Package 'QuantTools'

October 14, 2020

Type Package

Title Enhanced Quantitative Trading Modelling

Version 0.5.7.1

Author Stanislav Kovalevsky

Maintainer Stanislav Kovalevsky <so.kovalevsky@gmail.com>

Description Download and organize historical market data from multiple sources like Yahoo (<https://finance.yahoo.com>), Google (<https://www.google.com/finance>), Finam (<https://www.finam.ru/profile/moex-akcii/sberbank/export/>), MOEX (<https://www.moex.com/en/derivatives/contracts.aspx>) and IQFeed (<https://www.iqfe
ing algorithms in modern C++11 with powerful event driven tick processing API including trading costs and exchange communication latency and transform detailed data seamlessly into R. In just few lines of code you will be able to visualize every step of your trading model from tick data to multi dimensional heat maps.

URL https://quanttools.bitbucket.io

BugReports https://bitbucket.org/quanttools/quanttools/issues

License GPL-3

Encoding UTF-8

LazyData false

Depends data.table, R (>= 2.10)

Imports methods, fasttime, RCurl, readxl, Rcpp (>= 0.12.12), R6

LinkingTo Rcpp

SystemRequirements C++11

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-14 16:34:26 UTC

R topics documented:

`add_last_values` *Add last values marks to the right of active time series plot*

Description

Add last values marks to the right of active time series plot

Usage

```
add_last_values(data, ylim, col)
```

Arguments

`data`	`data.frame` or `data.table` object of plotted data
`ylim`	user specified range of `data`
`col`	same as in `plot_ts`

Details

Used in `plot_ts` internally.

See Also

Other graphical functions: `add_legend`, `distinct_colors`, `empty_plot`, `hist_dt`, `lines_ohlc`, `lines_stacked_hist`, `multi_heatmap`, `plot_table`, `plot_ts`

add_legend *Add legend to active time series plot*

Description

Add legend to active time series plot

Usage

```
add_legend(position = "topright", names, col = "auto", lty = 1, lwd = 1,
  pch = NA)
```

Arguments

position	same as in plot_ts except 'n'
names	line labels
col	same as in plot_ts
lty, lwd	same as in lines
pch	same as in points

Details

Used in plot_ts internally.

See Also

Other graphical functions: add_last_values, distinct_colors, empty_plot, hist_dt, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_table, plot_ts

back_test *Generic back test function*

Description

Back test by enter and exit signals with stop loss on price history. Execution is immediate. Useful for testing on daily data.

Usage

```
back_test(enter, exit, price, stop_loss = -1000, side = 1L)
```

Arguments

enter	bool vector of length n of enter signals
exit	bool vector of length n of exit signals
price	numeric vector of length n of prices
stop_loss	relative stop loss, must be negative
side	direction of enter order, -1:short, 1:long

Value

trades data.table with columns price_enter,price_exit,mtm_min,mtm_max,id_enter,id_exit,pnl_trade,side

---

BBands *C++ Bollinger Bands class*

Description

C++ class documentation

Arguments

n	indicator period
k	number of standard deviations

Details

R function bbands.

Usage

```
BBands( int n,double k )
```

Public Members and Methods

Name	Return Type	Description
Add(InputType value)	void	update indicator
Reset()	void	reset to initial state
IsFormed()	bool	is indicator value valid?
GetValue()	BBandsValue	has members double upper, lower, sma
GetUpperHistory()	std::vector< double >	return upper band history
GetLowerHistory()	std::vector< double >	return lower history
GetSmaHistory()	std::vector< double >	return sma history
GetHistory()	List	return values history data.table with columns upper, lower, sma

See Also

Other C++ indicators: Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

---

bbands *Bollinger Bands*

Description

Bollinger bands is a mix of Rolling Range and SMA indicators. It shows the average price and its range over n past values based on price volatility.

Usage

```
bbands(x, n, k)
```

Arguments

x	numeric vectors
n	window size
k	number of standard deviations

Value

Returns data.table with columns `upper`,`lower`,`sma`.

See Also

Other technical indicators: crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma, stochastic

---

bw *Check if values are between specified interval*

Description

Check if values are between specified interval

Usage

```
bw(x, interval)

x %bw% interval
```

Arguments

x	vector
interval	vector of length 1 or 2, see 'Examples' section

Details

If second element of interval contains time selection is closed on the left only (a <= x < b) otherwise selection is closed (a <= x <= b).

Examples

```
data( ticks )

# bw is very usefull to filter time series data:
# select single year
ticks[ time %bw% '2016' ]

# select single month
ticks[ time %bw% '2016-05' ]

# select single date
ticks[ time %bw% '2016-05-11' ]
# also works with Date class
ticks[ time %bw% as.Date( '2016-05-11' ) ]

# select single hour
ticks[ time %bw% '2016-05-11 10' ]

# select single minute
ticks[ time %bw% '2016-05-11 10:20' ]

# select single second
ticks[ time %bw% '2016-05-11 10:20:53' ]

# select between two months inclusive
ticks[ time %bw% '2016-05/2016-08' ]

# select from month begin and date
ticks[ time %bw% '2016-05/2016-06-23' ]

# select between two timestamps
ticks[ time %bw% '2016-05-02 09:30/2016-05-02 11:00' ]
# also works with incomplete timestamps
ticks[ time %bw% '2016-05-02 09:30/2016-05-02 11' ]

# select all dates but with time between 09:30 and 16:00
ticks[ time %bw% '09:30/16:00' ]

# also bw can be used as a shortcut for 'a <= x & x <= b' for non-'POSIXct' classes:
```

```
# numeric
15:25 %bw% c( 10, 20 )

# character
letters %bw% c( 'a', 'f' )

# dates
Sys.Date() %bw% ( Sys.Date() + c( -10, 10 ) )
```

`calc_decimal_resolution` *Calculate decimal resolution*

Description

Calculate decimal resolution

Usage

```
calc_decimal_resolution(x)
```

Arguments

`x`	numeric vector

Details

Used in `add_last_values` internally.

`Candle` *C++ Candle class*

Description

C++ class documentation

Arguments

`id`	id
`open`	price
`high`	price
`low`	price
`close`	price
`time`	seconds since epoch
`volume`	volume
`timeFrame`	timeframe in seconds

Usage

```
Candle{ int id,double open,double high,double low,double close,double time,int volume,int
timeFrame }
```

See Also

Other backtesting classes: Cost, Indicator, Order, Processor, Tick

Other C++ classes: BBands, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

---

Cost *C++ Trading Commissions class*

Description

C++ class documentation

Arguments

pointValue	price point value (1 for stocks)
cancel	absolute commission per order cancel
order	absolute commission per order
stockAbs	absolute commission per stock / contract
tradeAbs	absolute commission per trade
tradeRel	relative commission per trade volume
longAbs	absolute commission/refund per long position
longRel	relative commission/refund per long volume
shortAbs	absolute commission/refund per short position
shortRel	relative commission/refund per short volume

Usage

```
Cost = {}
```

IMPORTANT

Positive value means refund, negative value means cost!

See Also

Other backtesting classes: Candle, Indicator, Order, Processor, Tick

Other C++ classes: BBands, Candle, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

Crossover *C++ Crossover class*

Description

C++ class documentation

Details

R function crossover.

Usage

```
Crossover
```

Public Members and Methods

Name	Return Type	Description
`Add( std::pair< double, double > value )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`IsAbove()`	`bool`	first just went above second?
`IsBelow()`	`bool`	first just went below second?
`GetHistory()`	`factor`	factor vector with levels `UP, DN`

See Also

Other C++ indicators: BBands, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

crossover *Crossover*

Description

Crossover is binary indicator indicating the moment when one value goes above or below another.

Usage

```
crossover(x, y)
```

Arguments

x, y	numeric vectors

See Also

Other technical indicators: bbands, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma, stochastic

---

distinct_colors *Distinct colors vector*

Description

Distinct colors vector

Usage

```
distinct_colors
```

Format

An object of class character of length 25.

Details

Distinct colors vector.

See Also

Other graphical functions: add_last_values, add_legend, empty_plot, hist_dt, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_table, plot_ts

---

dof *Do calculation on data.table excluding first column*

Description

Do calculation on data.table excluding first column

Usage

```
dof(x, fun, ...)

dofc(x, fun, ...)

x %dof% fun

x %dofc% fun
```

Arguments

x	data.table
fun	function or text formula where x represents argument
...	additional parameters to function if action is function

Details

DO Function (Column-wise/Row-wise)

Ema *C++ Exponential Moving Average class*

Description

C++ class documentation

Arguments

n	indicator period

Details

R function ema.

Usage

```
Ema( int n )
```

Public Members and Methods

Name	Return Type	Description
Add(double value)	void	update indicator
GetValue()	double	return value
Reset()	void	reset to initial state
IsFormed()	bool	is indicator value valid?
GetHistory()	std::vector<double>	return values history

See Also

Other C++ indicators: BBands, Crossover, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

ema *Exponential Moving Average*

Description

Exponentially weighted moving average aka EMA is exponentially weighted SMA. EMAs have faster response to recent value changes than SMAs.

Usage

```
ema(x, n)
```

Arguments

x	numeric vectors
n	window size

See Also

Other technical indicators: bbands, crossover, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma, stochastic

empty_plot *Plot empty plot*

Description

Plot empty plot

Usage

```
empty_plot()
```

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, hist_dt, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_table, plot_ts

gen_futures_codes *Generate futures contract codes and schedule between dates*

Description

Generate futures contract codes and schedule between dates

Usage

```
gen_futures_codes(contract, from, to, frequency, day_exp,
  year_last_digit = FALSE)
```

Arguments

contract	contract base name
from, to	text dates in format "YYYY-MM-DD"
frequency	expiration frequency, e.g. 3 for quarterly contracts
day_exp	expiration day number, e.g. 15 for middle of month
year_last_digit	should only last digit of year present in code?

Value

returns data.table with columns code,from,to,contract_id.

get_market_data *Download historical market data*

Description

Download historical market data

Usage

```
get_yahoo_data(symbol, from, to, split.adjusted = TRUE,
  dividend.adjusted = TRUE)

get_yahoo_splits_and_dividends(symbol, from, to = from)

get_google_data(symbol, from, to = from)

get_finam_data(symbol, from, to = from, period = "day", local = FALSE)

get_iqfeed_data(symbol, from, to = from, period = "day", local = FALSE)
```

```
get_moex_options_data(code, from, to = from, period = "tick",
  local = TRUE)

get_moex_futures_data(code, from, to = from, period = "tick",
  local = TRUE)

get_moex_continuous_futures_data(contract, from, to, frequency, day_exp)
```

Arguments

`symbol`	symbol name
`from, to`	text dates in format `"YYYY-mm-dd"`
`split.adjusted`	should data be split adjusted?
`dividend.adjusted`	should data be split adjusted?
`period`	candle period `tick,1min,5min,10min,15min,30min,hour,day,week,month`
`local`	should data be loaded from local storage? See 'Local Storage' section
`code`	futures or option code name, e.g. `"RIU6"`
`contract, frequency, day_exp`	same as in `gen_futures_codes`

Details

Use external websites to get desired symbol name for Finam, MOEX, IQFeed, Yahoo and Google sources.

IQFeed:

`data.table` with following data returned:

daily:	date, open, high, low, close, volume, open_interest
intraday:	date, open, high, low, close, volume
tick:	time, price, volume, size, bid, ask, tick_id, basis_for_last, trade_market_center, trade_conditions

See iqfeed specification for details.
Note: `from` and `to` can be set as text in format `"YYYY-mm-dd HH:MM:SS"`.

Finam:

`data.table` with following data returned:

daily:	date, open, high, low, close, volume
intraday:	date, open, high, low, close, volume
tick:	time, price, volume

Yahoo:
`data.table` with following data returned:

daily:	date, open, high, low, close, adj_close, volume
splits and dividends:	date, value, event

Google:
`data.table` with following data returned:

daily:	date, open, high, low, close, volume

MOEX: data can be retrieved from local storage only in order to minimize load on MOEX data servers. See 'Local Storage' section.

Local Storage

It is recommended to store tick market data locally. Load time is reduced dramatically. It is a good way to collect market data as e.g. IQFeed gives only 180 days of tick data if you would need more it will cost you a lot. See `store_market_data` for details.
Only IQFeed, Finam and MOEX data supported.

Note

Timestamps timezones set to UTC.

Examples

```
get_finam_data( 'GAZP', '2015-01-01', '2016-01-01' )
get_finam_data( 'GAZP', '2015-01-01', '2016-01-01', 'hour' )
get_finam_data( 'GAZP', Sys.Date(), Sys.Date(), 'tick' )

get_iqfeed_data( 'MSFT', '2015-01-01', '2016-01-01' )
get_iqfeed_data( 'MSFT', '2015-01-01', '2016-01-01', 'hour' )
get_iqfeed_data( 'MSFT', Sys.Date() - 3, Sys.Date() , 'tick' )

get_google_data( 'MSFT', '2015-01-01', '2016-01-01' )
get_yahoo_data( 'MSFT', '2015-01-01', '2016-01-01' )

get_moex_futures_data( 'RIH9', '2009-01-01', '2009-02-01', 'tick', local = T )
get_moex_options_data( 'RI55000C9', '2009-01-01', '2009-02-01', 'tick', local = T )
get_moex_continuous_futures_data( 'RI', '2016-01-01', '2016-11-01', frequency = 3, day_exp = 15 )
```

hist_dt *Plot histogram of data.table by columns*

Description

Plot histogram of data.table by columns

Usage

```
hist_dt(dt, bin_width = diff(range(dt, na.rm = TRUE))/10, coeff = 0.8,
  main = "")
```

Arguments

dt	data.table
bin_width	truncate data by this value
coeff	group width in [0,1]
main	plot title

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, empty_plot, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_table, plot_ts

Indicator *C++ Indicator Base class*

Description

C++ class documentation

Arguments

InputType	input type
ValueType	output type
HistoryType	history type

Usage

```
class AnyIndicator : public Indicator< InputType,ValueType,HistoryType > { }
```

Public Members and Methods

Following methods must be specified for AnyIndicator

Name	Return Type	Description
`Add( InputType input )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetHistory()`	`HistoryType`	return values history

See Also

Other backtesting classes: `Candle`, `Cost`, `Order`, `Processor`, `Tick`

Other C++ indicators: `BBands`, `Crossover`, `Ema`, `RollLinReg`, `RollPercentRank`, `RollRange`, `RollSd`, `RollVolumeProfile`, `Rsi`, `Sma`, `Stochastic`

Other C++ classes: `BBands`, `Candle`, `Cost`, `Crossover`, `Ema`, `Order`, `Processor`, `RollLinReg`, `RollPercentRank`, `RollRange`, `RollSd`, `RollVolumeProfile`, `Rsi`, `Sma`, `Stochastic`, `Tick`

`iqfeed` *IQFeed*

Description

IQFeed

Details

Retrieves IQFeed historical market data like ticks and candles.

Basis For Last

C	Last Qualified Trade
E	Extended Trade = Form T trade
O	Other Trade = Any trade not accounted for by C or E.
S	Settle = Daily settle, only applicable to commodities.

Markets

Market Id	Short Name	Long Name
1	NGM	Nasdaq Global Market
2	NCM	National Capital Market
3	OTC	Nasdaq other OTC
4	OTCBB	Nasdaq OTC Bulletin Board
5	NASDAQ	Nasdaq
6	NYSE_MKT	NYSE MKT (Equities and Bonds)
7	NYSE	New York Stock Exchange

8	CHX	Chicago Stock Exchange
9	PHLX	Philadelphia Stock Exchange
10	NSX	National Stock Exchange
11	NYSE_ARCA	NYSE Archipelago
12	BX	Boston Stock Exchange
13	CBOE	Chicago Board Options Exchange
14	OPRA	OPRA System
15	NASD_ADF	Nasdaq Alternate Display facility
16	ISE	International Stock Exchange
17	BOX	Boston Options Exchange
18	BATS	Better Alternative Trading System
19	NTRF	Nasdaq Trade Reporting Facility
20	PBOT	Philadelphia Board Of Trade
21	NGSM	Nasdaq Global Select Market
22	CANTOR	Cantor Fitzgerald Exchange Treasury Funds
23	C2	CBOE C2 Options Exchange
24	NYSE_TRF	NYSE Trade Reporting Facility
25	EDGA	Direct Edge A
26	EDGX	Direct Edge X
27	DTN	DTN
28	BYX	BATS Y Exchange
29	RUSSELL-FL	Russell Investments (Fee-Liable)
30	CBOT	Chicago Board Of Trade
31	DJ	Dow Jones (CBOT)
32	CFE	CBOE Futures Exchange
33	KCBOT	Kansas City Board Of Trade
34	CME	Chicago Mercantile Exchange
35	MGE	Minneapolis Grain Exchange
36	NYMEX	New York Mercantile Exchange
37	COMEX	Commodities Exchange Center
38	ICEFU	International Commodities Exchange Futures US
39	NYLUS	NYSE LIFFE US
40	CME-FL	CME Indexes (Fee Liable)
42	CBOTMINI	Chicago Board Of Trade Mini Sized Contracts
43	CMEMINI	Chicago Mercantile Exchange Mini Sized Contracts
44	USFE	US Futures Exchange
45	NYMEXMINI	Commodities Exchange Center Mini Sized Contracts
46	GREENX	The Green Exchange
47	CLEARPORT	New York Mercantile Exchange
48	COMEXMINI	New York Mercantile Exchange Mini Sized Contracts
50	TSE	Toronto Stock Exchange
51	MSE	Montreal Stock Exchange
52	CVE	Canadian Venture Exchange
53	WSE	Winnipeg Stock Exchange
54	ICEFC	International Commodities Exchange Futures Canada
55	MX	Montreal Exchange
56	LSE	London Stock Exchange
57	FTSE	Financial Times Stock Exchange

60	MDEX	Bursa Malaysia Derivatives
61	ICEFI	International Commodities Exchange Futures Derivatives
62	LME	London Metals Exchange
63	ICEEC	International Commodities Exchange European Commodities
64	ASXCM	ASX24 Commodities Exchange
65	DME	Dubai Mercantile Exchange
66	BMF	Brazilian Mercantile & Future Exchange
67	SGX	Singapore International Monetary Exchange
68	EUREX	European Exchange
69	ENID	Euronext Index Derivatives
70	ICEEF	International Commodities Exchange European Financials
71	ENCOM	Euronext Commodities
72	TULLETT	Tullett Liberty (Forex)
73	BARCLAYS	Barclays Bank (Forex)
74	FXCM	Forex Capital Markets
75	WTB	Warenterminborse Hannover
76	MGKB	MGE-KCBOT (InterCommodity Spreads)
77	MGCB	MGE-CBOT (InterCommodity Spreads)
78	TENFORE	Tenfore Systems
79	NYDME	NYMEX-DME (InterCommodity Spreads)
80	PSX	Philadelphia Stock Exchange
81	TGE	Tokyo Grain Exchange
82	TOCOM	Tokyo Commodities Exchange
83	SAFEX	South African Futures Exchange
84	EEXP	European Energy Exchange - Power
85	EEXN	European Energy Exchange - Natural Gas
86	EEXE	European Energy Exchange - Emission Rights
87	EEXC	European Energy Exchange - Coal
88	MIAX	Miami International Securities Exchange
89	KBCB	KCBOT-CBOT (InterCommodity Spreads)
90	PK_SHEETS	Pink Sheets - No Tier
91	PK_QXPREM	Pink Sheets - OTCQX - PremierQX Tier
92	PK_QXPRIME	Pink Sheets - OTCQX - PrimeQX Tier
93	PK_IQXPREM	Pink Sheets - OTCQX - International PremierQX Tier
94	PK_IQXPRIME	Pink Sheets - OTCQX - International PrimeQX Tier
95	PK_OTCQB	Pink Sheets - OTCBB Pink Sheets dually Quoted Tier
96	PK_BBONLY	Pink Sheets - OTCBB Only Tier
97	PK_CURRENT	Pink Sheets - Current Tier
98	PK_LIMITED	Pink Sheets - Limited Tier
99	PK_NOINFO	Pink Sheets - No Information Tier
100	PK_GREY	Pink Sheets - Grey Market Tier
101	PK_YL_SHEETS	Yellow Sheets
102	PK_PR_SHEETS	Partner Sheets
103	PK_GL_SHEETS	Global Sheets
104	PK_NYSE	Pink Sheets - NYSE Listed
105	PK_NASDAQ	Pink Sheets - NASDAQ Listed
106	PK_NYSE_AMEX	Pink Sheets - NYSE AMEX Listed
107	PK_ARCA	Pink Sheets - ARCA Listed

108	NYSE_AMEX	NYSE AMEX Options Exchange
109	GLOBEX_RT	CME GLOBEX Group Authorization
110	CME_GBX	Chicago Mercantile Exchange (GLOBEX)
111	CBOT_GBX	Chicago Board Of Trade (GLOBEX)
112	NYMEX_GBX	New York Mercantile Exchange (GLOBEX)
113	COMEX_GBX	Commodities Exchange Center (GLOBEX)
114	DME_GBX	Dubai Mercantile Exchange (GLOBEX)
115	RUSSELL	Russell Investments
116	BZX	BATS Z Exchange
117	CFTC	US Commodity Futures Trading Commission
118	USDA	US Department of Agriculture
119	WASDE	World Supply and Demand Estimates Report
120	GRNST	Grain Stock Report
121	GEMINI	ISE Gemini Options Exchange
122	ARGUS	Argus Energy
123	RACKS	Racks Energy
124	SNL	SNL Energy
125	RFSPOT	Refined Fuels Spots Exchange
126	EOXNGF	EOX Live Natural Gas Forward Curve
127	EOXPWF	EOX Live Power Forward Curve
128	EOXCOR	EOX Live Correlations
129	ICEENDEX	ICE Energy Derivatives Exchange
130	KCBOT_GBX	Kansas City Board of Trade (GLOBEX)
131	MGE_GBX	Minneapolis Grain Exchange (GLOBEX)
132	BLOOMBERG	Bloomberg Indices
133	ELSPOT	Nord Pool Spot
134	N2EX	NASDAQ OMX-Nord Pool
135	ICEEA	International Commodities Exchange European Agriculture
136	CMEUR	Chicage Mercantile Exchange Europe Ltd
137	COMM3	Commodity 3 Exchange
138	JACOBSEN	The Jacobsen
139	NFX	NASDAQ OMX Futures
140	SGXAC	SGX Asia Clear
141	PJMISO	Pa-Nj-Md Independent System Operator
142	NYISO	New York Independent System Operator
143	NEISO	New England Independent System Operator
144	MWISO	Mid West Independent System Operator
145	SPISO	SW Power Pool Independent System Operator
146	CAISO	California Independent System Operator
147	ERCOT	ERCOT Independent System Operator
148	ABISO	Alberta Independent System Operator
149	ONISO	Ontario Independent System Operator
150	MERCURY	ISE Mercury Options Exchange
151	DCE	Dalian Commodity Exchange
152	ZCE	Zengchou Commodity Exchange
153	IEX	Investors Exchange LLC
154	MCX	Multi Commodity Exchange of India Limited
155	NCDEX	National Commodity Exchange of India Limited

156	PEARL	MIAX PEARL Options exchange
157	CTS	CTS System
158	LSEI	London Stock Exchange International
159	UNKNOWN	Unknown Market

* to retireve above table use `QuantTools:::.get_iqfeed_markets_info()`

Trade Conditions

Condition Code	Short Name	Description
01	REGULAR	Normal Trade
02	ACQ	Acquisition
03	CASHM	Cash Only Market
04	BUNCHED	Bunched Trade
05	AVGPRI	Average Price Trade
06	CASH	Cash Trade (same day clearing)
07	DIST	Distribution
08	NEXTDAY	Next Day Market
09	BURSTBSKT	Burst Basket Execution
0A	BUNCHEDSOLD	Bunched Sold Trade
0B	ORDETAIL	Opening/Reopening Trade Detail
0C	INTERDAY	Intraday Trade Detail
0D	BSKTONCLOSE	Basket Index on Close
0E	RULE127	Rule - 127 Trade NYSE
0F	RULE155	Rule - 155 Trade AMEX
10	SOLDLAST	Sold Last (late reporting)
11	NEXTDAYCLR	Next Day Clearing
12	LATEREP	Opened - Late Report of Opening Trade (in or out of sequence)
13	PRP	Prior Reference Price
14	SELLER	Seller
15	SPLIT	Split Trade
16	RSVD	(Reserved)
17	FORMT	Form-T Trade
18	CSTMBSKTX	Custom Basket Cross
19	SOLDOSEQ	Sold Out of Sequence
1A	CANC	Cancelled Previous Transaction
1B	CANCLAST	Cancelled Last Transaction
1C	CANCOPEN	Cancelled Open Transaction
1D	CANCONLY	Cancelled Only Transaction
1E	OPEN	Late Report of Opening Trade - out of sequence
1F	OPNL	Late Report of Opening Trade - in correct sequence
20	AUTO	Transaction Executed Electronically
21	HALT	Halt
22	DELAYED	Delayed
23	NON_BOARDLOT	NON_BOARDLOT
24	POSIT	POSIT
25	REOP	Reopen After Halt
26	AJST	Contract Adjustment for Stock Dividend - Split - etc.

27	SPRD	Spread - Trade in Two Options in the Same Class (a buy and a sell in the same
28	STDL	Straddle - Trade in Two Options in the Same Class (a buy and a sell in a put ar
29	STPD	Follow a Non-stopped Trade
2A	CSTP	Cancel Stopped Transaction
2B	BWRT	Option Portion of a Buy/Write
2C	CMBO	Combo - Trade in Two Options in the Same Options Class (a buy and a sell in
2D	UNSPEC	Unspecified
2E	MC_OFCLCLOSE	Market Center Official Closing Price
2F	STPD_REGULAR	Stopped Stock - Regular Trade
30	STPD_SOLDLAST	Stopped Stock - Sold Last
31	STPD_SOLDOSEQ	Stopped Stock - Sold out of sequence
32	BASIS	Basis
33	VWAP	Volume-Weighted Average Price
34	STS	Special Trading Session
35	STT	Special Terms Trading
36	CONTINGENT	Contingent Order
37	INTERNALX	Internal Cross
38	MOC	Market On Close Trade
39	MC_OFCLOPEN	Market Center Official Opening Price
3A	FORTMTSOLDOSEQ	Form-T Sold Out of Sequence
3B	YELLOWFLAG	Yellow Flag
3C	AUTOEXEC	Auto Execution
3D	INTRMRK_SWEEP	Intramaket Sweep
3E	DERIVPRI	Derivately Priced
3F	REOPNING	Re-Opeing Prints
40	CLSING	Closing Prints
41	CAP_ELCTN	CAP (Conversion and Parity) election trade
42	CROSS_TRADE	Cross Trade
43	PRICE_VAR	Price Variation
44	STKOPT_TRADE	Stock-Option Trade
45	SPIM	stopped at price that did not constitute a Trade-Through
46	BNMT	Benchmark Trade
47	TTEXEMPT	Transaction is Trade Through Exempt
48	LATE	Late Market
49	XCHG_PHYSICAL	Exchange for Physical
4A	CABINET	Cabinet
4B	DIFFERENTIAL	Differential
4C	HIT	Hit
4D	IMPLIED	Implied
4E	LG_ORDER	Large Order
4F	SM_ORDER	Small Order
50	MATCH	Match/Cross Trade
51	NOMINAL	Nominal
52	OPTION_EX	Option Exercise
53	PERCENTAGE	Percentage
54	AUTOQUOTE	Auto Quotes
55	INDICATIVE	Indicative
56	TAKE	Take

57	NOMINAL_CABINET	Nominal Cabinet
58	CHNG_TRANSACTION	Changing Transaction
59	CHNG_TRANS_CAB	Changing Transaction Cabinet
5A	FAST	Fast Market (ssfutures)
5B	NOMINAL_UPDATE	Nominal Update
5C	INACTIVE	Inactive - Nominal - No Trade
5D	DELTA	Last Trade with Delta Exchange
5E	ERRATIC	Erratic
5F	RISK_FACTOR	Risk Factor
60	OPT_ADDON	Short Option Add-On
61	VOLATILITY	Volatility Trade
62	SPD_RPT	Spread Reporting
63	VOL_ADJ	Volume Adjustment
64	BLANK	Blank out associated price
65	SOLDLATE	Late report of transaction - in correct sequence
66	BLKT	Block Trade
67	EXPH	Exchange Future for Physical
68	SPECIALIST_A	Ask from specialist Book
69	SPECIALIST_B	Bid from specialist Book
6A	SPECIALIST_BA	Both Bid and Ask from Specialist Book
6B	ROTATION	Rotation
6C	HOLIDAY	Holiday
6D	PREOPENING	Pre Opening
6E	POST_FULL	Post Full
6F	POST_RESTRICTED	Post Restricted
70	CLOSING_AUCTION	Closing Auction
71	BATCH	Batch
72	TRADING	Trading
73	OFFICIAL	Official Bid/Ask price
74	UNOFFICIAL	Unofficial Bid/Ask price
75	MIDPRICE	Midprice last
76	FLOOR	Floor B/A price
77	CLOSE	Closing Price
78	HIGH	End of Session High Price
79	LOW	End of Session Low Price
7A	BACKWARDATION	Backwardation - immediate delivery costing more than future delivery
7B	CONTANGO	Contango - future delivery costing more than immediate delivery
7C	RF_SETTLEMENT	Refined Fuel Spot Settlement
7D	RF_RESERVED1	Refined Fuel Spot Reserved - 1
7E	RF_RESERVED2	Refined Fuel Spot Reserved - 2
7F	RF_RESERVED3	Refined Fuel Spot Reserved - 3
80	RF_RESERVED4	Refined Fuel Spot Reserved - 4
81	YIELD	Yield Price
82	BASIS_HIGH	Current Basis High Value
83	BASIS_LOW	Current Bases Low Value
84	UNCLEAR	bid or offer price is unclear
85	OTC	Over the counter trade
86	MS	Trade entered by Market Supervision

87	ODDLOT	Odd lot trade
88	CORRCSLDLAST	Corrected Consolidated last
89	QUALCONT	Qualified Contingent Trade
8A	MC_OPEN	Market Center Opening Trade
8B	CONFIRMED	Confirmed
8C	OUTAGE	Outage
8D	SPRD_LEG	CME spread leg trade
8E	BNDL_SPRD_LEG	Final CME MDP3 trade from Trade Summary message that could not be Un-E
8F	LATECORR	LSE - Late Correction
90	CONTRA	LSE - Previous days contra
91	IF_TRANSFER	LSE - Inter-fund transfer
92	IF_CROSS	LSE - Inter-fund Cross
93	NEG_TRADE	LSE - Negotiated Trade
94	OTC_CANC	LSE - OTC Trade Cancellation
95	OTC_TRADE	LSE - OTC Trade
96	SI_LATECORR	LSE - SI Late Correction
97	SI_TRADE	LSE - SI Trade
98	AUCT_TRADE	LSE - Auctions (bulk;individual)
99	LATE	LSE - Late trade
9A	STRAT	LSE - Strategy vs. Strategy Trade trade
9B	INDICATIVE_AUCT	LSE - Indicative Auction Uncrossing Data

* to retireve above table use `QuantTools::::.get_iqfeed_trade_conditions_info()`

Examples

```
symbol = 'MSFT'
to = format( Sys.time() )
from = format( Sys.time() - as.difftime( 3, units = 'days' ) )
days = 10
# ticks
get_iqfeed_data( symbol, from, to, 'tick' )
# candles
get_iqfeed_data( symbol, from, to, '1min' )
# daily candles
get_iqfeed_data( symbol, from, to )
```

`iround` *Round numbers to specified interval*

Description

Round numbers to specified interval

Usage

```
iround(x, interval)
```

Arguments

x	numeric vector to be rounded
interval	the interval the values should be rounded towards

Value

A numeric vector with x rounded to the desired interval.

---

lapply_named *lapply which returns named list*

Description

lapply which returns named list

Usage

```
lapply_named(X, FUN, ...)
```

Arguments

X, FUN, ...	same as lapply arguments

---

lines_ohlc *Add candles to active time series plot*

Description

Add candles to active time series plot

Usage

```
lines_ohlc(x = 1:nrow(ohlc), ohlc, width = 0.3, candle.col.up = "blue",
  candle.col.dn = "red", ch = TRUE)
```

Arguments

x	location coordinates
ohlc	time_series data.frame or data.table object with 4 columns 'open','high','low','close'
width	width of candles body
candle.col.up, candle.col.dn	colors of up and down candles
ch	use Chinese style?

Details

Used in plot_ts internally.

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, empty_plot, hist_dt, lines_stacked_hist, multi_heatmap, plot_table, plot_ts

---

lines_stacked_hist *Add stacked histogram to active time series plot*

Description

Add stacked histogram to active time series plot

Usage

```
lines_stacked_hist(x = 1:nrow(data), data, width = "auto", col = "auto",
  ordered = TRUE)
```

Arguments

x	location coordinates
data	time_series data.frame or data.table object with 4 columns 'open','high','low','close'
width	width of histogram segment
col	colors of segments
ordered	should stacked bars be in order?

Details

Used in plot_ts internally.

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, empty_plot, hist_dt, lines_ohlc, multi_heatmap, plot_table, plot_ts

`lmerge` *Merge list of data.frames into data.table by key column*

Description

Merge list of data.frames into data.table by key column

Usage

```
lmerge(x, key, value, na.omit = T)
```

Arguments

`x`	named list of data.frames
`key`	column name to merge by
`value`	column name of value variable
`na.omit`	should leading NA values be omitted?

Examples

```
from = '1990-01-01'
to = '2016-08-30'

symbols = fread( '
               symbol, comment
               EFA, iShares MSCI EAFE Index Fund
               VTI, Vanguard Total Stock Market
               TLT, iShares 20+ Year Treasury Bond
               RWX, SPDR Dow Jones International RelEst
               IEV, iShares Europe
               IEF, iShares 7-10 Year Treasury Bond
               ICF, iShares Cohen & Steers Realty Maj.
               GLD, SPDR Gold Shares
               EWJ, iShares MSCI Japan
               EEM, iShares MSCI Emerging Markets
               DBC, PowerShares DB Commodity Tracking' )

# download historical market data
prices_list = lapply_named( symbols$'symbol', get_yahoo_data, from, to )

# table of close prices
prices = lmerge( prices_list, 'date' , 'close' )

# calculate returns and performance
dates = prices[, date ]
prices[, date := NULL ]
returns = lapply( prices, returns ) %>% setDT
performance = lapply( returns + 1, cumprod ) %>% setDT
```

```
# plot historical values
plot_ts( data.table( dates, returns ), legend = 'topleft' )
plot_ts( data.table( dates, prices ), legend = 'topleft' )
plot_ts( data.table( dates, performance ), legend = 'topleft' )
```

`multi_heatmap` *Multi Dimensional Heat Map*

Description

Multi Dimensional Heat Map

Usage

```
multi_heatmap(x, pars, value, col_neg = c("darkblue", "lightblue"),
  col_pos = c("yellow", "darkgreen"), peak_value = x[, max(abs(get(value)),
  na.rm = T)])
```

Arguments

`x`	`data.table` object
`pars`	names of parameters. Parameters combinations must be unique. To specify x and y axes use `list( x = ...,y = ... )`.
`value`	name of value parameter
`col_pos, col_neg`	used to generate gradient
`peak_value`	normalization value

Details

Plots multi dimensional heatmap. Axes drawn automatically by layers. Inner axes are most frequent and outer axes are less frequent.

See Also

Other graphical functions: `add_last_values`, `add_legend`, `distinct_colors`, `empty_plot`, `hist_dt`, `lines_ohlc`, `lines_stacked_hist`, `plot_table`, `plot_ts`

na_locf *Last Observation Carried Forward*

Description

Last Observation Carried Forward

Usage

```
na_locf(x, na = NA)
```

Arguments

x	list or vector to roll through
na	leading NA substitution

Order *C++ Order class*

Description

C++ class documentation

Arguments

side	BUY or SELL
type	LIMIT, MARKET, STOP, TRAIL
price	limit order price level, ignored for market orders
comment	arbitrary comment
idTrade	trade id for grouping multiple orders into trades

Usage

```
Order( OrderSide side,OrderType type,double price,std::string comment,int idTrade )
```

Public Members and Methods

Name	Return Type	Description
isNew()	bool	order is new or just sent to exchange?
isRegistered()	bool	placement confirmation received from exchange?
isCancelling()	bool	cancel request sent to exchange?
isCancelled()	bool	cancel confirmation received from exchange?
isExecuted()	bool	execution confirmation received from exchange?
isBuy?	bool	buy order?

isSell?	bool	sell order?
isLimit?	bool	limit order?
isMarket?	bool	market order?
GetTradeId()	int	trade id for grouping multiple orders into trades
GetExecutionPrice()	double	execution price, price for limit order and market price for market order
GetExecutionTime()	double	execution time
GetProcessedTime()	double	processed time
GetState()	OrderState	order state
comment	std::string	arbitrary comment, useful to identify order when analyzing backtest results
onExecuted	std::function	called when execution confirmation received from exchange
onCancelled	std::function	called when cancellation confirmation received from exchange
onRegistered	std::function	called when placement confirmation received from exchange
onCancelFailed	std::function	called when execution confirmation received from exchange but order was about
Cancel()	void	sends cancel request to exchange if state is REGISTERED and type is LIMIT

See Also

Other backtesting classes: Candle, Cost, Indicator, Processor, Tick

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

---

plot_dts *Plot data.table time series*

Description

Plot data.table time series

Methods

plot_dts Add data to be plotted.

$lines Add lines with following arguments:

names	vector of column names to plot
labels	vector of labels if different from column names
type	vector or single value, see lines
lty,pch,col,lwd,lend	vector or single value, see par
bg	vector or single value, see points

$candles Add candles with following arguments:

ohlc	vector of open, high, low and close names
timeframe	candle timeframe in minutes for intraday candles

position	relative to time position only 'end' supported
type	'barchart' or 'candlestick'
gap	gap between candles in fraction of width
mono	should all candles have same color?
col,col_up,col_flat,col_down	colors

xlim	vector of length two to limit plot area horizontally
ylim	vector of length two to limit plot area vertically
tlim	date or time vector of length two
time_range	intraday time limit in format 'H:M:S/H:M:S'

$limits

$style Change default plot options. Available options are:

grid		
minute	list(col,lty)	minute vertical gridline color and line type
hour	list(col,lty)	hour vertical gridline color and line type
day	list(col,lty)	day vertical gridline color and line type
month	list(col,lty)	month vertical gridline color and line type
year	list(col,lty)	year vertical gridline color and line type
zero	list(col,lty)	zero horizontal gridline color and line type
time		
grid	logical	should vertical gridlines be plotted?
resolution	character	auto, minute, hour, day, month, year or years
round	numeric	time axis rounding in minutes
visible	logical	should time axis be plotted?
value		
grid	logical	should horizontal gridlines be plotted?
last	logical	should last values be plotted?
log	logical	should y axis be in logarithmic scale?
visible	logical	should y axis be plotted?
candle		
auto	logical	shoud candles be automatically detected and plotted?
col	list(mono,up,flat,down)	colors
gap	numeric	gap between candles in fraction of width
mono	logical	should all candles have same color?
position	character	relative to time position only 'end' supported
type	character	'candlestick' or 'barchart'
line		
auto	logical	shoud lines be automatically detected and plotted?
legend		
col	list(background,frame)	colors
horizontal	logical	should legend be horizontal?
inset	numeric	see legend
position	character	see legend

visible	logical	should legend be plotted?

`plot_table` *Plot data.table as table*

Description

Plot data.table as table

Usage

```
plot_table(dt, transpose = F, justify = c("middle", "left", "right"), ...)
```

Arguments

dt	data.table
transpose	should table be transposed?
justify	'middle','left','right'
...	further graphical parameters as in par

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, empty_plot, hist_dt, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_ts

---

`plot_ts` *Plot time series !PLEASE USE plot_dts!*

Description

Plot time series !PLEASE USE plot_dts!

Usage

```
plot_ts(dt, type = "auto", col = "auto", lty = par("lty"),
  lwd = par("lwd"), pch = par("pch"), legend = c("topright", "topleft",
  "bottomright", "bottomleft", "n"), last_values = TRUE, main = "",
  ylim = "auto", xlim = "auto", time_range = "auto",
  resolution = "auto", log = par("ylog"), mar = par("mar"),
  xaxt = par("xaxt"), add = par("new"))

t_to_x(t)
```

Arguments

dt	data.table with date/time index represented by first column. If OHLC detected then only candles plotted. Use lines for the rest of data
type	type vector or single value. Same as in plot but 'candle' supports. Default is 'l'. 'h' triggers adding zero to plot range
col	color vector or single value. Default is 'auto' so colors generated automatically
lty, lwd, pch	parameters vectors or single values. Same as in plot
legend	position of plot legend. Supported positions are 'topright','topleft','bottomright','bottomleft or 'n' to hide legend
last_values	whether to add last values marks to the right of the plot. If vector specified marks added only for columns specified in vector
main	title of the plot. Default is ''
ylim	y range of data to plot
xlim	x range of data to plot
time_range	time range in format 'HH:MM:SS/HH:MM:SS'
resolution	frequency of time marks on time axis. Supported resolutions are 'auto','minute','hour','day','mon Default is 'auto'
log	should y axis be in logarithmic scale?
mar	same as in par
xaxt	same as in par
add	add to existing plot?
t	date/time vector to be converted to plot x coordinates

Details

Plots time series each represented by columns of times_series on single plot.
As for OHLC series, only one can be plotted and should be passed as times_series with 4 columns 'open','high','low','close'.

See Also

Other graphical functions: add_last_values, add_legend, distinct_colors, empty_plot, hist_dt, lines_ohlc, lines_stacked_hist, multi_heatmap, plot_table

Examples

```
data( ticks )

time_series = to_candles( ticks, 60 * 10 )

plot_ts( time_series[ time %bw% '2016-05-13', list( time, open, high, low, close ) ] )
plot_ts( time_series[ time %bw% '2016-05-13', list( time, volume = volume / 1e6 ) ] , type = 'h' )
plot_ts( time_series[ time %bw% '2016-05', list( time, close ) ] )
```

```
plot_ts( time_series[ , list( time, close ) ] )



mar = par( 'mar' )
par( mar = c( 0, 4, 0, 4 ), xaxt = 'n' )
layout ( matrix( 1:(3 + 2) ), heights = c( 1, 4, 2, 2, 1 ) )
  empty_plot()
  plot_ts( time_series[ , list( time, open, high, low, close ) ] )
  plot_ts( time_series[ , list( time, close ) ] )
  par( xaxt = 's' )
  plot_ts( time_series[ , list( time, volume = volume / 1e6 ) ], type = 'h' )
  empty_plot()
par( mar = mar )
layout( matrix(1) )
```

`Processor` *C++ Processor class*

Description

C++ class documentation

Arguments

`timeFrame`	candle timeframe in seconds
`latencySend, latencyReceive`	latency in seconds

Usage

```
Processor( int timeFrame,double latencySend,double latencyReceive )
```

Public Members and Methods

Name	Return Type	Description
`onCandle( Candle candle )`	`std::function`	called on new
`onTick( Tick tick )`	`std::function`	called on new
`onMarketOpen()`	`std::function`	called on tradi
`onMarketClose()`	`std::function`	called on tradi
`onIntervalOpen()`	`std::function`	called on inter
`onIntervalClose()`	`std::function`	called on inter
`Feed( Tick tick )`	`void`	process by ind
`Feed( Rcpp::DataFrame ticks )`	`void`	batch process,
`SendOrder( Order* order )`	`void`	send order to e
`SetCost( Cost cost )`	`void`	set trading cos

SetCost(Rcpp::List cost)	void	see 'cost' in 'C
SetStop(Rcpp::List stop)	void	see 'stop' in 'C
SetStartTradingTime(double t)	void	see 'trade_star
SetLatencyReceive(double x)	void	see 'latency_re
SetLatencySend(double x)	void	see 'latency_se
SetLatency(double x)	void	see 'latency' in
SetTradingHours(double start, double end)	void	see 'trading_ho
SetPriceStep(double priceStep)	void	see 'price_step
SetExecutionType(ExecutionType executionType)	void	see 'execution_
SetExecutionType(std::string executionType)	void	see 'execution_
SetIntervals(std::vector<double> starts, std::vector<double> ends)	void	see 'intervals'
AllowLimitToHitMarket()	void	see 'allow_lim
AllowExactStop()	void	see 'allow_exa
SetOptions(Rcpp::List options)	void	see 'Options' s
StopTrading()	void	if called tradin
CanTrade()	bool	check if tradin
IsTradingHoursSet()	bool	check if tradin
CancelOrders()	void	cancel active o
GetCandle()	Candle	get current can
GetPosition()	int	total executed
GetPositionPlanned()	int	total number o
GetMarketValue()	double	total portfolio
GetCandles()	Rcpp::List	candles history
GetOrders()	Rcpp::List	orders history,
GetTrades()	Rcpp::List	trades history,
GetSummary()	Rcpp::List	trades summar
GetOnCandleMarketValueHistory()	std::vector<double>	vector of portf
GetOnCandleDrawDownHistory()	std::vector<double>	vector of portf
GetOnDayClosePerformanceHistory()	Rcpp::List	daily performa
Reset()	void	resets to initial

Execution Model

System sends new order and after `latencySend` seconds it reaches exchange. System receives confirmation of order placement `latencyReceive` seconds later. When execution conditions met on exchange - order is executed and system receives execution confirmation `latencyReceive` seconds later.

When system sends cancel request to exchange and after `latencySend` seconds when exchange receives cancel request if order is not executed yet it is cancelled and cancellation confirmation is received by system after `latencyReceive` seconds.

Two execution types supported `trade`(default) and `bbo`. `trade` type processes orders using tick `prices` and `bbo` processes orders using preceding tick `bid` and `ask` values. Market orders in `bbo` mode executed at worst price: at `bid` for sells and at `ask` for buys, in `trade` mode at current tick `price`. Buy limit orders executed when `ask` goes under order price and sell orders executed when `bid` goes above order price. In case limit order is placed in the market it is executed as market order if `allow_limit_to_hit_market` set to `TRUE` (default is `FALSE`).

Ticks

Ticks must be a data.frame/data.table with at least the following columns:

Name	Description
time	time
price	price
volume	volume

tick id is ticks row number.

Candles

Candles returned as data.table with the following columns:

Name	Description
time	time when formed
open	first tick price
high	maximum tick price
low	minimum tick price
close	last tick price
volume	total volume traded
id	tick id when formed (first tick after time formed)

Orders

Orders returned as data.table with the following columns:

Name	Description
id_trade	trade id
id_sent	tick id when order was sent to exchange
id_processed	tick id when enter order execution or cancelled confirmation was received (first tick after `time_processed`
time_sent	time when order was sent to exchange
time_processed	time when order execution or cancelled confirmation was received
price_init	initial price
price_exec	execution price
side	`buy/sell`
type	`limit/market/stop/trail`
state	`new/registered/executed/cancelling/cancelled`
comment	comment

Trades

Two orders are combined into trade by trade id. The first and the second orders are called enter and exit respectively.
Trade side is long if enter order is buy and short if enter order is sell.
Orders must be buy and sell only. Two buys or two sells not allowed. Trade can be

- `new` when order to open trade is just placed
- `opened` when trade is not closed yet
- `closed` when trade is flat.

Trades returned as data.table with the following columns:

Name	Description
id_trade	trade id
id_sent	tick id when enter order was sent to exchange
id_enter	tick id when enter order execution confirmation was received (first tick after enter `time_executed`)
id_exit	tick id when exit order execution confirmation was received (first tick after exit `time_executed`)
time_sent	time when enter order sent to exchange
time_enter	time when enter order execution confirmation was received
time_exit	time when exit order execution confirmation was received
side	side `long`/`short`
price_enter	enter order execution price
price_exit	exit order execution price
pnl	trade pnl net
mtm	mark-to-market
mtm_min	min mark-to-market
mtm_max	max mark-to-market
cost	absolute trading cost
pnl_rel	trade pnl net in basis points
mtm_rel	mark-to-market in basis points
mtm_min_rel	min mark-to-market in basis points
mtm_max_rel	max mark-to-market in basis points
cost_rel	relative trading cost in basis points
state	`new`/`opened`/`closed`

Summary

Back test summary statistics:

Name	Description
from	first tick time
to	last tick time
days_tested	number of trading days tested
days_traded	number of trading days traded (at least one order was executed)
n_per_day	number of trades per day
n	number of trades
n_long	number of long trades
n_short	number of short trades
n_win	number of winning trades
n_loss	number of loosing trades
pct_win	percent of winning trades
pct_loss	percent of loosing trades

avg_win	average winning trade in basis points
avg_loss	average loosing trade in basis points
avg_pnl	average trade pnl in basis points
win	total won in percent
loss	total lost in percent
pnl	total pnl in percent
max_dd	maximum drawdown in percent
max_dd_start	time the maximum drawdown started
max_dd_end	time the maximum drawdown recovered
max_dd_length	number of calendar days in the maximum drawdown period
sharpe	annualized Sharpe ratio calulated on daily returns
sortino	annualized Sortino ratio calulated on daily returns
r_squared	R Squared calulated on daily PnL values
avg_dd	average drawdown calulated on daily drawdown history

Daily Performance

Back test daily performance history:

Name	Description
date	date
return	return
pnl	cumulative pnl
drawdown	drawdown
n_per_day	number of closed trades
avg_pnl	average trade pnl

Options

List of following elements. All options are optional.

cost list or data.table with items identical to Cost C++ class.
E.g. if set to `data.table( tradeAbs = -0.01,shortRel = -0.05 / 360 )` means you pay -$0.01 per executed order and -5% p.a. overnight short.

stop list or data.table with at least one item:

drawdown Trading stops when drawdown exceeds set value. E.g. if set to -0.02 then when drawdown exceeds 2% trading stops.

loss Trading stops when market value (P&L) is lower set value. E.g. if set to -0.05 then when market value (P&L) is lower than -5% trading stops.
If stop rule triggered no orders sent to exchange and opened trades closed by market orders.

trade_start POSIXct timestamp. All orders ignored until specified time. Useful to 'warm-up' strategy.

latency_send, latency_receive, latency numeric value. Latency can be set by send/receive or overall. 'latency' sets send and receive latency as `x / 2`. See 'Execution Model' section.

trading_hours numeric vector of length two. Sets trading hours start and end according to formula: `hours + minutes / 60 + seconds / 3600`.

If set onMarketOpen, onMarketClose events are executed at corresponding times.
E.g. if set to c(10.25,17.5) means onMarketOpen event called every day at '10:15' and onMarketClose event called every day at '17:30'.
For convenience IsTradingHoursSet() method can be used to check wether trading hours are set.

allow_limit_to_hit_market if TRUE, limit order execution price set to market price if executed on same tick as registered.

allow_exact_stop if TRUE, stop order executed at set price.

price_step if positive, limit order init price rounded to price_step down for buy orders and up for sell orders before placement. if negative, limit order init price rounded to price_step up for buy orders and down for sell orders before placement.

execution_type trade or bbo.

intervals sorted multi row data.table with POSIXct timestamps columns start,end. Represents time intervals. At time start onIntervalOpen called and at time end onIntervalClose called.

See Also

Other backtesting classes: Candle, Cost, Indicator, Order, Tick

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

Examples

```
#####################################
## Simple Moving Averages Crossover ##
#####################################

# load tick data
data( 'ticks' )

# define strategy
strategy_source = system.file( package = 'QuantTools', 'examples/sma_crossover.cpp' )
# compile strategy
Rcpp::sourceCpp( strategy_source )

# set strategy parameters
parameters = data.table(
  period_fast = 50,
  period_slow = 30,
  timeframe   = 60
)

# set options, see 'Options' section
options = list(
  cost    = list( tradeAbs = -0.01 ),
  latency = 0.1 # 100 milliseconds
)
```

```
# run test
test_summary = sma_crossover( ticks, parameters, options, fast = TRUE )
print( test_summary )

# run test
test = sma_crossover( ticks, parameters, options, fast = FALSE )

# plot result
indicators = plot_dts(
test$indicators,
test$orders[ side == 'buy' , .( time_processed, buy  = price_exec ) ],
test$orders[ side == 'sell', .( time_processed, sell = price_exec ) ] )$
lines( c( 'sma_fast', 'sma_slow' ) )$
lines( c( 'buy', 'sell' ), type = 'p', pch = c( 24, 25 ), col = c( 'blue', 'red' ) )

performance = plot_dts( test$indicators[, .( time, pnl = pnl * 100, drawdown = drawdown * 100 ) ] )$
lines( c( 'pnl', 'drawdown' ), c( '% pnl', '% drawdown' ), col = c( 'darkolivegreen', 'darkred' ) )

interval = '2016-01-19 12/13'
par( mfrow = c( 2, 1 ), oma = c( 5, 4, 2, 4 ) + 0.1, mar = c( 0, 0, 0, 0 ) )
indicators $limits( tlim = interval )$style( time = list( visible = FALSE ) )
performance$limits( tlim = interval )
title( 'Simple Moving Averages Crossover', outer = TRUE )
par( mfrow = c( 1, 1 ), oma = c( 0, 0, 0, 0 ), mar = c( 5, 4, 4, 2 ) + 0.1 )




######################
## Bollinger Bands ##
######################

# load tick data
data( 'ticks' )

# define strategy
strategy_source = system.file( package = 'QuantTools', 'examples/bbands.cpp' )
# compile strategy
Rcpp::sourceCpp( strategy_source )

# set strategy parameters
parameters = data.table(
  n         = 100,
  k         = 0.5,
  timeframe = 60
)

# set options, see 'Options' section
options = list(
  cost    = list( tradeAbs = -0.01 ),
  latency = 0.1 # 100 milliseconds
)
```

```
# run test
test_summary = bbands( ticks, parameters, options, fast = TRUE )
print( test_summary )

# run test
test = bbands( ticks, parameters, options, fast = FALSE )

# plot result
indicators = plot_dts(
test$indicators,
test$orders[ side == 'buy' , .( time_processed, buy  = price_exec ) ],
test$orders[ side == 'sell', .( time_processed, sell = price_exec ) ] )$
lines( c( 'lower', 'sma', 'upper' ) )$
lines( c( 'buy', 'sell' ), type = 'p', pch = c( 24, 25 ), col = c( 'blue', 'red' ) )

performance = plot_dts( test$indicators[, .( time, pnl = pnl * 100, drawdown = drawdown * 100 ) ] )$
lines( c( 'pnl', 'drawdown' ), c( '% pnl', '% drawdown' ), col = c( 'darkolivegreen', 'darkred' ) )

interval = '2016-01-19 12/13'
par( mfrow = c( 2, 1 ), oma = c( 5, 4, 2, 4 ) + 0.1, mar = c( 0, 0, 0, 0 ) )
indicators $limits( tlim = interval )$style( time = list( visible = FALSE ) )
performance$limits( tlim = interval )
title( 'Bollinger Bands', outer = TRUE )
par( mfrow = c( 1, 1 ), oma = c( 0, 0, 0, 0 ), mar = c( 5, 4, 4, 2 ) + 0.1 )




##################################
## Bollinger Bands Market Maker ##
##################################

# load tick data
data( 'ticks' )

# define strategy
strategy_source = system.file( package = 'QuantTools', 'examples/bbands_market_maker.cpp' )
# compile strategy
Rcpp::sourceCpp( strategy_source )

# set strategy parameters
parameters = data.table(
  n         = 100,
  k         = 0.5,
  timeframe = 60
)

# set options, see 'Options' section
options = list(
  cost    = list( tradeAbs = -0.01 ),
  latency = 0.1, # 100 milliseconds
  allow_limit_to_hit_market = TRUE
```

```
)

# run test
test_summary = bbands_market_maker( ticks, parameters, options, fast = TRUE )
print( test_summary )

# run test
test = bbands_market_maker( ticks, parameters, options, fast = FALSE )

# plot result
indicators = plot_dts(
test$indicators,
test$orders[ side == 'buy' , .( time_processed, buy  = price_exec ) ],
test$orders[ side == 'sell', .( time_processed, sell = price_exec ) ] )$
lines( c( 'lower', 'sma', 'upper' ) )$
lines( c( 'buy', 'sell' ), type = 'p', pch = c( 24, 25 ), col = c( 'blue', 'red' ) )

performance = plot_dts( test$indicators[, .( time, pnl = pnl * 100, drawdown = drawdown * 100 ) ] )$
lines( c( 'pnl', 'drawdown' ), c( '% pnl', '% drawdown' ), col = c( 'darkolivegreen', 'darkred' ) )

interval = '2016-01-19 12/13'
par( mfrow = c( 2, 1 ), oma = c( 5, 4, 2, 4 ) + 0.1, mar = c( 0, 0, 0, 0 ) )
indicators $limits( tlim = interval )$style( time = list( visible = FALSE ) )
performance$limits( tlim = interval )
title( 'Bollinger Bands On Limit Orders', outer = TRUE )
par( mfrow = c( 1, 1 ), oma = c( 0, 0, 0, 0 ), mar = c( 5, 4, 4, 2 ) + 0.1 )
```

`returns` *Calculate returns*

Description

Calculate returns

Usage

```
returns(x, type = "r", n = 1)
```

Arguments

x	numeric vector
type	'r' = x[t] / x[t-n] -1 , 'l' = ln(x[t] / x[t-n])
n	lookback

Value

Vector of same length as x with absent returns converted to 0 for relative and 1 for logarithmic.

RollLinReg *C++ Rolling Linear Regression class*

Description

C++ class documentation

Arguments

n	indicator period

Details

R functions roll_lm.

Usage

```
RollLinReg( int n )
```

Public Members and Methods

Name	Return Type	Description
Add(InputType value)	void	update indicator
Reset()	void	reset to initial state
IsFormed()	bool	is indicator value valid?
GetAlphaHistory()	std::vector< double >	return alpha history
GetBetaHistory()	std::vector< double >	return beta history
GetRHistory()	std::vector< double >	return r history
GetRSquaredHistory()	std::vector< double >	return r squared history
GetValue()	LinRegCoeffs	has members double alpha, beta, r, rSquared
GetHistory()	List	return values history data.table with columns alpha, beta, r, r.squ

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

RollPercentRank *C++ Rolling Percent Rank class*

Description

C++ class documentation

Arguments

n	indicator period

Usage

```
RollPercentRank( int n )
```

Public Members and Methods

Name	Return Type	Description
Add(InputType value)	void	update indicator
Reset()	void	reset to initial state
IsFormed()	bool	is indicator value valid?
GetHistory()	std::vector<double>	history vector

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

RollRange *C++ Rolling Range / Quantile class*

Description

C++ class documentation

Arguments

n	indicator period
p	probability value [0,1]

Details

R functions roll_range, roll_quantile, roll_min, roll_max.

Usage

```
RollRange( int n,double p = 0.5 )
```

Public Members and Methods

Name	Return Type	Description
`Add( InputType value )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetValue()`	`Range`	has members `double min, max, quantile`
`GetMinHistory()`	`std::vector< double >`	return min history
`GetMaxHistory()`	`std::vector< double >`	return max history
`GetQuantileHistory()`	`std::vector< double >`	return quantile history
`GetHistory()`	`List`	return values history data.table with columns `min, max`

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollSd, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

---

RollSd *C++ Rolling Standard Deviation class*

Description

C++ class documentation

Arguments

n	indicator period

Usage

```
RollSd( int n )
```

Public Members and Methods

Name	Return Type	Description
`Add( InputType value )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetHistory()`	`std::vector<double>`	factor vector with levels `UP, DN`

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollVolumeProfile, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollVolumeProfile, Rsi, Sma, Stochastic, Tick

---

RollVolumeProfile *C++ Rolling Volume Profile class*

Description

C++ class documentation

Arguments

`timeFrame`	indicator period in seconds, when to apply alpha correction
`step`	price round off value, bar width
`alpha`	multiplication coefficient must be between (0,1]
`cut`	threshold volume when to delete bar

Details

R functions roll_volume_profile.

Usage

```
RollVolumeProfile( int timeFrame,double step,double alpha,double cut )
```

Public Members and Methods

Name	Return Type	Description
`Add( Tick tick )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetValue()`	`std::map<double,double>`	histogram where first is price and second is volume
`GetHistory()`	`List`	return values history data.table with columns `time, profile` where profi

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, Rsi, Sma, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, Rsi, Sma, Stochastic, Tick

---

`roll_futures` *Combine multiple futures market data into continuous contract*

Description

Combine multiple futures market data into continuous contract

Usage

```
roll_futures(prices_by_contract, days_before_expiry)
```

Arguments

`prices_by_contract`	list of data.tables with futures market data
`days_before_expiry`	number of dates before expiration to roll

`roll_lm` *Rolling Linear Regression*

Description

Rolling linear regression calculates regression coefficients over n past paired values.
Others return numeric vector

Usage

```
roll_lm(x, y, n)

roll_correlation(x, y, n)
```

Arguments

`x, y`	numeric vectors
`n`	window size

Value

roll_lm returns data.table with columns `alpha,beta,r,r.squared`

See Also

Other technical indicators: bbands, crossover, ema, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma, stochastic

---

roll_percent_rank *Rolling Percent Rank*

Description

Rolling percent rank normalizes values to a range from 0 to 100.

Usage

```
roll_percent_rank(x, n)
```

Arguments

x	numeric vector
n	window size

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_range, roll_sd, roll_volume_profile, rsi, sma, stochastic

---

roll_range *Rolling Range*

Description

Rolling range is minimum and maximum values over n past values. Can be used to identify price range.

Usage

```
roll_range(x, n)

roll_quantile(x, n, p)

roll_min(x, n)

roll_max(x, n)
```

Arguments

x	numeric vectors
n	window size
p	probability value [0,1]

Value

roll_range returns data.table with columns min,max
others return numeric vector

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_percent_rank, roll_sd, roll_volume_profile, rsi, sma, stochastic

---

roll_sd *Rolling Standard Deviation*

Description

Rolling standard deviation shows standard deviation over n past values.

Usage

```
roll_sd(x, n)
```

Arguments

x	numeric vector
n	window size

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_volume_profile, rsi, sma, stochastic

roll_sd_filter *Rolling Filter*

Description

Logical vector is returned. This function is useful to filter ticks. Finds consequent elements which absolute change is higher than k standard deviation of past n changes and mark them FALSE. If sequence length greater than m values become TRUE.

Usage

```
roll_sd_filter(x, n, k = 1, m = 10L)
```

Arguments

x	numeric vector
n	window size
k	number of standard deviations
m	number of consequent large returns to stop filtering out

roll_volume_profile *Rolling Volume Profile*

Description

This indicator is not common. Volume profile is the distribution of volume over price. It is formed tick by tick and partially forgets past values over time interval. When volume on any bar is lower than specified critical value the bar is cut.

Usage

```
roll_volume_profile(ticks, timeFrame, step, alpha, cut)
```

Arguments

ticks	read 'Ticks' section in Processor
timeFrame	indicator period in seconds, when to apply alpha correction
step	price round off value, bar width
alpha	multiplication coefficient must be between (0,1]
cut	threshold volume when to delete bar

Value

data.table with columns time,profile where profile is data.table with columns time,price,volume

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, rsi, sma, stochastic

---

`round_POSIXct` *Round POSIXct timestamps*

Description

Round POSIXct timestamps

Usage

```
round_POSIXct(x, n = 1, units = c("secs", "mins", "hours", "days"),
  method = round)

ceiling_POSIXct(x, n = 1, units = c("secs", "mins", "hours", "days"))

trunc_POSIXct(x, n = 1, units = c("secs", "mins", "hours", "days"))
```

Arguments

x	POSIXct vector
n	number of units to round off
units	to round off to
method	round method, see Round

Details

Rounds POSIXct vector with specified method.

`Rsi` *C++ Relative Strength Index class*

Description

C++ class documentation

Arguments

n	indicator period

Details

R function rsi.

Usage

```
Rsi( int n )
```

Public Members and Methods

Name	Return Type	Description
`Add( double value )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetValue()`	`double`	return value
`GetHistory()`	`std::vector<double>`	return values history

See Also

Other C++ indicators: `BBands`, `Crossover`, `Ema`, `Indicator`, `RollLinReg`, `RollPercentRank`, `RollRange`, `RollSd`, `RollVolumeProfile`, `Sma`, `Stochastic`

Other C++ classes: `BBands`, `Candle`, `Cost`, `Crossover`, `Ema`, `Indicator`, `Order`, `Processor`, `RollLinReg`, `RollPercentRank`, `RollRange`, `RollSd`, `RollVolumeProfile`, `Sma`, `Stochastic`, `Tick`

---

rsi *Relative Strength Index*

Description

Relative strength index aka RSI measures the velocity and magnitude of directional price movements.

Usage

```
rsi(x, n)
```

Arguments

`x`	numeric vectors
`n`	window size

See Also

Other technical indicators: `bbands`, `crossover`, `ema`, `roll_lm`, `roll_percent_rank`, `roll_range`, `roll_sd`, `roll_volume_profile`, `sma`, `stochastic`

`settings` *QuantTools settings*

Description

QuantTools settings

Usage

```
QuantTools_settings(settings = NULL)

QuantTools_settings_defaults()
```

Arguments

`settings`	named list of settings values or settings names vector

Details

Controls package settings.

List of available settings:

finam_storage	Finam local storage path
iqfeed_storage	IQFeed local storage path
moex_storage	MOEX local storage path
moex_data_url	MOEX data url
finam_storage_from	Finam storage first date
iqfeed_storage_from	IQFeed storage first date
moex_storage_from	MOEX storage first date
finam_symbols	Finam symbols to store
iqfeed_symbols	IQFeed symbols to store
iqfeed_port	IQFeed historical port number
iqfeed_host	IQFeed host
iqfeed_timeout	IQFeed connection timeout
iqfeed_buffer	IQFeed number of bytes buffer
iqfeed_verbose	IQFeed verbose internals?
temp_directory	temporary directory location

Examples

```
# list all settings
QuantTools_settings()

# set defaults
QuantTools_settings_defaults()
```

```
# change a setting
QuantTools_settings( list( iqfeed_verbose = TRUE ) )

# To make R remember your settings please add the following code
# to .Rprofile file stored in your home directory path.expand('~'):

suppressMessages( library( QuantTools ) )

QuantTools_settings( settings = list(
  iqfeed_storage = paste( path.expand('~') , 'Market Data', 'iqfeed', sep = '/' ),
  iqfeed_symbols = c( 'AAPL', '@ES#' ),
  iqfeed_storage_from = format( Sys.Date() - 3 )
) )
```

Sma *C++ Simple Moving Average class*

Description

C++ class documentation

Arguments

n	indicator period

Details

R function sma.

Usage

```
Sma( int n )
```

Public Members and Methods

Name	Return Type	Description
`Add( double value )`	`void`	update indicator
`GetValue()`	`double`	return value
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetHistory()`	`std::vector<double>`	return values history

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Stochastic

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Stochastic, Tick

---

`sma` *Simple Moving Average*

Description

Simple moving average also called SMA is the most popular indicator. It shows the average of n past values. Can be used for time series smoothing.

Usage

```
sma(x, n)
```

Arguments

`x`	numeric vectors
`n`	window size

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, stochastic

---

`Stochastic` *C++ Stochastic class*

Description

C++ class documentation

Arguments

`InputType`	`Tick` or `double`
`n`	indicator period
`nFast`	fast smooth
`nSlow`	slow smooth

Details

R function stochastic.

Usage

```
Stochastic< InputType >( int n,int nFast,int nSlow )
```

Public Members and Methods

Name	Return Type	Description
`Add( InputType value )`	`void`	update indicator
`Reset()`	`void`	reset to initial state
`IsFormed()`	`bool`	is indicator value valid?
`GetValue()`	`StochasticValue`	has members `double kFast, dFast, dSlow`
`GetKFastnHistory()`	`std::vector< double >`	return k fast history
`GetDFastHistory()`	`std::vector< double >`	return d fast history
`GetDSlowHistory()`	`std::vector< double >`	return d slow history
`GetHistory()`	`List`	return values history data.table with columns `k_fast, d_fast, d_sl`

See Also

Other C++ indicators: BBands, Crossover, Ema, Indicator, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma

Other C++ classes: BBands, Candle, Cost, Crossover, Ema, Indicator, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Sma, Tick

---

stochastic *Stochastic*

Description

Stochastic oscillator shows position of price in respect to its range over n past values.

Usage

```
stochastic(x, n, nFast, nSlow)
```

Arguments

x	`high,low,close` data.frame or numeric vector
n	window size
nFast	fast smooth
nSlow	slow smooth

Value

data.table with columns k_fast,d_fast,d_slow

See Also

Other technical indicators: bbands, crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma

---

store_market_data *Store historical market data*

Description

Store historical market data

Usage

```
store_finam_data(from = NULL, to = format(Sys.Date()), verbose = TRUE)

store_iqfeed_data(from = NULL, to = format(Sys.Date()), verbose = TRUE)

store_moex_data(from = NULL, to = format(Sys.Date()), verbose = TRUE)
```

Arguments

from, to	text dates in format "YYYY-mm-dd"
verbose	show progress?

Details

See example below.

Examples

```
## Finam data storage
settings = list(
  # set storage path, it is perfect to use Solid State Drive for data storage
  # it is no problem to move storage folder just don't forget to set new path in settings
  finam_storage = paste( path.expand('~') , 'Market Data', 'finam', sep = '/' ),
  # add some symbols
  finam_symbols = c( 'GAZP', 'SBER' ),
  # and set storage start date
  finam_storage_from = '2016-09-01'
)
QuantTools_settings( settings )
# now it is time to add some data into storage. You have three options here:
```

```
  # 1 update storage with data from last date available until today
  # it is very convenient to create a script with this function and
  # run it every time you need to update your storage
  store_finam_data()

  # 2 update storage with data from last date available until specified date
  store_finam_data( to = '2016-09-28' )

  # 3 update storage with data between from and to dates,
  # if data already present it will be overwritten
  store_finam_data( from = '2016-01-01', to = '2016-01-10' )

# set local = TRUE to load from just created local market data storage
get_finam_data( 'GAZP', '2016-09-01', '2016-09-28', 'tick', local = T )

## IQFeed data storage
settings = list(
  # set storage path, it is perfect to use Solid State Drive for data storage
  # it is no problem to move storage folder just don't forget to set new path in settings
  iqfeed_storage = paste( path.expand('~') , 'Market Data', 'iqfeed', sep = '/' ),
  # add some symbols
  iqfeed_symbols = c( 'AAPL', '@ES#' ),
  # and set storage start date
  iqfeed_storage_from = format( Sys.Date() - 3 )
)
QuantTools_settings( settings )
# now it is time to add some data into storage. You have three options here:

  # 1 update storage with data from last date available until today
  # it is very convenient to create a script with this function and
  # run it every time you need to update your storage
  store_iqfeed_data()

  # 2 update storage with data from last date available until specified date
  store_iqfeed_data( to = format( Sys.Date() ) )

  # 3 update storage with data between from and to dates,
  # if data already present it will be overwritten
  store_iqfeed_data( from = format( Sys.Date() - 3 ), to = format( Sys.Date() ) )

# set local = TRUE to load from just created local market data storage
get_iqfeed_data( 'AAPL', format( Sys.Date() - 3 ), format( Sys.Date() ), 'tick', local = T )

## MOEX data storage
settings = list(
  # set MOEX data url
  moex_data_url = 'url/to/moex/data',
  # set storage path, it is perfect to use Solid State Drive for data storage
  # it is no problem to move storage folder just don't forget to set new path in settings
  moex_storage = paste( path.expand('~') , 'Market Data', 'moex', sep = '/' ),
  # and set storage start date
  moex_storage_from = '2003-01-01'
```

```
)
QuantTools_settings( settings )
# now it is time to add some data into storage. You have three options here:

  # 1 update storage with data from last date available until today
  # it is very convenient to create a script with this function and
  # run it every time you need to update your storage
  store_moex_data()

  # 2 update storage with data from last date available until specified date
  store_moex_data( to = format( Sys.Date() ) )

  # 3 update storage with data between from and to dates,
  # if data already present it will be overwritten
  store_moex_data( from = format( Sys.Date() - 3 ), to = format( Sys.Date() ) )

# set local = TRUE to load from just created local market data storage
get_moex_futures_data( 'RIH9', '2009-01-01', '2009-02-01', 'tick', local = T )
```

`Tick` *C++ Tick class*

Description

C++ class documentation

Arguments

`id`	id
`time`	seconds since epoch
`price`	price
`volume`	volume
`bid`	best bid
`ask`	best offer
`system`	`true` ignore all except `time` and `id` value, default is `false`

Usage

`Tick{ int id,double time,double price,int volume,double bid,double ask,bool system }`

See Also

Other backtesting classes: `Candle`, `Cost`, `Indicator`, `Order`, `Processor`

Other C++ classes: `BBands`, `Candle`, `Cost`, `Crossover`, `Ema`, `Indicator`, `Order`, `Processor`, `RollLinReg`, `RollPercentRank`, `RollRange`, `RollSd`, `RollVolumeProfile`, `Rsi`, `Sma`, `Stochastic`

ticks *Example intraday tick data*

Description

Example intraday tick data

to_candles *Convert ticks to candles*

Description

Convert ticks to candles

Usage

```
to_candles(ticks, timeframe)
```

Arguments

ticks	read 'Ticks' section in Processor
timeframe	candle timeframe in seconds

Value

data.table with columns time,open,high,low,close,volume,id. Where id is row number of last tick in candle.
Note: last candle is always omitted.

to_ticks *Convert candles to ticks*

Description

Convert candles to ticks

Usage

```
to_ticks(x)
```

Arguments

x	candles, read 'Candles' in Processor

Details

Convert OHLCV candles to ticks using the following model. One candle is equivalent to four ticks (time,price,volume): (time -period,open,volume / 4); (time -period / 2,high,volume / 4); (time -period / 2,low,volume / 4); (time -period / 100,close,volume / 4). Assuming provided candles have frequent period (less than a minute) it is a good approximation for tick data which can be used to speed up back testing or if no raw tick data available.

Examples

```
data( ticks )
candles = to_candles( ticks, timeframe = 60 )
to_ticks( candles )
```

---

to_UTC *Convert time zone to 'UTC' without changing value*

Description

Convert time zone to 'UTC' without changing value

Usage

```
to_UTC(x)
```

Arguments

x	POSIXct vector

Examples

```
Sys.time()
to_UTC( Sys.time() )
```

wo *Select values in one vector not present in another*

Description

Select values in one vector not present in another

Usage

```
x %w/o% y
```

Arguments

x, y	vectors

Value

x elements without y elements

Index