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.iqfeed.net/symbolguide/index.cfm?symbolguide=lookup>) to code your trading 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 ........................................ 3
add_legend .................................................. 4
back_test ...................................................... 4
BBands ........................................................ 5
bbands .......................................................... 6
bw ............................................................... 6
calc_decimal_resolution .................................... 8
Candle .......................................................... 8
Cost ............................................................. 9
Crossover ....................................................... 10
crossover ...................................................... 10
distinct_colors ............................................... 11
dof ............................................................. 11
Ema ............................................................. 12
ema ............................................................. 13
empty_plot .................................................... 13
gen_futures_codes ........................................... 14
get_market_data ............................................. 14
hist_dt .......................................................... 17
Indicator ....................................................... 17
iqfeed ........................................................... 18
iround .......................................................... 25
lapply_named ................................................ 26
lines_ohlc ................................................... 26
lines_stacked_hist ......................................... 27
lmerge .......................................................... 28
multi_heatmap ................................................ 29
na_locf ........................................................ 30
Order ............................................................ 30
plot_dts ....................................................... 31
plot_table ..................................................... 33
plot_ts .......................................................... 33
Processor ...................................................... 35
returns ......................................................... 43
RollLinReg .................................................... 44
RollPercentRank ............................................. 45
RollRange ..................................................... 45
RollSd ........................................................ 46
RollVolumeProfile ........................................... 47
roll_futures .................................................. 48
roll_lm .......................................................... 48
roll_percent_rank ........................................... 49
roll_range ..................................................... 49
roll_sd .......................................................... 50
roll_sd_filter ................................................ 51
roll_volume_profile ....................................... 51
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**

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

**Usage**

```c++
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**

```c++
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

Description

C++ class documentation

Details

R function `crossover`.

Usage

Crossover

Public Members and Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add( std::pair&lt; double, double &gt; value )</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>IsAbove()</td>
<td>bool</td>
<td>first just went above second?</td>
</tr>
<tr>
<td>IsBelow()</td>
<td>bool</td>
<td>first just went below second?</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>factor</td>
<td>factor vector with levels UP, DN</td>
</tr>
</tbody>
</table>

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

Description

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

Usage

crossover(x, y)
distinct_colors

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

  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

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(double value)</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>GetValue()</td>
<td>double</td>
<td>return value</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>std::vector&lt;double&gt;</td>
<td>return values history</td>
</tr>
</tbody>
</table>

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_1m, 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

```r
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.

generate_data  Download historical market data

Description
Download historical market data

Usage

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

```r
generate_splits_and_dividends(symbol, from, to = from)
```

```r
generate_google_data(symbol, from, to = from)
```

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

```r
generate_iqfeed_data(symbol, from, to = from, period = "day", local = FALSE)
```
get_market_data

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

```r
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('R155000C9', '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

```r
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

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

### Public Members and Methods

Following methods must be specified for AnyIndicator
See Also

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

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

Other C++ classes: *BBands, Candle, Cost, Crossover, Ema, Order, Processor, RollLinReg, RollPercentRank, RollRange, RollSd, RollVolumeProfile, Rsi, Smo, 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**

<table>
<thead>
<tr>
<th>Market Id</th>
<th>Short Name</th>
<th>Long Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NGM</td>
<td>Nasdaq Global Market</td>
</tr>
<tr>
<td>2</td>
<td>NCM</td>
<td>National Capital Market</td>
</tr>
<tr>
<td>3</td>
<td>OTC</td>
<td>Nasdaq Other OTC</td>
</tr>
<tr>
<td>4</td>
<td>OTCBB</td>
<td>Nasdaq OTC Bulletin Board</td>
</tr>
<tr>
<td>5</td>
<td>NASDAQ</td>
<td>Nasdaq</td>
</tr>
<tr>
<td>6</td>
<td>NYSE_MKT</td>
<td>NYSE MKT (Equities and Bonds)</td>
</tr>
<tr>
<td>7</td>
<td>NYSE</td>
<td>New York Stock Exchange</td>
</tr>
<tr>
<td></td>
<td>Exchange Name</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CHX   Chicago Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PHLX  Philadelphia Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>NSX   National Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>NYSE_ARCA NYSE Archipelago</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BX    Boston Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>CBOE  Chicago Board Options Exchange</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>OPRA  OPRA System</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>NASD_ADF Nasdaq Alternate Display facility</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ISE   International Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>BOX   Boston Options Exchange</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>BATS  Better Alternative Trading System</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>NTRF  Nasdaq Trade Reporting Facility</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>PBOT  Philadelphia Board Of Trade</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>NGSM  Nasdaq Global Select Market</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>CANTOR Cantor Fitzgerald Exchange Treasury Funds</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>C2    CBOE C2 Options Exchange</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>NYSE_TRF NYSE Trade Reporting Facility</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>EDGA  Direct Edge A</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>EDGX  Direct Edge X</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>DTN   DTN</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>BYX   BATS Y Exchange</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>RUSSELL-FL Russell Investments (Fee-Liable)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>CBOT  Chicago Board Of Trade</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>DJ    Dow Jones (CBOT)</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>CFE   CBOE Futures Exchange</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>KCBOT Kansas City Board Of Trade</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>CME   Chicago Mercantile Exchange</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>MGE   Minneapolis Grain Exchange</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>NYMEX New York Mercantile Exchange</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>COMEX Commodities Exchange Center</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>ICEFU International Commodities Exchange Futures US</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>NYLUS NYSE Liffe US</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>CME-FL CME Indexes (Fee Liable)</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>CBOTMINI Chicago Board Of Trade Mini Sized Contracts</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>CMEMINI Chicago Mercantile Exchange Mini Sized Contracts</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>USFE  US Futures Exchange</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>NYMEXMINI Commodities Exchange Center Mini Sized Contracts</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>GREENX The Green Exchange</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>CLEARPORT New York Mercantile Exchange</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>COMEXMINI New York Mercantile Exchange Mini Sized Contracts</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>TSE   Toronto Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>MSE   Montreal Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>CVE   Canadian Venture Exchange</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>WSE   Winnipeg Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>ICEFC International Commodities Exchange Futures Canada</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>MX    Montreal Exchange</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>LSE   London Stock Exchange</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>FTSE  Financial Times Stock Exchange</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exchange</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>60</td>
<td>MDEX</td>
<td>Bursa Malaysia Derivatives</td>
</tr>
<tr>
<td>61</td>
<td>ICEFI</td>
<td>International Commodities Exchange Futures Derivatives</td>
</tr>
<tr>
<td>62</td>
<td>LME</td>
<td>London Metals Exchange</td>
</tr>
<tr>
<td>63</td>
<td>ICEEC</td>
<td>International Commodities Exchange European Commodities</td>
</tr>
<tr>
<td>64</td>
<td>ASXCM</td>
<td>ASX24 Commodities Exchange</td>
</tr>
<tr>
<td>65</td>
<td>DME</td>
<td>Dubai Mercantile Exchange</td>
</tr>
<tr>
<td>66</td>
<td>BMF</td>
<td>Brazilian Mercantile &amp; Future Exchange</td>
</tr>
<tr>
<td>67</td>
<td>SGX</td>
<td>Singapore International Monetary Exchange</td>
</tr>
<tr>
<td>68</td>
<td>EUREX</td>
<td>European Exchange</td>
</tr>
<tr>
<td>69</td>
<td>ENID</td>
<td>Euronext Index Derivatives</td>
</tr>
<tr>
<td>70</td>
<td>ICEEF</td>
<td>International Commodities Exchange European Financials</td>
</tr>
<tr>
<td>71</td>
<td>ENCOM</td>
<td>Euronext Commodities</td>
</tr>
<tr>
<td>72</td>
<td>TULLETT</td>
<td>Tullett Liberty (Forex)</td>
</tr>
<tr>
<td>73</td>
<td>BARCLAYS</td>
<td>Barclays Bank (Forex)</td>
</tr>
<tr>
<td>74</td>
<td>FXCM</td>
<td>Forex Capital Markets</td>
</tr>
<tr>
<td>75</td>
<td>WTB</td>
<td>Warenterminborse Hannover</td>
</tr>
<tr>
<td>76</td>
<td>MGKB</td>
<td>MGE-KCBOT (InterCommodity Spreads)</td>
</tr>
<tr>
<td>77</td>
<td>MGCBB</td>
<td>MGE-CBOT (InterCommodity Spreads)</td>
</tr>
<tr>
<td>78</td>
<td>TENFORE</td>
<td>Tenfore Systems</td>
</tr>
<tr>
<td>79</td>
<td>NYDME</td>
<td>NYMEX-DME (InterCommodity Spreads)</td>
</tr>
<tr>
<td>80</td>
<td>PSX</td>
<td>Philadelphia Stock Exchange</td>
</tr>
<tr>
<td>81</td>
<td>TGE</td>
<td>Tokyo Grain Exchange</td>
</tr>
<tr>
<td>82</td>
<td>TOCOM</td>
<td>Tokyo Commodities Exchange</td>
</tr>
<tr>
<td>83</td>
<td>SAFEX</td>
<td>South African Futures Exchange</td>
</tr>
<tr>
<td>84</td>
<td>EEXP</td>
<td>European Energy Exchange - Power</td>
</tr>
<tr>
<td>85</td>
<td>EEXN</td>
<td>European Energy Exchange - Natural Gas</td>
</tr>
<tr>
<td>86</td>
<td>EEXE</td>
<td>European Energy Exchange - Emission Rights</td>
</tr>
<tr>
<td>87</td>
<td>EEXC</td>
<td>European Energy Exchange - Coal</td>
</tr>
<tr>
<td>88</td>
<td>MIAX</td>
<td>Miami International Securities Exchange</td>
</tr>
<tr>
<td>89</td>
<td>KBCB</td>
<td>KCBOT-CBOT (InterCommodity Spreads)</td>
</tr>
<tr>
<td>90</td>
<td>PK_SHEETS</td>
<td>Pink Sheets - No Tier</td>
</tr>
<tr>
<td>91</td>
<td>PK_QXPREM</td>
<td>Pink Sheets - OTCQX - PremierQX Tier</td>
</tr>
<tr>
<td>92</td>
<td>PK_QXPRIME</td>
<td>Pink Sheets - OTCQX - PrimeQX Tier</td>
</tr>
<tr>
<td>93</td>
<td>PK_IQXPREM</td>
<td>Pink Sheets - OTCQX - International PremierQX Tier</td>
</tr>
<tr>
<td>94</td>
<td>PK_IQXPRIME</td>
<td>Pink Sheets - OTCQX - International PrimeQX Tier</td>
</tr>
<tr>
<td>95</td>
<td>PK_OTCQB</td>
<td>Pink Sheets - OTCBB Pink Sheets dually Quoted Tier</td>
</tr>
<tr>
<td>96</td>
<td>PK_BBONLY</td>
<td>Pink Sheets - OTCBB Only Tier</td>
</tr>
<tr>
<td>97</td>
<td>PK_CURRENT</td>
<td>Pink Sheets - Current Tier</td>
</tr>
<tr>
<td>98</td>
<td>PK_LIMITED</td>
<td>Pink Sheets - Limited Tier</td>
</tr>
<tr>
<td>99</td>
<td>PK_NOINFO</td>
<td>Pink Sheets - No Information Tier</td>
</tr>
<tr>
<td>100</td>
<td>PK_GREY</td>
<td>Pink Sheets - Grey Market Tier</td>
</tr>
<tr>
<td>101</td>
<td>PK_YL_SHEETS</td>
<td>Yellow Sheets</td>
</tr>
<tr>
<td>102</td>
<td>PK_PR_SHEETS</td>
<td>Partner Sheets</td>
</tr>
<tr>
<td>103</td>
<td>PK_GL_SHEETS</td>
<td>Global Sheets</td>
</tr>
<tr>
<td>104</td>
<td>PK_NYSE</td>
<td>Pink Sheets - NYSE Listed</td>
</tr>
<tr>
<td>105</td>
<td>PK_NASDAQ</td>
<td>Pink Sheets - NASDAQ Listed</td>
</tr>
<tr>
<td>106</td>
<td>PK_NYSE_AMEX</td>
<td>Pink Sheets - NYSE AMEX Listed</td>
</tr>
<tr>
<td>107</td>
<td>PK_ARCA</td>
<td>Pink Sheets - ARCA Listed</td>
</tr>
<tr>
<td>No.</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>108</td>
<td>NYSE_AMEX</td>
<td>NYSE AMEX Options Exchange</td>
</tr>
<tr>
<td>109</td>
<td>GLOBEX_RT</td>
<td>CME GLOBEX Group Authorization</td>
</tr>
<tr>
<td>110</td>
<td>CME_GBX</td>
<td>Chicago Mercantile Exchange (GLOBEX)</td>
</tr>
<tr>
<td>111</td>
<td>CBOT_GBX</td>
<td>Chicago Board Of Trade (GLOBEX)</td>
</tr>
<tr>
<td>112</td>
<td>NYMEX_GBX</td>
<td>New York Mercantile Exchange (GLOBEX)</td>
</tr>
<tr>
<td>113</td>
<td>COMEX_GBX</td>
<td>Commodities Exchange Center (GLOBEX)</td>
</tr>
<tr>
<td>114</td>
<td>DME_GBX</td>
<td>Dubai Mercantile Exchange (GLOBEX)</td>
</tr>
<tr>
<td>115</td>
<td>RUSSELL</td>
<td>Russell Investments</td>
</tr>
<tr>
<td>116</td>
<td>BZX</td>
<td>BATS Z Exchange</td>
</tr>
<tr>
<td>117</td>
<td>CFTC</td>
<td>US Commodity Futures Trading Commission</td>
</tr>
<tr>
<td>118</td>
<td>USDA</td>
<td>US Department of Agriculture</td>
</tr>
<tr>
<td>119</td>
<td>WASDE</td>
<td>World Supply and Demand Estimates Report</td>
</tr>
<tr>
<td>120</td>
<td>GRNST</td>
<td>Grain Stock Report</td>
</tr>
<tr>
<td>121</td>
<td>GEMINI</td>
<td>ISE Gemini Options Exchange</td>
</tr>
<tr>
<td>122</td>
<td>ARGUS</td>
<td>Argus Energy</td>
</tr>
<tr>
<td>123</td>
<td>RACKS</td>
<td>Racks Energy</td>
</tr>
<tr>
<td>124</td>
<td>SNL</td>
<td>SNL Energy</td>
</tr>
<tr>
<td>125</td>
<td>RFSPOT</td>
<td>Refined Fuels Spots Exchange</td>
</tr>
<tr>
<td>126</td>
<td>EOXNGF</td>
<td>EOX Live Natural Gas Forward Curve</td>
</tr>
<tr>
<td>127</td>
<td>EOXPWF</td>
<td>EOX Live Power Forward Curve</td>
</tr>
<tr>
<td>128</td>
<td>EOXCOR</td>
<td>EOX Live Correlations</td>
</tr>
<tr>
<td>129</td>
<td>ICEENDEX</td>
<td>ICE Energy Derivatives Exchange</td>
</tr>
<tr>
<td>130</td>
<td>KCBOT_GBX</td>
<td>Kansas City Board of Trade (GLOBEX)</td>
</tr>
<tr>
<td>131</td>
<td>MGE_GBX</td>
<td>Minneapolis Grain Exchange (GLOBEX)</td>
</tr>
<tr>
<td>132</td>
<td>BLOOMBERG</td>
<td>Bloomberg Indices</td>
</tr>
<tr>
<td>133</td>
<td>ELSLOT</td>
<td>Nord Pool Spot</td>
</tr>
<tr>
<td>134</td>
<td>N2EX</td>
<td>NASDAQ OMX-Nord Pool</td>
</tr>
<tr>
<td>135</td>
<td>ICEEA</td>
<td>International Commodities Exchange European Agriculture</td>
</tr>
<tr>
<td>136</td>
<td>CMEUR</td>
<td>Chicago Mercantile Exchange Europe Ltd</td>
</tr>
<tr>
<td>137</td>
<td>COMM3</td>
<td>Commodity 3 Exchange</td>
</tr>
<tr>
<td>138</td>
<td>JACOBSEN</td>
<td>The Jacobsen</td>
</tr>
<tr>
<td>139</td>
<td>NFX</td>
<td>NASDAQ OMX Futures</td>
</tr>
<tr>
<td>140</td>
<td>SGXAC</td>
<td>SGX Asia Clear</td>
</tr>
<tr>
<td>141</td>
<td>PJMISO</td>
<td>Pa-Nj-Md Independent System Operator</td>
</tr>
<tr>
<td>142</td>
<td>NYISO</td>
<td>New York Independent System Operator</td>
</tr>
<tr>
<td>143</td>
<td>NEISO</td>
<td>New England Independent System Operator</td>
</tr>
<tr>
<td>144</td>
<td>MWISO</td>
<td>Mid West Independent System Operator</td>
</tr>
<tr>
<td>145</td>
<td>SPISO</td>
<td>SW Power Pool Independent System Operator</td>
</tr>
<tr>
<td>146</td>
<td>CAISO</td>
<td>California Independent System Operator</td>
</tr>
<tr>
<td>147</td>
<td>ERCOT</td>
<td>ERCOT Independent System Operator</td>
</tr>
<tr>
<td>148</td>
<td>ABISO</td>
<td>Alberta Independent System Operator</td>
</tr>
<tr>
<td>149</td>
<td>ONISO</td>
<td>Ontario Independent System Operator</td>
</tr>
<tr>
<td>150</td>
<td>MERCURY</td>
<td>ISE Mercury Options Exchange</td>
</tr>
<tr>
<td>151</td>
<td>DCE</td>
<td>Dalian Commodity Exchange</td>
</tr>
<tr>
<td>152</td>
<td>ZCE</td>
<td>Zengchou Commodity Exchange</td>
</tr>
<tr>
<td>153</td>
<td>IEX</td>
<td>Investors Exchange LLC</td>
</tr>
<tr>
<td>154</td>
<td>MCX</td>
<td>Multi Commodity Exchange of India Limited</td>
</tr>
<tr>
<td>155</td>
<td>NCDEX</td>
<td>National Commodity Exchange of India Limited</td>
</tr>
<tr>
<td>Condition Code</td>
<td>Short Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>01</td>
<td>REGULAR</td>
<td>Normal Trade</td>
</tr>
<tr>
<td>02</td>
<td>ACQ</td>
<td>Acquisition</td>
</tr>
<tr>
<td>03</td>
<td>CASHM</td>
<td>Cash Only Market</td>
</tr>
<tr>
<td>04</td>
<td>BUNCHEED</td>
<td>Bunched Trade</td>
</tr>
<tr>
<td>05</td>
<td>AVGPRI</td>
<td>Average Price Trade</td>
</tr>
<tr>
<td>06</td>
<td>CASH</td>
<td>Cash Trade (same day clearing)</td>
</tr>
<tr>
<td>07</td>
<td>DIST</td>
<td>Distribution</td>
</tr>
<tr>
<td>08</td>
<td>NEXTDAY</td>
<td>Next Day Market</td>
</tr>
<tr>
<td>09</td>
<td>BURSTBSKT</td>
<td>Burst Basket Execution</td>
</tr>
<tr>
<td>0A</td>
<td>BUNCHEDSOLD</td>
<td>Bunched Sold Trade</td>
</tr>
<tr>
<td>0B</td>
<td>ORDETAIL</td>
<td>Opening/Reopening Trade Detail</td>
</tr>
<tr>
<td>0C</td>
<td>INTERDAY</td>
<td>Intraday Trade Detail</td>
</tr>
<tr>
<td>0D</td>
<td>BSKTONCLOSE</td>
<td>Basket Index on Close</td>
</tr>
<tr>
<td>0E</td>
<td>RULE127</td>
<td>Rule - 127 Trade NYSE</td>
</tr>
<tr>
<td>0F</td>
<td>RULE155</td>
<td>Rule - 155 Trade AMEX</td>
</tr>
<tr>
<td>10</td>
<td>SOLDLAST</td>
<td>Sold Last (late reporting)</td>
</tr>
<tr>
<td>11</td>
<td>NEXTDAYCLR</td>
<td>Next Day Clearing</td>
</tr>
<tr>
<td>12</td>
<td>LATEREP</td>
<td>Opened - Late Report of Opening Trade (in or out of sequence)</td>
</tr>
<tr>
<td>13</td>
<td>PRP</td>
<td>Prior Reference Price</td>
</tr>
<tr>
<td>14</td>
<td>SELLER</td>
<td>Seller</td>
</tr>
<tr>
<td>15</td>
<td>SPLIT</td>
<td>Split Trade</td>
</tr>
<tr>
<td>16</td>
<td>RSVD</td>
<td>(Reserved)</td>
</tr>
<tr>
<td>17</td>
<td>FORMT</td>
<td>Form-T Trade</td>
</tr>
<tr>
<td>18</td>
<td>CSTMBSKTX</td>
<td>Custom Basket Cross</td>
</tr>
<tr>
<td>19</td>
<td>SOLDSEQ</td>
<td>Sold Out of Sequence</td>
</tr>
<tr>
<td>1A</td>
<td>CANC</td>
<td>Cancelled Previous Transaction</td>
</tr>
<tr>
<td>1B</td>
<td>CANCLAST</td>
<td>Cancelled Last Transaction</td>
</tr>
<tr>
<td>1C</td>
<td>CANCOPEN</td>
<td>Cancelled Open Transaction</td>
</tr>
<tr>
<td>1D</td>
<td>CANCONLY</td>
<td>Cancelled Only Transaction</td>
</tr>
<tr>
<td>1E</td>
<td>OPEN</td>
<td>Late Report of Opening Trade - out of sequence</td>
</tr>
<tr>
<td>1F</td>
<td>OPNL</td>
<td>Late Report of Opening Trade - in correct sequence</td>
</tr>
<tr>
<td>20</td>
<td>AUTO</td>
<td>Transaction Executed Electronically</td>
</tr>
<tr>
<td>21</td>
<td>HALT</td>
<td>Halt</td>
</tr>
<tr>
<td>22</td>
<td>DELAYED</td>
<td>Delayed</td>
</tr>
<tr>
<td>23</td>
<td>NON_BOARDLOT</td>
<td>NON_BOARDLOT</td>
</tr>
<tr>
<td>24</td>
<td>POSIT</td>
<td>POSIT</td>
</tr>
<tr>
<td>25</td>
<td>REOP</td>
<td>Reopen After Halt</td>
</tr>
<tr>
<td>26</td>
<td>AJST</td>
<td>Contract Adjustment for Stock Dividend - Split - etc.</td>
</tr>
<tr>
<td>Code</td>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>27</td>
<td>SPRD</td>
<td>Spread - Trade in Two Options in the Same Class (a buy and a sell in the same</td>
</tr>
<tr>
<td>28</td>
<td>STDL</td>
<td>Straddle - Trade in Two Options in the Same Class (a buy and a sell in a put and</td>
</tr>
<tr>
<td>29</td>
<td>STPD</td>
<td>Follow a Non-stopped Trade</td>
</tr>
<tr>
<td>2A</td>
<td>CSTP</td>
<td>Cancel Stopped Transaction</td>
</tr>
<tr>
<td>2B</td>
<td>BWRT</td>
<td>Option Portion of a Buy/Write</td>
</tr>
<tr>
<td>2C</td>
<td>CMBO</td>
<td>Combo - Trade in Two Options in the Same Options Class (a buy and a sell in</td>
</tr>
<tr>
<td>2D</td>
<td>UNSPEC</td>
<td>Unspecified</td>
</tr>
<tr>
<td>2E</td>
<td>MC_OFCLCLOSE</td>
<td>Market Center Official Closing Price</td>
</tr>
<tr>
<td>2F</td>
<td>STPD_REGULAR</td>
<td>Stopped Stock - Regular Trade</td>
</tr>
<tr>
<td>30</td>
<td>STPD_SOLDLAST</td>
<td>Stopped Stock - Sold Last</td>
</tr>
<tr>
<td>31</td>
<td>STPD_SOLDSEQ</td>
<td>Stopped Stock - Sold out of sequence</td>
</tr>
<tr>
<td>32</td>
<td>BASIS</td>
<td>Basis</td>
</tr>
<tr>
<td>33</td>
<td>VWAP</td>
<td>Volume-Weighted Average Price</td>
</tr>
<tr>
<td>34</td>
<td>STS</td>
<td>Special Trading Session</td>
</tr>
<tr>
<td>35</td>
<td>STT</td>
<td>Special Terms Trading</td>
</tr>
<tr>
<td>36</td>
<td>CONTINGENT</td>
<td>Contingent Order</td>
</tr>
<tr>
<td>37</td>
<td>INTERNALX</td>
<td>Internal Cross</td>
</tr>
<tr>
<td>38</td>
<td>MOC</td>
<td>Market On Close Trade</td>
</tr>
<tr>
<td>39</td>
<td>MC_OFCLOPEN</td>
<td>Market Center Official Opening Price</td>
</tr>
<tr>
<td>3A</td>
<td>FORTMTSOLDSEQ</td>
<td>Form-T Sold Out of Sequence</td>
</tr>
<tr>
<td>3B</td>
<td>YELLOWFLAG</td>
<td>Yellow Flag</td>
</tr>
<tr>
<td>3C</td>
<td>AUTOEXEC</td>
<td>Auto Execution</td>
</tr>
<tr>
<td>3D</td>
<td>INTRMRK_SWEEP</td>
<td>Intramaket Sweep</td>
</tr>
<tr>
<td>3E</td>
<td>DERIVPRI</td>
<td>Derivately Priced</td>
</tr>
<tr>
<td>3F</td>
<td>REOPENING</td>
<td>Re-Opening Prints</td>
</tr>
<tr>
<td>40</td>
<td>CLSING</td>
<td>Closing Prints</td>
</tr>
<tr>
<td>41</td>
<td>CAP_ELCTN</td>
<td>CAP (Conversion and Parity) election trade</td>
</tr>
<tr>
<td>42</td>
<td>CROSS_TRADE</td>
<td>Cross Trade</td>
</tr>
<tr>
<td>43</td>
<td>PRICE_VAR</td>
<td>Price Variation</td>
</tr>
<tr>
<td>44</td>
<td>STKOPT_TRADE</td>
<td>Stock-Option Trade</td>
</tr>
<tr>
<td>45</td>
<td>SPIM</td>
<td>stopped at price that did not constitute a Trade-Through</td>
</tr>
<tr>
<td>46</td>
<td>BNMT</td>
<td>Benchmark Trade</td>
</tr>
<tr>
<td>47</td>
<td>TTEXEMPT</td>
<td>Transaction is Trade Through Exempt</td>
</tr>
<tr>
<td>48</td>
<td>LATE</td>
<td>Late Market</td>
</tr>
<tr>
<td>49</td>
<td>XCHG_PHYSICAL</td>
<td>Exchange for Physical</td>
</tr>
<tr>
<td>4A</td>
<td>CABINET</td>
<td>Cabinet</td>
</tr>
<tr>
<td>4B</td>
<td>DIFFERENTIAL</td>
<td>Differential</td>
</tr>
<tr>
<td>4C</td>
<td>HIT</td>
<td>Hit</td>
</tr>
<tr>
<td>4D</td>
<td>IMPLIED</td>
<td>Implied</td>
</tr>
<tr>
<td>4E</td>
<td>LG_ORDER</td>
<td>Large Order</td>
</tr>
<tr>
<td>4F</td>
<td>SM_ORDER</td>
<td>Small Order</td>
</tr>
<tr>
<td>50</td>
<td>MATCH</td>
<td>Match/Cross Trade</td>
</tr>
<tr>
<td>51</td>
<td>NOMINAL</td>
<td>Nominal</td>
</tr>
<tr>
<td>52</td>
<td>OPTION_EX</td>
<td>Option Exercise</td>
</tr>
<tr>
<td>53</td>
<td>PERCENTAGE</td>
<td>Percentage</td>
</tr>
<tr>
<td>54</td>
<td>AUTOQUOTE</td>
<td>Auto Quotes</td>
</tr>
<tr>
<td>55</td>
<td>INDICATIVE</td>
<td>Indicative</td>
</tr>
<tr>
<td>56</td>
<td>TAKE</td>
<td>Take</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>NOMINAL_CABINET</td>
<td>Nominal Cabinet</td>
</tr>
<tr>
<td>58</td>
<td>CHNG_TRANSACTION</td>
<td>Changing Transaction</td>
</tr>
<tr>
<td>59</td>
<td>CHNG_TRANS_CAB</td>
<td>Changing Transaction Cabinet</td>
</tr>
<tr>
<td>5A</td>
<td>FAST</td>
<td>Fast Market (ssfutures)</td>
</tr>
<tr>
<td>5B</td>
<td>NOMINAL_UPDATE</td>
<td>Nominal Update</td>
</tr>
<tr>
<td>5C</td>
<td>INACTIVE</td>
<td>Inactive - Nominal - No Trade</td>
</tr>
<tr>
<td>5D</td>
<td>DELTA</td>
<td>Last Trade with Delta Exchange</td>
</tr>
<tr>
<td>5E</td>
<td>ERRATIC</td>
<td>Erratic</td>
</tr>
<tr>
<td>5F</td>
<td>RISK_FACTOR</td>
<td>Risk Factor</td>
</tr>
<tr>
<td>60</td>
<td>OPT_ADDON</td>
<td>Short Option Add-On</td>
</tr>
<tr>
<td>61</td>
<td>VOLATILITY</td>
<td>Volatility Trade</td>
</tr>
<tr>
<td>62</td>
<td>SPD_RPT</td>
<td>Spread Reporting</td>
</tr>
<tr>
<td>63</td>
<td>VOL_ADJ</td>
<td>Volume Adjustment</td>
</tr>
<tr>
<td>64</td>
<td>BLANK</td>
<td>Blank out associated price</td>
</tr>
<tr>
<td>65</td>
<td>SOLDLATE</td>
<td>Late report of transaction - in correct sequence</td>
</tr>
<tr>
<td>66</td>
<td>BLKT</td>
<td>Block Trade</td>
</tr>
<tr>
<td>67</td>
<td>EXPH</td>
<td>Exchange Future for Physical</td>
</tr>
<tr>
<td>68</td>
<td>SPECIALIST_A</td>
<td>Ask from specialist Book</td>
</tr>
<tr>
<td>69</td>
<td>SPECIALIST_B</td>
<td>Bid from specialist Book</td>
</tr>
<tr>
<td>6A</td>
<td>SPECIALIST_BA</td>
<td>Both Bid and Ask from Specialist Book</td>
</tr>
<tr>
<td>6B</td>
<td>ROTATION</td>
<td>Rotation</td>
</tr>
<tr>
<td>6C</td>
<td>HOLIDAY</td>
<td>Holiday</td>
</tr>
<tr>
<td>6D</td>
<td>PREOPENING</td>
<td>Pre Opening</td>
</tr>
<tr>
<td>6E</td>
<td>POST_FULL</td>
<td>Post Full</td>
</tr>
<tr>
<td>6F</td>
<td>POST_RESTRICTED</td>
<td>Post Restricted</td>
</tr>
<tr>
<td>70</td>
<td>CLOSING_AUCTION</td>
<td>Closing Auction</td>
</tr>
<tr>
<td>71</td>
<td>BATCH</td>
<td>Batch</td>
</tr>
<tr>
<td>72</td>
<td>TRADING</td>
<td>Trading</td>
</tr>
<tr>
<td>73</td>
<td>OFFICIAL</td>
<td>Official Bid/Ask price</td>
</tr>
<tr>
<td>74</td>
<td>UNOFFICIAL</td>
<td>Unofficial Bid/Ask price</td>
</tr>
<tr>
<td>75</td>
<td>MIDPRICE</td>
<td>Midprice last</td>
</tr>
<tr>
<td>76</td>
<td>FLOOR</td>
<td>Floor B/A price</td>
</tr>
<tr>
<td>77</td>
<td>CLOSE</td>
<td>Closing Price</td>
</tr>
<tr>
<td>78</td>
<td>HIGH</td>
<td>End of Session High Price</td>
</tr>
<tr>
<td>79</td>
<td>LOW</td>
<td>End of Session Low Price</td>
</tr>
<tr>
<td>7A</td>
<td>BACKWARDATION</td>
<td>Backwardation - immediate delivery costing more than future delivery</td>
</tr>
<tr>
<td>7B</td>
<td>CONTANGO</td>
<td>Contango - future delivery costing more than immediate delivery</td>
</tr>
<tr>
<td>7C</td>
<td>RF_SETTLEMENT</td>
<td>Refined Fuel Spot Settlement</td>
</tr>
<tr>
<td>7D</td>
<td>RF_RESERVED1</td>
<td>Refined Fuel Spot Reserved - 1</td>
</tr>
<tr>
<td>7E</td>
<td>RF_RESERVED2</td>
<td>Refined Fuel Spot Reserved - 2</td>
</tr>
<tr>
<td>7F</td>
<td>RF_RESERVED3</td>
<td>Refined Fuel Spot Reserved - 3</td>
</tr>
<tr>
<td>80</td>
<td>RF_RESERVED4</td>
<td>Refined Fuel Spot Reserved - 4</td>
</tr>
<tr>
<td>81</td>
<td>YIELD</td>
<td>Yield Price</td>
</tr>
<tr>
<td>82</td>
<td>BASIS_HIGH</td>
<td>Current Basis High Value</td>
</tr>
<tr>
<td>83</td>
<td>BASIS_LOW</td>
<td>Current Bases Low Value</td>
</tr>
<tr>
<td>84</td>
<td>UNCLEAR</td>
<td>Bid or offer price is unclear</td>
</tr>
<tr>
<td>85</td>
<td>OTC</td>
<td>Over the counter trade</td>
</tr>
<tr>
<td>86</td>
<td>MS</td>
<td>Trade entered by Market Supervision</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>ODDLOT</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>CORRCSLDLAST</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>QUALCONT</td>
<td></td>
</tr>
<tr>
<td>8A</td>
<td>MC_OPEN</td>
<td></td>
</tr>
<tr>
<td>8B</td>
<td>CONFIRMED</td>
<td></td>
</tr>
<tr>
<td>8C</td>
<td>OUTAGE</td>
<td></td>
</tr>
<tr>
<td>8D</td>
<td>SPRD_LEG</td>
<td></td>
</tr>
<tr>
<td>8E</td>
<td>BNDL_SPRD_LEG</td>
<td></td>
</tr>
<tr>
<td>8F</td>
<td>LATECORR</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>CONTRA</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>IFTRANSFER</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>IF_CROSS</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>NEG_TRADE</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>OTC_CANC</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>OTC_TRADE</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>SI_LATECORR</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>SI_TRADE</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>AUCT_TRADE</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>LATE</td>
<td></td>
</tr>
<tr>
<td>9A</td>
<td>STRAT</td>
<td></td>
</tr>
<tr>
<td>9B</td>
<td>INDICATIVE_AUCT</td>
<td></td>
</tr>
<tr>
<td>9C</td>
<td>OUTAGE</td>
<td></td>
</tr>
<tr>
<td>9D</td>
<td>SPRD_LEG</td>
<td></td>
</tr>
<tr>
<td>9E</td>
<td>BNDL_SPRD_LEG</td>
<td></td>
</tr>
<tr>
<td>9F</td>
<td>LATECORR</td>
<td></td>
</tr>
<tr>
<td>9G</td>
<td>CONTRA</td>
<td></td>
</tr>
<tr>
<td>9H</td>
<td>IFTRANSFER</td>
<td></td>
</tr>
<tr>
<td>9I</td>
<td>IF_CROSS</td>
<td></td>
</tr>
<tr>
<td>9J</td>
<td>NEG_TRADE</td>
<td></td>
</tr>
<tr>
<td>9K</td>
<td>OTC_CANC</td>
<td></td>
</tr>
<tr>
<td>9L</td>
<td>OTC_TRADE</td>
<td></td>
</tr>
<tr>
<td>9M</td>
<td>SI_LATECORR</td>
<td></td>
</tr>
<tr>
<td>9N</td>
<td>SI_TRADE</td>
<td></td>
</tr>
<tr>
<td>9O</td>
<td>AUCT_TRADE</td>
<td></td>
</tr>
<tr>
<td>9P</td>
<td>LATE</td>
<td></td>
</tr>
<tr>
<td>9Q</td>
<td>STRAT</td>
<td></td>
</tr>
<tr>
<td>9R</td>
<td>INDICATIVE_AUCT</td>
<td></td>
</tr>
</tbody>
</table>

*to retrieve above table use QuantTools:::.get_iqfeed_trade_conditions_info()

**Examples**

```r
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?
**lines_stacked_hist**

Add stacked histogram to active time series plot

---

**Description**

Add stacked histogram to active time series plot

**Usage**

```r
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_stacked_hist, 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**

```r
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 Real Est
   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

```r
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isNew()</td>
<td>bool</td>
<td>order is new or just sent to exchange?</td>
</tr>
<tr>
<td>isRegistered()</td>
<td>bool</td>
<td>placement confirmation received from exchange?</td>
</tr>
<tr>
<td>isCancelling()</td>
<td>bool</td>
<td>cancel request sent to exchange?</td>
</tr>
<tr>
<td>isCancelled()</td>
<td>bool</td>
<td>cancel confirmation received from exchange?</td>
</tr>
<tr>
<td>isExecuted()</td>
<td>bool</td>
<td>execution confirmation received from exchange?</td>
</tr>
<tr>
<td>isBuy?</td>
<td>bool</td>
<td>buy order?</td>
</tr>
</tbody>
</table>
### 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 to cancel

### 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

---

### 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 should 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 should 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
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

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)
plot_ts

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', 'month', 'year'. 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

SetCost(Rcpp::List cost) void see 'cost' in 'Options' section
SetStop(Rcpp::List stop) void see 'stop' in 'Options' section
SetStartTradingTime(double t) void see 'trade_start' in 'Options' section
SetLatencyReceive(double x) void see 'latency_receive' in 'Options' section
SetLatencySend(double x) void see 'latency_send' in 'Options' section
SetLatency(double x) void see 'latency' in 'Options' section
SetTradingHours(double start, double end) void see 'trading_hours' in 'Options' section
SetPriceStep(double priceStep) void see 'price_step' in 'Options' section
SetExecutionType(ExecutionType executionType) void see 'execution_type' in 'Options' section
SetExecutionType(std::string executionType) void see 'execution_type' in 'Options' section
SetIntervals(std::vector<double> starts, std::vector<double> ends) void see 'intervals' in 'Options' section
AllowLimitToHitMarket() void see 'allow_limit_to_hit_market' in 'Options' section
AllowExactStop() void see 'allow_exact_stop' in 'Options' section
StopTrading() void if called trading stop triggered. See 'stop' in 'Options' section
CanTrade() bool check if trading not stopped
IsTradingHoursSet() bool check if trading hours set
CancelOrders() void cancel active orders
GetCandle() Candle get current candle
GetPosition() int total executed position, positive means long, negative means short
GetPositionPlanned() int total number of orders processing (not executed or cancelled yet)
GetMarketValue() double total portfolio percent value (initial value is 0)
GetCandles() Rcpp::List candles history, see 'Candles' section
GetOrders() Rcpp::List orders history, see 'Orders' section
GetTrades() Rcpp::List trades history, see 'Trades' section
GetSummary() Rcpp::List trades summary, see 'Summary' section
GetOnCandleMarketValueHistory() std::vector<double> vector of portfolio value history recalculated on candle complete
GetOnCandleDrawDownHistory() std::vector<double> vector of portfolio drawdown history recalculated on candle complete
GetOnDayClosePerformanceHistory() Rcpp::List daily performance history, see 'Daily Performance' section
Reset() void resets to initial state

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 later.

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td>price</td>
<td>price</td>
</tr>
<tr>
<td>volume</td>
<td>volume</td>
</tr>
</tbody>
</table>

tick id is ticks row number.

Candles

Candles returned as data.table with the following columns:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>time when formed</td>
</tr>
<tr>
<td>open</td>
<td>first tick price</td>
</tr>
<tr>
<td>high</td>
<td>maximum tick price</td>
</tr>
<tr>
<td>low</td>
<td>minimum tick price</td>
</tr>
<tr>
<td>close</td>
<td>last tick price</td>
</tr>
<tr>
<td>volume</td>
<td>total volume traded</td>
</tr>
<tr>
<td>id</td>
<td>tick id when formed (first tick after time formed)</td>
</tr>
</tbody>
</table>

Orders

Orders returned as data.table with the following columns:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_trade</td>
<td>trade id</td>
</tr>
<tr>
<td>id_sent</td>
<td>tick id when order was sent to exchange</td>
</tr>
<tr>
<td>id_processed</td>
<td>tick id when enter order execution or cancelled confirmation was received</td>
</tr>
<tr>
<td>time_sent</td>
<td>time when order was sent to exchange</td>
</tr>
<tr>
<td>time_processed</td>
<td>time when order execution or cancelled confirmation was received</td>
</tr>
<tr>
<td>price_init</td>
<td>initial price</td>
</tr>
<tr>
<td>price_exec</td>
<td>execution price</td>
</tr>
<tr>
<td>side</td>
<td>buy/sell</td>
</tr>
<tr>
<td>type</td>
<td>limit/market/stop/trail</td>
</tr>
<tr>
<td>state</td>
<td>new/registered/executed/cancelling/cancelled</td>
</tr>
<tr>
<td>comment</td>
<td>comment</td>
</tr>
</tbody>
</table>

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:

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

Summary

Back test summary statistics:

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

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 calculated on daily returns
sortino annualized Sortino ratio calculated on daily returns
r_squared R Squared calculated on daily PnL values
avg_dd average drawdown calculated on daily drawdown history

Daily Performance

Back test daily performance history:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>return</td>
<td>return</td>
</tr>
<tr>
<td>pnl</td>
<td>cumulative pnl</td>
</tr>
<tr>
<td>drawdown</td>
<td>drawdown</td>
</tr>
<tr>
<td>n_per_day</td>
<td>number of closed trades</td>
</tr>
<tr>
<td>avg_pnl</td>
<td>average trade pnl</td>
</tr>
</tbody>
</table>

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 whether 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, Sm, Stochastic, Tick

Examples

```R
#-----------------------------
## 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 )

# load tick data
data( 'ticks' )

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

# 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 )


returns

# 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

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

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

\[
\text{RollPercentRank}( \text{int} \ n )
\]

Public Members and Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add( InputType value )</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>std::vector&lt; double &gt;</td>
<td>history vector</td>
</tr>
</tbody>
</table>

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

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

See Also

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

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

RollSd

C++ Rolling Standard Deviation class

Description

C++ class documentation

Arguments

n indicator period

Usage

RollSd( int n )
Public Members and Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(InputType value)</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>std::vector&lt;double&gt;</td>
<td>factor vector with levels UP, DN</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(Tick tick)</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetValue()</td>
<td>std::map&lt;double,double&gt;</td>
<td>histogram where first is price and second is volume</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>List</td>
<td>return values history data.table with columns time, profile where profile</td>
</tr>
</tbody>
</table>
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**

```r
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**

```r
roll_lm(x, y, n)
```

```r
roll_correlation(x, y, n)
```

**Arguments**

- `x, y`: numeric vectors
- `n`: window size
roll_percent_rank

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

<table>
<thead>
<tr>
<th>roll_percent_rank</th>
<th>Rolling Percent Rank</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>roll_range</th>
<th>Rolling Range</th>
</tr>
</thead>
</table>

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 vector
- \( n \) 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**

```r
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_sd`, `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**

```r
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**

```r
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: \texttt{bbands}, \texttt{crossover}, \texttt{ema}, \texttt{roll_lm}, \texttt{roll_percent_rank}, \texttt{roll_range}, \texttt{roll_sd}, \texttt{rsi}, \texttt{sma}, \texttt{stochastic}

---

**round\_POSIXct**

*Round POSIXct timestamps*

**Description**

Round POSIXct timestamps

**Usage**

\begin{verbatim}
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"))
\end{verbatim}

**Arguments**

- \texttt{x} POSIXct vector
- \texttt{n} number of units to round off
- \texttt{units} to round off to
- \texttt{method} round method, see \texttt{Round}

**Details**

Rounds POSIXct vector with specified method.

---

**Rsi**

*C++ Relative Strength Index class*

**Description**

C++ class documentation

**Arguments**

- \texttt{n} indicator period

**Details**

R function \texttt{rsi}.
**rsi**

**Usage**

\[ \text{Rsi}(\text{int} \ n) \]

**Public Members and Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add( double value )</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetValue()</td>
<td>double</td>
<td>return value</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>std::vector&lt;double&gt;</td>
<td>return values history</td>
</tr>
</tbody>
</table>

**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**

\[ \text{rsi}(x, n) \]

**Arguments**

- \( x \): numeric vectors
- \( n \): window size

**See Also**

Other technical indicators: bbands, crossover, ema, roll_dm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, sma, stochastic
QuantTools settings

### Description

QuantTools settings

### Usage

```r
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

```r
# 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add( double value )</td>
<td>void</td>
<td>update indicator</td>
</tr>
<tr>
<td>GetValue()</td>
<td>double</td>
<td>return value</td>
</tr>
<tr>
<td>Reset()</td>
<td>void</td>
<td>reset to initial state</td>
</tr>
<tr>
<td>IsFormed()</td>
<td>bool</td>
<td>is indicator value valid?</td>
</tr>
<tr>
<td>GetHistory()</td>
<td>std::vector&lt;double&gt;</td>
<td>return values history</td>
</tr>
</tbody>
</table>
**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

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

See Also

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

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

Value
data.table with columns k_fast, d_fast, d_slow

See Also
Other technical indicators: \texttt{bbands, crossover, ema, roll_lm, roll_percent_rank, roll_range, roll_sd, roll_volume_profile, rsi, sma}

---

Description
Store historical market data

Usage
\begin{verbatim}
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)
\end{verbatim}

Arguments
\begin{itemize}
  \item \texttt{from, to} text dates in format "YYYY-mm-dd"
  \item \texttt{verbose} show progress?
\end{itemize}

Details
See example below.

Examples

```r
## 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:
```

```r
# 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', 'QES#' ),
  # 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'
)
```
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**
- **time** seconds since epoch
- **price**
- **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

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: 
\[(\text{time}, \text{price}, \text{volume}) \times 4; (\text{time} - \text{period}/2, \text{high}, \text{volume}/4); (\text{time} - \text{period}/2, \text{low}, \text{volume}/4); (\text{time} - \text{period}/100, \text{close}, \text{volume}/4).\]

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

* datasets
  distinct_colors, 11
* data
  ticks, 61
  %bw (%bw), 6
  %dof (%dof), 11
  %dofc (%dofc), 11
  %w/o (%w/o), 63

add_last_values, 3, 4, 11, 13, 17, 27, 29
add_legend, 3, 4, 11, 13, 17, 27, 29, 33, 34

back_test, 4
Bands, 5, 9, 10, 12, 18, 31, 40, 44–48, 53, 56, 57, 60
bbands, 5, 6, 11, 13, 49, 50, 52, 53, 56, 58
bw, 6

calc_decimal_resolution, 8
Candle, 6, 8, 9, 10, 12, 18, 31, 35, 40, 44–48, 53, 56, 57, 60
ceiling_POSIXct (round_POSIXct), 52
Cost, 6, 9, 10, 12, 18, 31, 35, 39, 40, 44–48, 53, 56, 57, 60
Crossover, 6, 9, 10, 12, 18, 31, 40, 44–48, 53, 56, 57, 60
crossover, 6, 10, 10, 13, 49, 50, 52, 53, 56, 58

distinct_colors, 3, 4, 11, 13, 17, 27, 29, 33, 34
dof, 11
dofc (dofc), 11

Ema, 6, 9, 10, 12, 18, 31, 40, 44–48, 53, 56, 57, 60
ema, 6, 11, 12, 13, 49, 50, 52, 53, 56, 58
empty_plot, 3, 4, 11, 13, 17, 27, 29, 33, 34
gen_futures_codes, 14, 15
get_finam_data (get_market_data), 14

get_google_data (get_market_data), 14
get_iqfeed_data (get_market_data), 14
get_market_data, 14
get_moex_continuous_futures_data (get_market_data), 14
get_moex_futures_data (get_market_data), 14
get_moex_options_data (get_market_data), 14
get_yahoo_data (get_market_data), 14
get_yahoo_splits_and_dividends (get_market_data), 14

hist_dt, 3, 4, 11, 13, 17, 27, 29, 33, 34

Indicator, 6, 9, 10, 12, 17, 31, 40, 44–48, 53, 56, 57, 60
iqfeed, 15, 18
iround, 25

lapply, 26
lapply_named, 26
legend, 32
lines, 4, 4, 31, 34
lines__ohlc, 3, 4, 11, 13, 17, 26, 27, 29, 33, 34
lines_stacked_hist, 3, 4, 11, 13, 17, 27, 29, 33, 34
lmerge, 28

multi_heatmap, 3, 4, 11, 13, 17, 27, 29, 33, 34

na_locf, 30

Order, 6, 9, 10, 12, 18, 30, 35, 40, 44–48, 53, 56, 57, 60

par, 31, 33, 34
plot, 34
plot_dts, 31
plot_table, 3, 4, 11, 13, 17, 27, 29, 33, 34
plot_ts, 3, 4, 11, 13, 17, 27, 29, 33, 33

64
<table>
<thead>
<tr>
<th>Term</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>4, 31</td>
</tr>
<tr>
<td>Processor</td>
<td>6, 9, 10, 12, 18, 31, 35, 44–48, 51, 53, 56, 57, 60, 61</td>
</tr>
<tr>
<td>QuantTools_settings (settings)</td>
<td>54</td>
</tr>
<tr>
<td>QuantTools_settings_defaults</td>
<td>(settings), 54</td>
</tr>
<tr>
<td>returns</td>
<td>43</td>
</tr>
<tr>
<td>roll_correlation (roll_lm)</td>
<td>48</td>
</tr>
<tr>
<td>roll_futures</td>
<td>48</td>
</tr>
<tr>
<td>roll_max</td>
<td>6, 11, 13, 44, 48, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_min</td>
<td>6, 11, 13, 44, 48, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_percent_rank</td>
<td>6, 11, 13, 49, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_quantile</td>
<td>6, 11, 13, 49, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_range</td>
<td>6, 11, 13, 46, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_sd</td>
<td>6, 11, 13, 49, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>roll_sd_filter</td>
<td>51</td>
</tr>
<tr>
<td>roll_volume_profile</td>
<td>6, 11, 13, 47, 49, 50, 51, 53, 56, 58</td>
</tr>
<tr>
<td>RollLinReg</td>
<td>6, 9, 10, 12, 18, 31, 35, 40, 44, 45–48, 53, 56, 57, 60</td>
</tr>
<tr>
<td>RollPercentRank</td>
<td>6, 9, 10, 12, 18, 31, 35, 40, 44, 45–48, 53, 56, 57, 60</td>
</tr>
<tr>
<td>RollRange</td>
<td>6, 9, 10, 12, 18, 31, 35, 40, 44, 45, 47, 48, 53, 56, 57, 60</td>
</tr>
<tr>
<td>RollSd</td>
<td>6, 9, 10, 12, 18, 31, 35, 40, 44–46, 46, 48, 53, 56, 57, 60</td>
</tr>
<tr>
<td>RollVolumeProfile</td>
<td>6, 9, 10, 12, 18, 31, 40, 44–47, 47, 53, 56, 57, 60</td>
</tr>
<tr>
<td>Round</td>
<td>52</td>
</tr>
<tr>
<td>round_POSIXct</td>
<td>52</td>
</tr>
<tr>
<td>Rsi</td>
<td>6, 9, 10, 12, 18, 31, 40, 44–48, 52, 56, 57, 60</td>
</tr>
<tr>
<td>rsi</td>
<td>6, 11, 13, 49, 50, 52, 53, 56, 58</td>
</tr>
<tr>
<td>settings</td>
<td>54</td>
</tr>
<tr>
<td>Sma</td>
<td>6, 9, 10, 12, 18, 31, 40, 44–48, 53, 55, 57, 60</td>
</tr>
<tr>
<td>sma</td>
<td>6, 11, 13, 49, 50, 52, 53, 55, 56, 58</td>
</tr>
<tr>
<td>Stochastic</td>
<td>6, 9, 10, 12, 18, 31, 40, 44–48, 53, 56, 56, 60</td>
</tr>
<tr>
<td>stochastic</td>
<td>6, 11, 13, 49, 50, 52, 53, 56, 57, 57</td>
</tr>
<tr>
<td>store_finam_data (store_market_data)</td>
<td>58</td>
</tr>
<tr>
<td>store_iqfeed_data (store_market_data)</td>
<td>58</td>
</tr>
<tr>
<td>store_market_data</td>
<td>16, 58</td>
</tr>
<tr>
<td>store_moex_data</td>
<td>58</td>
</tr>
<tr>
<td>t_to_x (plot_ts)</td>
<td>33</td>
</tr>
<tr>
<td>ticks</td>
<td>61</td>
</tr>
<tr>
<td>to_candles</td>
<td>51</td>
</tr>
<tr>
<td>to_ticks</td>
<td>61</td>
</tr>
<tr>
<td>to.UTC</td>
<td>62</td>
</tr>
<tr>
<td>trunc_POSIXct (round_POSIXct)</td>
<td>52</td>
</tr>
<tr>
<td>wo</td>
<td>63</td>
</tr>
</tbody>
</table>