding the indicator to the correct entry in the *docs/source/indicators.yaml* file. This file ensures that correct links to the underlying source code can be made when building the docs.
3. Add the correct specifications to the *docs/source/indicators.rst* directory.

INDEX

V

W