IT-FINANCE

Information Technology for Finance

Programming Guide

ProBacktest

2011 edition v3.1

TABLE OF CONTENTS

PROBACKTEST INTRODUCTION

ProBacktest is a tool that will allow you to create and test personalized investment strategies on any available timeframe and period of historical data for a given security.

ProBacktest uses ProBuilder programming language (it is advised that you also consult the ProBuilder manual) with extensions that are used exclusively in creating strategies with ProBacktest.

In this module, you can simulate taking based on on personalized conditions including:

- Technical indicators with personalized parameters
- Your current market position (open positions or not...)
- Dates and times you want to open or close the position (ex: open of next day...)
- The way in which you take a position (market order, limit order, ...)
- Positioning stops
- Performance of previous trades
- Execution price of previous orders

The results of a ProBacktest simulation are presented in the form of :

- The "Equity curve" (or "Gain and Loss Curve"), which shows you the state of your virtual portfolio over the course of the period on which you are simulating your trading strategy.
- The positions histogram which shows your open positions (green bar for a long position, red bar for a short position, no bar for no position).
- The detailed report which shows the general results of your strategy simulation over the period tested and on the security tested.

This document is written as a continuation of the ProBuilder manual but can be read independently

Readers who are used to programming can skip directly to chapter 2 or consult the glossary to quickly find an explanation related to a function they are looking for.

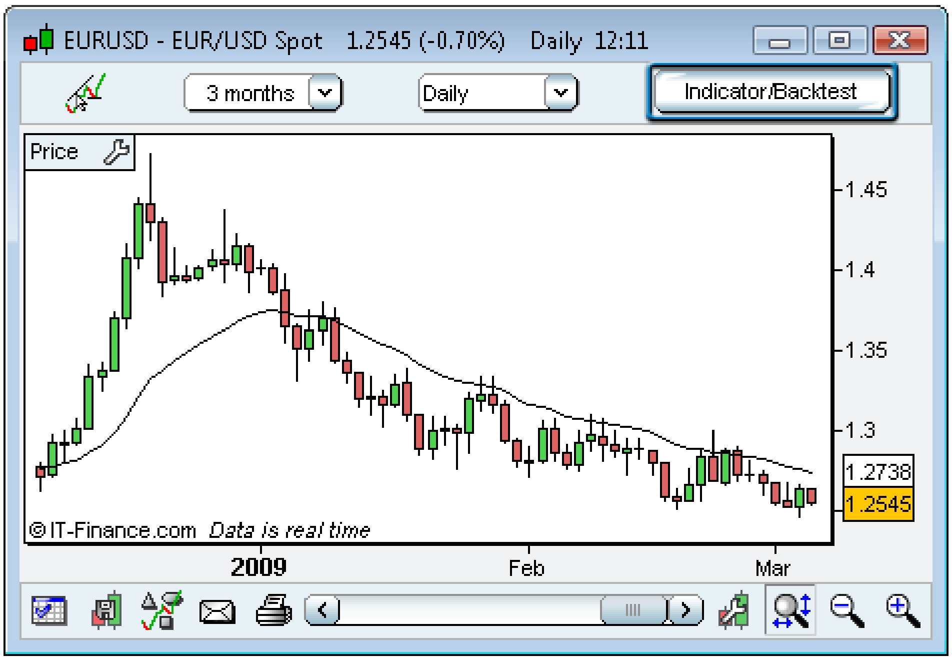
The ideas in this section and in the rest of the manual are meant to help you code and test your own ideas. They are not investment advice.

We wish you the best of success in your trading and hope you will enjoy the manual.

CHAPTER I: INTRODUCTION

Accessing ProBacktest

The zone for ProBacktest creation may be accessed with the "Indicator/Backtest" button in the upper-right corner of every chart.


EURUSD - EUR/USD Spot 1.2545 (-0.70%) Daily 12:11
3 months
Daily
Indicator/Backtest
Price
1.45
1.4
1.35
1.3
1.2738
1.2545
© IT-Finance.com Data is real time
2009
Feb
Mar

You can then access the indicator and backtest management window. Click the “ProBacktest” tab. You will be able to:

- Access the list of existing ProBacktests (predefined or your own)
- Create a new ProBacktest which can then be applied to any security or timeframe
- Modify or delete an existing ProBacktest

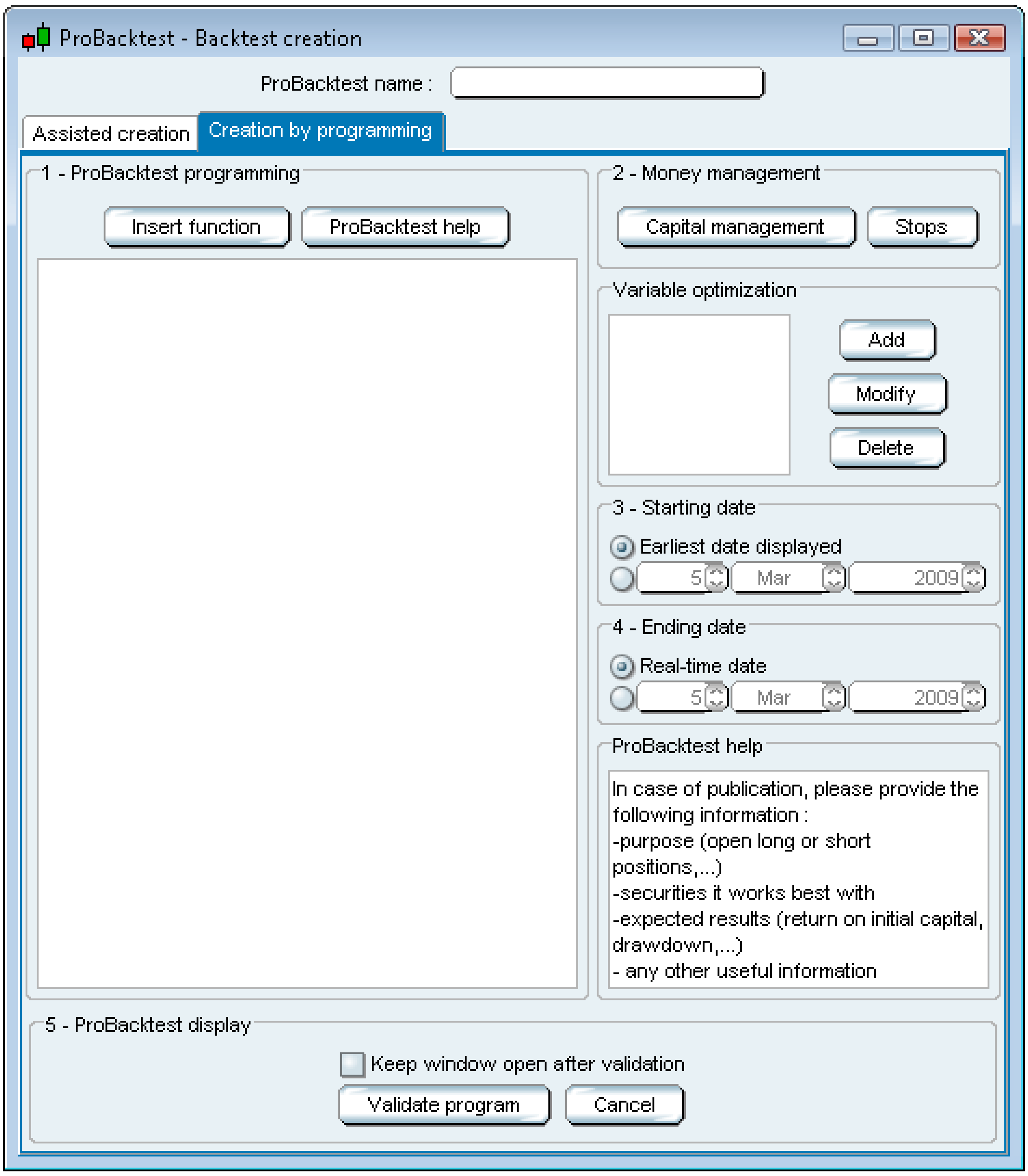
To create a new backtest, click “Create ProBacktest”. You will then be able to create a backtest with the assisted creation wizard (no programming required) or create a backtest by programming.

ProBacktest setup window sections

Lets concentrate on creation by programming by clicking on the appropriate tab.

The window is composed of 4 sections that can be setup:

- Programming of ProBacktest
- Money Management
- Variable Optimization
- Beginning and ending date



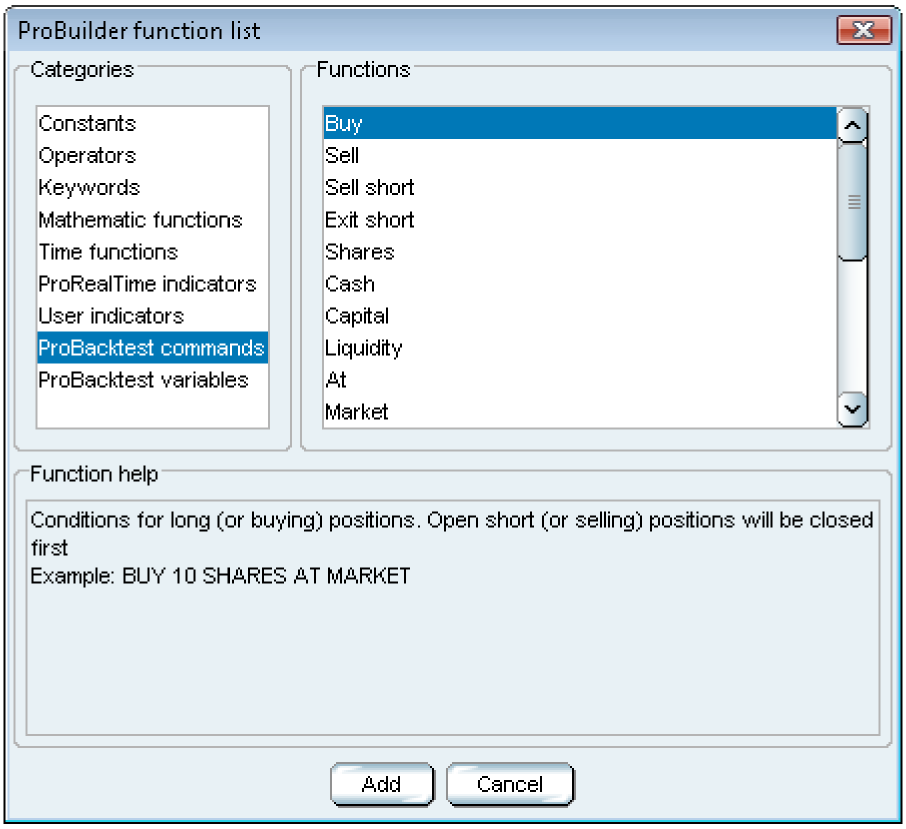
In case of publication, please provide the following information :
-purpose (open long or short positions,...)
-securities it works best with
-expected results (return on initial capital, drawdown,...)
- any other useful information

The **first** section allows you to:

- Program your ProBacktest directly in the text zone or
- Use the "Insert function" button which allows you to open a new window with a list of ProBuilder and ProBacktest commands separated into 9 categories to give you contextual help while programming. You can see a help text related to the command or function selected in the lower part of the window.

Let's use the function library by clicking on "Insert Function".

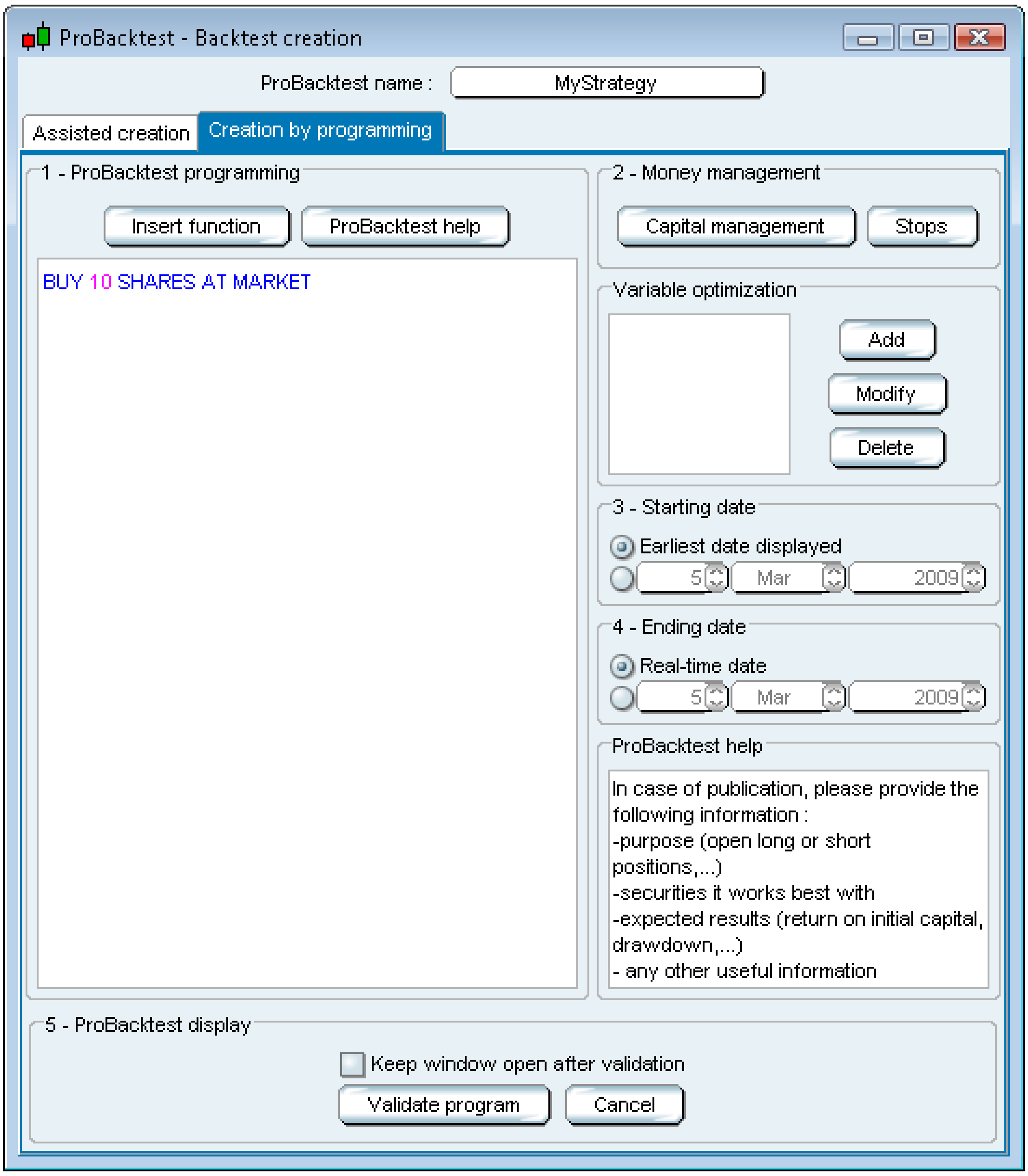
Choose the section "ProBacktest Commands" and click "**BUY**", then click the button "Add". The command will be inserted into your program.



Conditions for long (or buying) positions. Open short (or selling) positions will be closed first
Example: BUY 10 SHARES AT MARKET

Let's create a backtest. Suppose we want toy bu 10 shares at market price.

Proceed as above to find the functions "**SHARES**", "**AT**" and "**MARKET**" (separating each word with a space). Specify between "**BUY**" and "**SHARES**" the number to buy (10). Then give a name to your backtest: in this example, we have named it "MyStrategy".



ProBacktest name : MyStrategy

```
BUY 10 SHARES AT MARKET
```

3 - Starting date: (●) Earliest date displayed; ( ) 5 Mar 2009

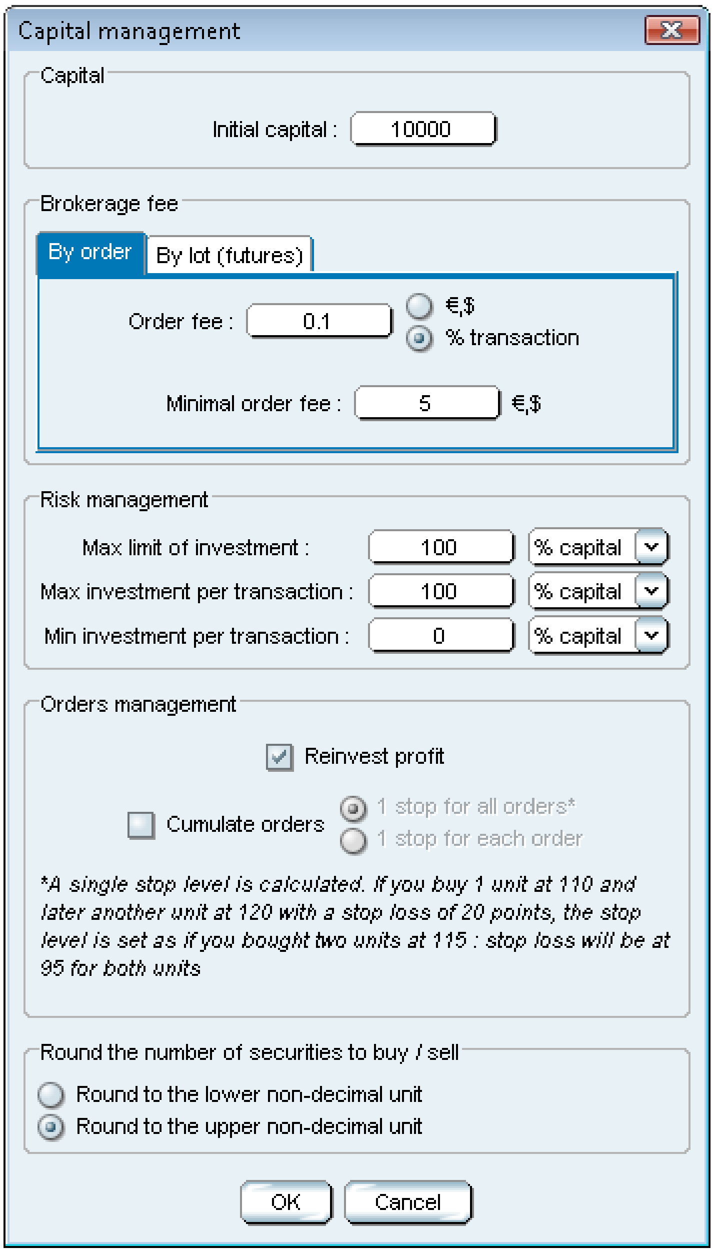
4 - Ending date: (●) Real-time date; ( ) 5 Mar 2009

ProBacktest help:
In case of publication, please provide the following information :
-purpose (open long or short positions,...)
-securities it works best with
-expected results (return on initial capital, drawdown,...)
- any other useful information

☐ Keep window open after validation

The **second** section (Money Management), allows you to setup the cost of trading, the capital to invest and your stops.

- In "Capital Management", ,you can define your initial capital for your Backtest, your brokerage fees and the method by which you are charged (by lot or by order) for these fees, your risk management and position management strategy.
- In “Stops”, you can choose to include or not 4 different types of stops: Stop losses, Profit stops (take-profit), Trailing stops and Inactivity stops.



Capital management

Capital

Initial capital : 10000

Brokerage fee

[✓ By order] By lot (futures)

Order fee : 0.1 ○ €,$ [✓ % transaction]

Minimal order fee : 5 €,$

Risk management

Max limit of investment :	100	% capital
Max investment per transaction :	100	% capital
Min investment per transaction :	0	% capital

Orders management

✓ Reinvest profit

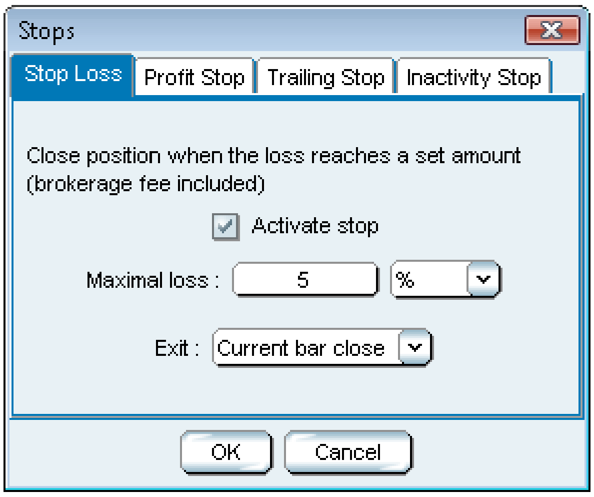
☐ Cumulate orders [✓ 1 stop for all orders*] ○ 1 stop for each order

**A single stop level is calculated. If you buy 1 unit at 110 and later another unit at 120 with a stop loss of 20 points, the stop level is set as if you bought two units at 115 : stop loss will be at 95 for both units*

Round the number of securities to buy / sell

○ Round to the lower non-decimal unit

[✓ Round to the upper non-decimal unit]



Stops

[✓ Stop Loss] Profit Stop Trailing Stop Inactivity Stop

Close position when the loss reaches a set amount (brokerage fee included)

✓ Activate stop

Maximal loss : 5 %

Exit : Current bar close

For more details concerning capital management, read the annexes at the end of the manual (page 37).

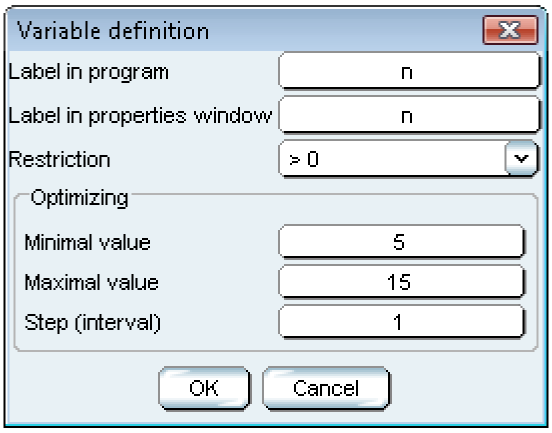
The third **section** allows you to optimize variables. This function allows you to test different combinations of values for the variables to know which give the best performance in your simulation.

The result of the optimization is presented in an "Optimization report". You will learn the results of each value tested and determine which combination of variables would best optimize your strategy.

Here is an example strategy where we will optimize the moving average number of periods n and m:

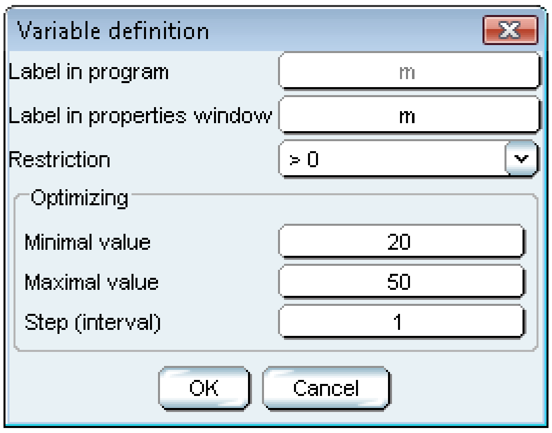
```
Indicator1 = Momentum[n](Close)
Indicator2 = Average[m](Indicator1)
IF Indicator1 CROSSES OVER Indicator2 THEN
        BUY 1 SHARES AT MARKET THISBARONCLOSE
ENDIF
IF Indicator1 CROSSES UNDER Indicator2 THEN
        SELLSHORT 1 SHARES AT MARKET THISBARONCLOSE
ENDIF
```

We define the variables to be optimized in the following way:



Variable definition

Label in program	n
Label in properties window	n
Restriction	> 0
Optimizing	
Minimal value	5
Maximal value	15
Step (interval)	1

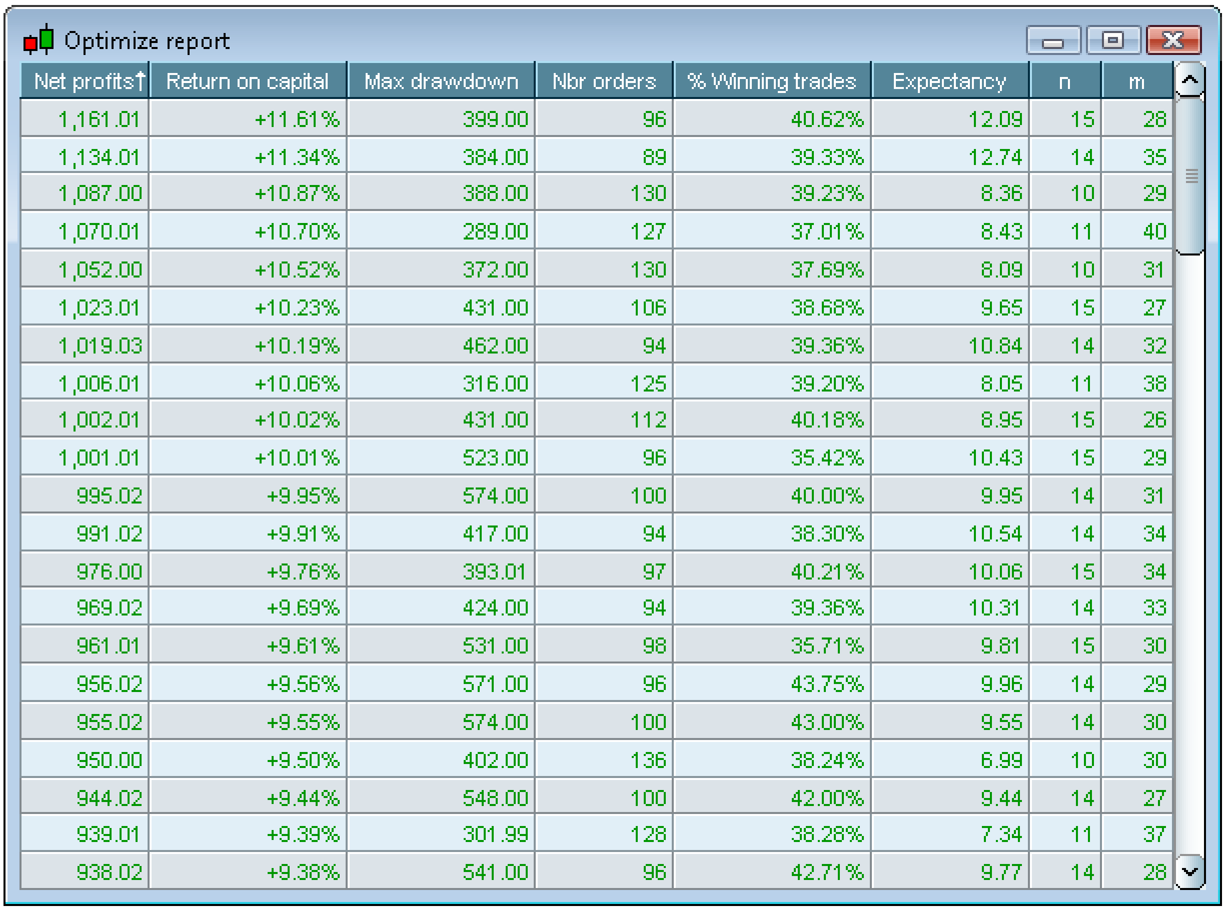


Variable definition

Label in program	m
Label in properties window	m
Restriction	> 0
Optimizing	
Minimal value	20
Maximal value	50
Step (interval)	1

- **"Name in the program"** is the name of the variable in the program (here n and m). This variable is case sensitive (upper case or lower case).
- **"Label in the properties window"** is the name attributed to the variable that will be shown in the properties window of the backtest to make it more easy to recognize (for example: "Number of periods" for n).
- **"Minimum value" and "maximum value"** are the highest and lowest values of the variable to be tested in the optimization tests.
- **"Step"** defines the interval of values to test in the optimization process.

Here is an example of an optimization report :



Net profits↑	Return on capital	Max drawdown	Nbr orders	% Winning trades	Expectancy	n	m
1,161.01	+11.61%	399.00	96	40.62%	12.09	15	28
1,134.01	+11.34%	384.00	89	39.33%	12.74	14	35
1,087.00	+10.87%	388.00	130	39.23%	8.36	10	29
1,070.01	+10.70%	289.00	127	37.01%	8.43	11	40
1,052.00	+10.52%	372.00	130	37.69%	8.09	10	31
1,023.01	+10.23%	431.00	106	38.68%	9.65	15	27
1,019.03	+10.19%	462.00	94	39.36%	10.84	14	32
1,006.01	+10.06%	316.00	125	39.20%	8.05	11	38
1,002.01	+10.02%	431.00	112	40.18%	8.95	15	26
1,001.01	+10.01%	523.00	96	35.42%	10.43	15	29
995.02	+9.95%	574.00	100	40.00%	9.95	14	31
991.02	+9.91%	417.00	94	38.30%	10.54	14	34
976.00	+9.76%	393.01	97	40.21%	10.06	15	34
969.02	+9.69%	424.00	94	39.36%	10.31	14	33
961.01	+9.61%	531.00	98	35.71%	9.81	15	30
956.02	+9.56%	571.00	96	43.75%	9.96	14	29
955.02	+9.55%	574.00	100	43.00%	9.55	14	30
950.00	+9.50%	402.00	136	38.24%	6.99	10	30
944.02	+9.44%	548.00	100	42.00%	9.44	14	27
939.01	+9.39%	301.99	128	38.28%	7.34	11	37
938.02	+9.38%	541.00	96	42.71%	9.77	14	28

The report includes 6 statistics for each combination of variables tested (here: n and m). These statistics are as follows:

- **Net Profits**, shows the gain from trades. Mathematically, profit is equal to:

  Net Profits = Capital at the end of the simulation – Initial Capital

  This statistic allows you to evaluate the potential gains of the strategy defined (for each value of the variable tested).

  Note: the brokerage fees defined in the "Capital management" section are taken into account in this calculation.

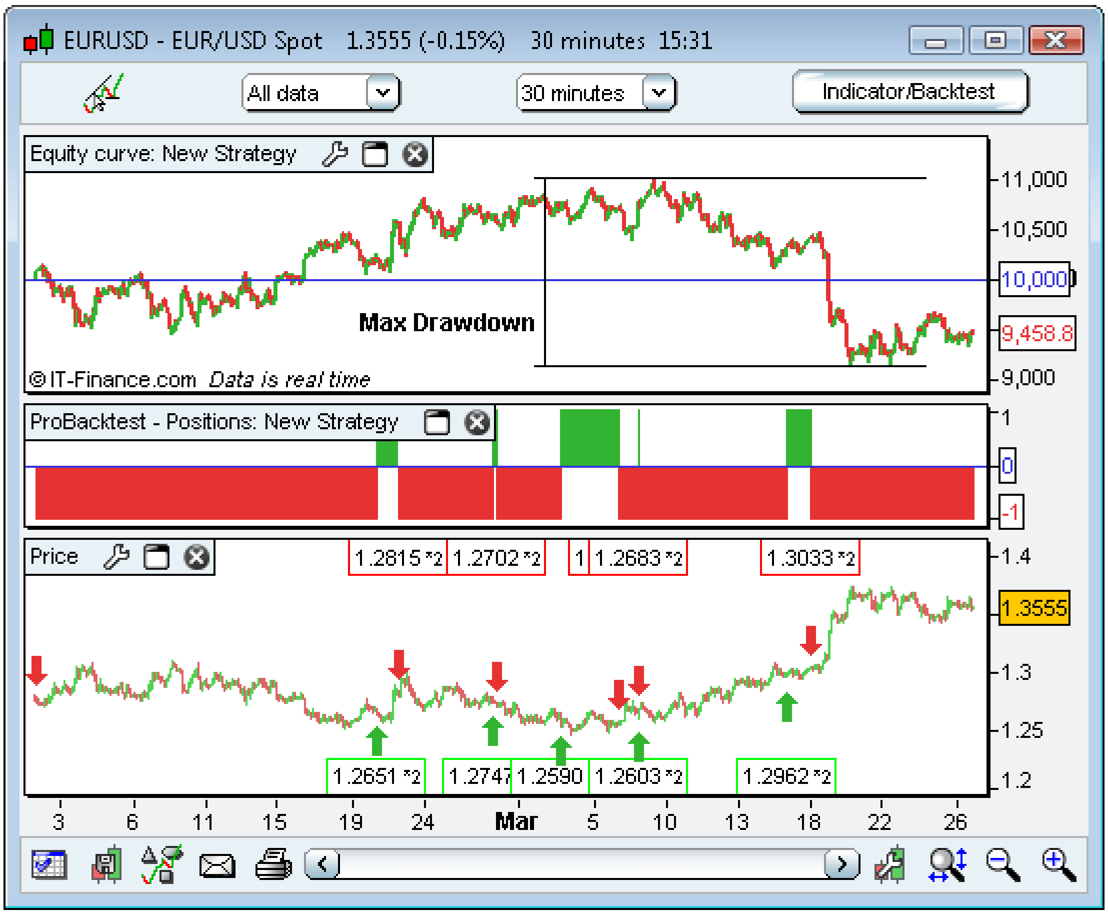
➡ **Return on capital**, is the Profit in percentage. The formula is :

Return on capital = (100 x Net Profits) / Initial Capital

It indicates the relative performance of the strategy simulated with the different variable values tested.

➡ **Max drawdown**, designates the maximum loss at any given point in the simulation measured by the difference between the highest level of the equity curve and the lowest subsequent point.

Let's look at an example of maximum drawdown on this chart:



The max drawdown of the strategy can be considered as a measure of riskiness of the trading strategy: if you are not ready to accept the risk of losing the max drawdown amount, you could choose a different strategy with less risk.

➡ **Nbr orders**, indicates the number of orders since the beginning of the strategy simulation.

➡ **% Winning trades** shows the % of winning trades in the simulation and is also an indicator of riskiness complementary to **Max drawdown**. Mathematically, the formula is:

% Winning trades = (100 x Number of winning trades) / Number of total trades

➡ **Expectancy** is the average gain per trade and is useful to determine the average efficiency of each trade. Expectancy is particularly important if you want to limit the number of orders in the strategy. In this case, it can be an important factor in the decision to apply the strategy or not;

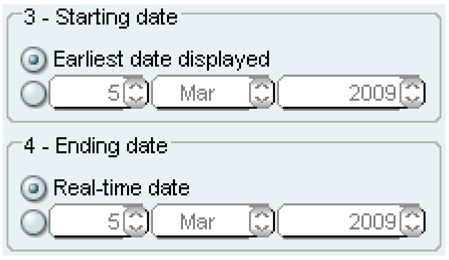
Mathematically, the formula is:

Expectancy = Net Profit / Number of Trades

Note: the optimal values of variables for a strategy may be different for the same security depending on the unit of time used in the chart and the historical dataset you are looking at.

The **fourth** section allows you to define the period of time you will backtest your strategy over. The beginning date corresponds to the beginning date of the strategy and the end date is the date at which all of your remaining positions will be liquidated if you choose a date other than the real-time date.

This function of ProBacktest is configured by default to test your strategy over all the historical data displayed (in this case, open positions are closed only when the exit conditions are verified).



3 - Starting date

[✓ Earliest date displayed] 5 Mar 2009

4 - Ending date

[✓ Real-time date] 5 Mar 2009

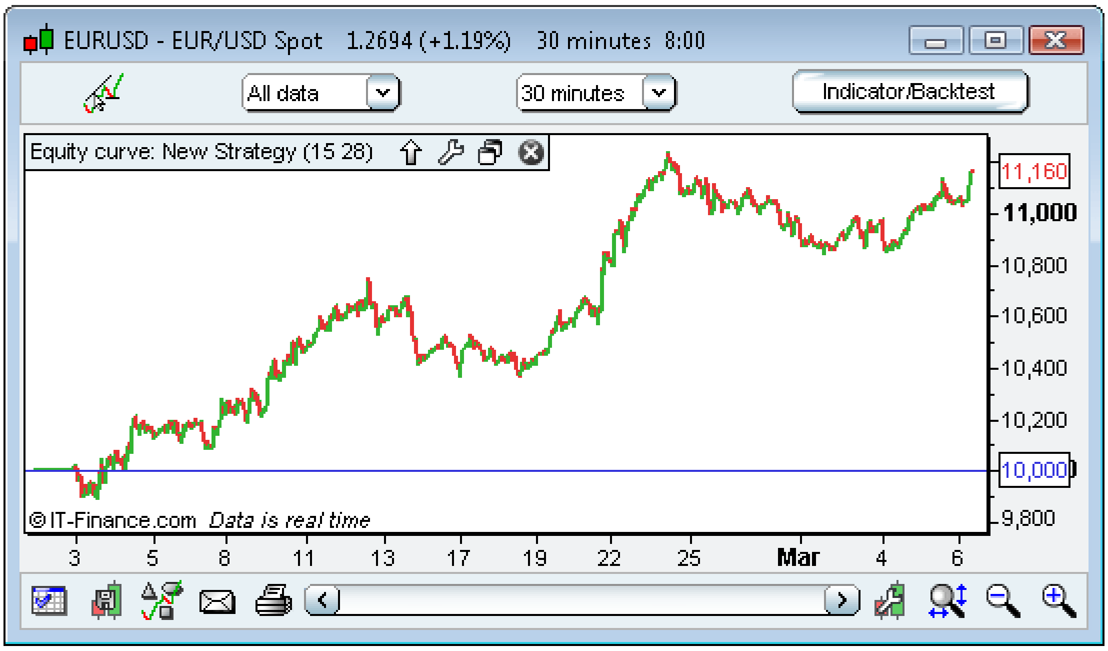
➡ ProBacktest Results

In addition to the optimization report presented above, ProBacktest displays results in 3 complementary forms.

1) Equity Curve

The Equity Curve shows the evolution of the initial invested capital (defined in the capital management section) since the beginning of your strategy simulation.

The color of the equity curve will be green to indicate a positive variation from the previous level or red to indicate a negative variation from the previous level.


EURUSD - EUR/USD Spot 1.2694 (+1.19%) 30 minutes 8:00
All data
30 minutes
Indicator/Backtest
Equity curve: New Strategy (15 28)
11,160
11,000
10,800
10,600
10,400
10,200
10,000
9,800
3 5 8 11 13 17 19 22 25 Mar 4 6
© IT-Finance.com Data is real time

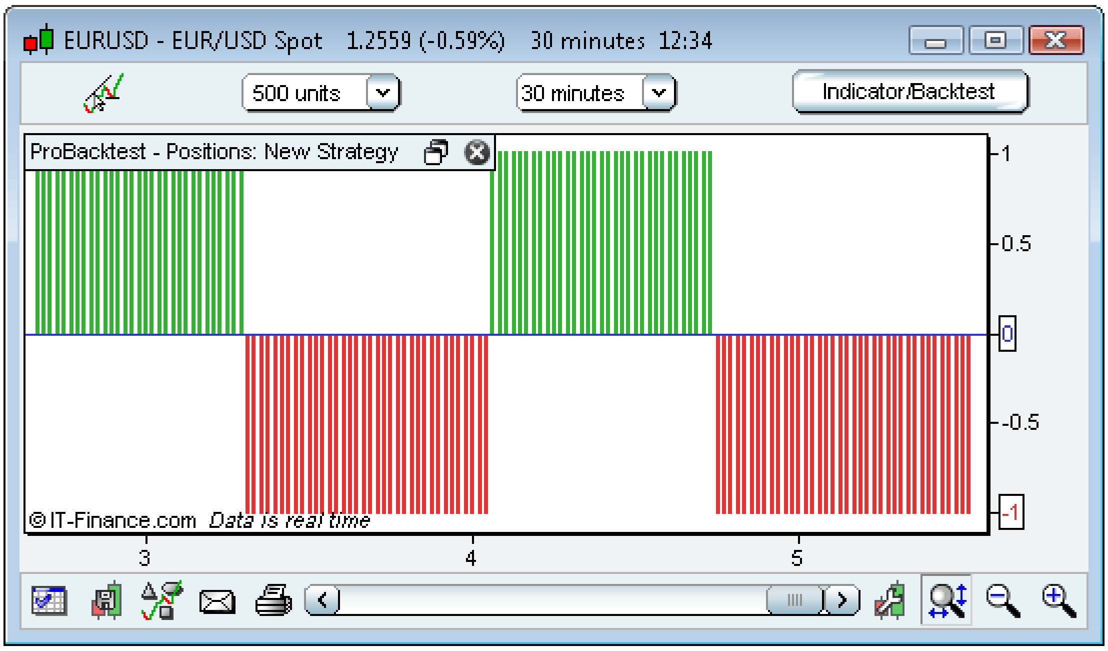
2) Positions histogram

The positions histogram allows you to show in histogram form the evolution of your positions during the strategy simulation.

- A green bar indicates an open long position.
- A red bar indicates an open short position.
- No bar indicates no open position.

Several consecutive bars of the same color indicate the position(s) is still open.

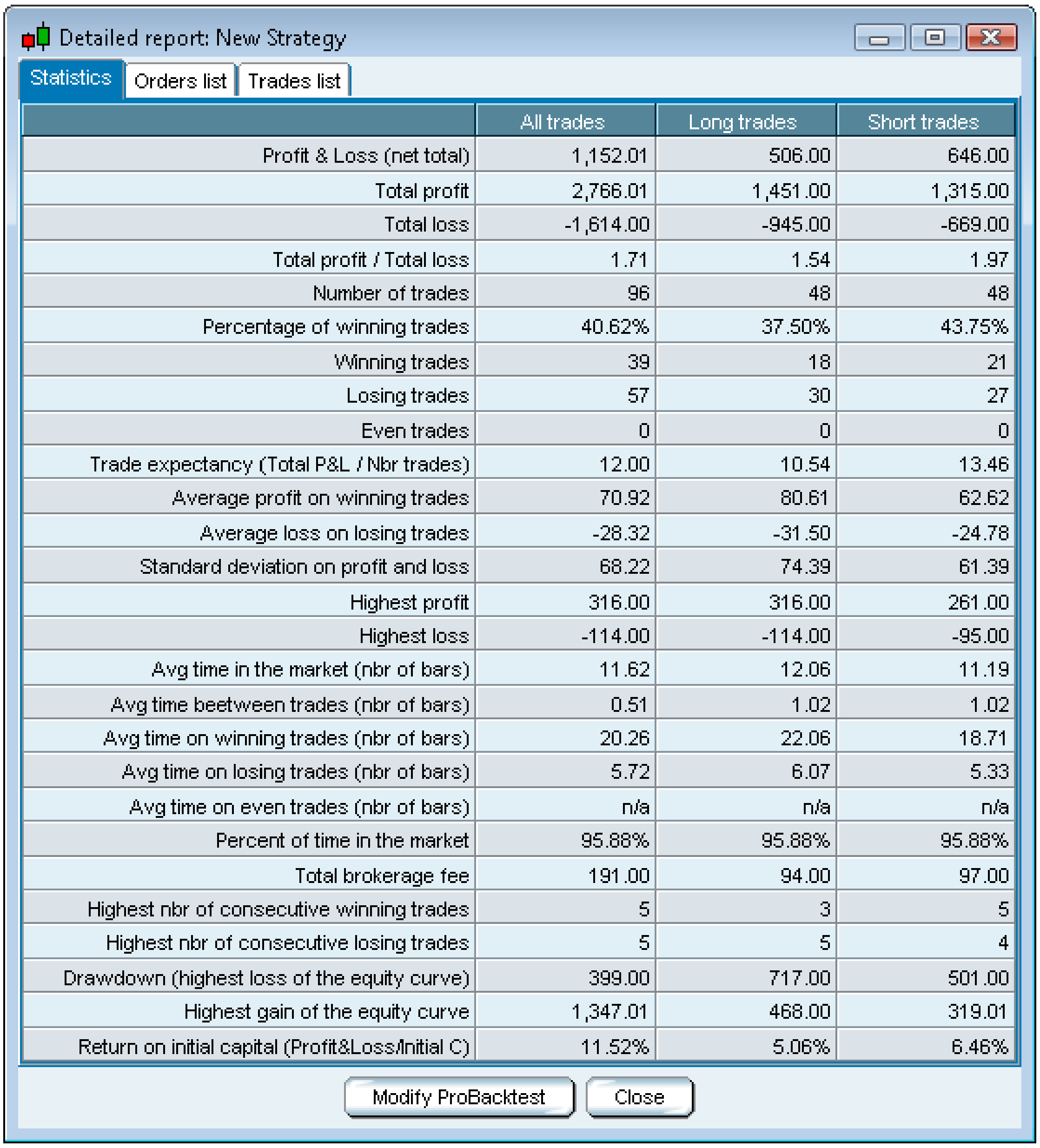
On the vertical axis on the right-side of the chart, you will see how many positions you have open currently (highlighted). In the example below, there is currently 1 short position open.


EURUSD - EUR/USD Spot 1.2559 (-0.59%) 30 minutes 12:34
500 units
30 minutes
Indicator/Backtest
ProBacktest - Positions: New Strategy
1
0.5
0
-0.5
-1
3 4 5
© IT-Finance.com Data is real time

3) Detailed report

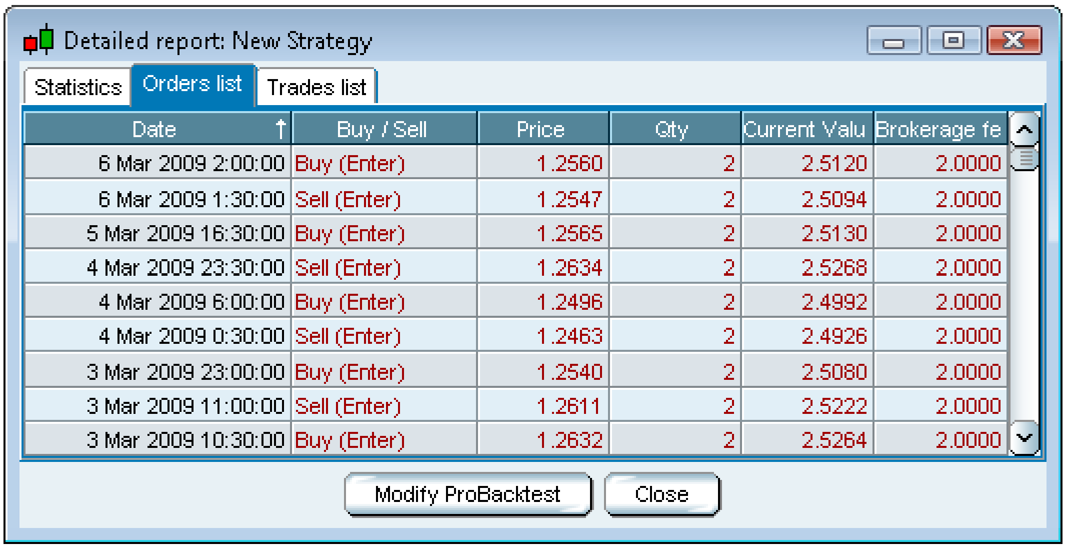
The detailed report allows you to view the statistics of your strategy in terms of performance, length of positions and list of orders. The detailed report is shown in an independent window made up of 3 tabs:

- In **Statistics**, you will get an exhaustive view of the performance of your strategy (net gains or loss, number of winning and losing trades, and many other statistics…). Beyond the classical statistics displayed, the statistics “Highest Profit” and “Highest Loss” and “Max Drawdown” may help determine if this strategy is acceptable to you in terms of risk.



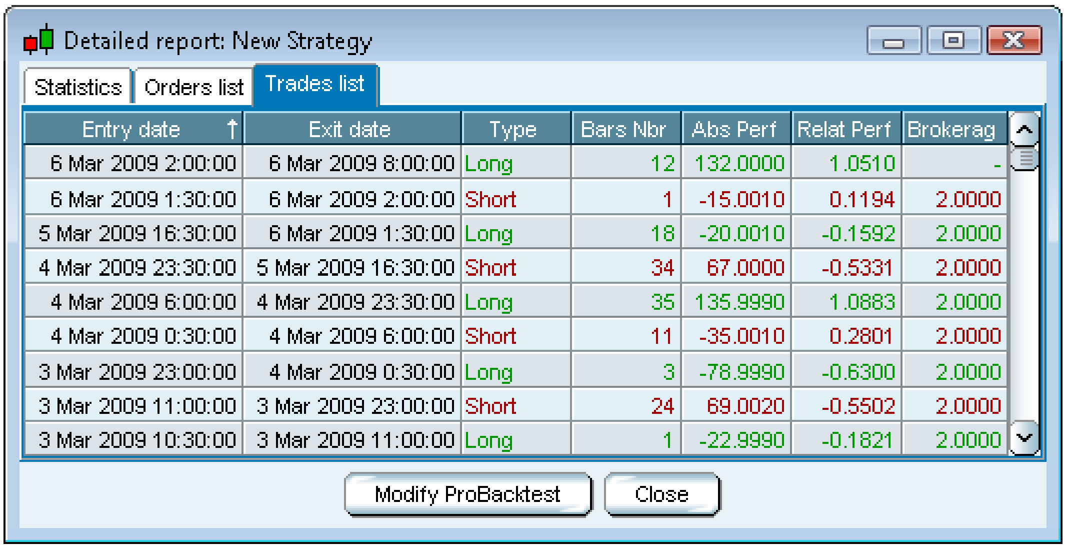
	All trades	Long trades	Short trades
Profit & Loss (net total)	1,152.01	506.00	646.00
Total profit	2,766.01	1,451.00	1,315.00
Total loss	-1,614.00	-945.00	-669.00
Total profit / Total loss	1.71	1.54	1.97
Number of trades	96	48	48
Percentage of winning trades	40.62%	37.50%	43.75%
Winning trades	39	18	21
Losing trades	57	30	27
Even trades	0	0	0
Trade expectancy (Total P&L / Nbr trades)	12.00	10.54	13.46
Average profit on winning trades	70.92	80.61	62.62
Average loss on losing trades	-28.32	-31.50	-24.78
Standard deviation on profit and loss	68.22	74.39	61.39
Highest profit	316.00	316.00	261.00
Highest loss	-114.00	-114.00	-95.00
Avg time in the market (nbr of bars)	11.62	12.06	11.19
Avg time beetween trades (nbr of bars)	0.51	1.02	1.02
Avg time on winning trades (nbr of bars)	20.26	22.06	18.71
Avg time on losing trades (nbr of bars)	5.72	6.07	5.33
Avg time on even trades (nbr of bars)	n/a	n/a	n/a
Percent of time in the market	95.88%	95.88%	95.88%
Total brokerage fee	191.00	94.00	97.00
Highest nbr of consecutive winning trades	5	3	5
Highest nbr of consecutive losing trades	5	5	4
Drawdown (highest loss of the equity curve)	399.00	717.00	501.00
Highest gain of the equity curve	1,347.01	468.00	319.01
Return on initial capital (Profit&Loss/Initial C)	11.52%	5.06%	6.46%

➡ In the **"Orders list"** you will find details on the time, the direction (buy or sell), the quantity and the price of orders. The order times displayed will be in local market time.



Date	Buy / Sell	Price	Qty	Current Valu	Brokerage fe
6 Mar 2009 2:00:00	Buy (Enter)	1.2560	2	2.5120	2.0000
6 Mar 2009 1:30:00	Sell (Enter)	1.2547	2	2.5094	2.0000
5 Mar 2009 16:30:00	Buy (Enter)	1.2565	2	2.5130	2.0000
4 Mar 2009 23:30:00	Sell (Enter)	1.2634	2	2.5268	2.0000
4 Mar 2009 6:00:00	Buy (Enter)	1.2496	2	2.4992	2.0000
4 Mar 2009 0:30:00	Sell (Enter)	1.2463	2	2.4926	2.0000
3 Mar 2009 23:00:00	Buy (Enter)	1.2540	2	2.5080	2.0000
3 Mar 2009 11:00:00	Sell (Enter)	1.2611	2	2.5222	2.0000
3 Mar 2009 10:30:00	Buy (Enter)	1.2632	2	2.5264	2.0000

➡ Finally, "**Trades list**" gives details about the positions taken during the simulation (long or short, duration expressed in number of bars, absolute and relative performance, entry and exit dates...).



Entry date	Exit date	Type	Bars Nbr	Abs Perf	Relat Perf	Brokerag
6 Mar 2009 2:00:00	6 Mar 2009 8:00:00	Long	12	132.0000	1.0510	-
6 Mar 2009 1:30:00	6 Mar 2009 2:00:00	Short	1	-15.0010	0.1194	2.0000
5 Mar 2009 16:30:00	6 Mar 2009 1:30:00	Long	18	-20.0010	-0.1592	2.0000
4 Mar 2009 23:30:00	5 Mar 2009 16:30:00	Short	34	67.0000	-0.5331	2.0000
4 Mar 2009 6:00:00	4 Mar 2009 23:30:00	Long	35	135.9990	1.0883	2.0000
4 Mar 2009 0:30:00	4 Mar 2009 6:00:00	Short	11	-35.0010	0.2801	2.0000
3 Mar 2009 23:00:00	4 Mar 2009 0:30:00	Long	3	-78.9990	-0.6300	2.0000
3 Mar 2009 11:00:00	3 Mar 2009 23:00:00	Short	24	69.0020	-0.5502	2.0000
3 Mar 2009 10:30:00	3 Mar 2009 11:00:00	Long	1	-22.9990	-0.1821	2.0000

CHAPTER II: PROGRAMMING PROBACKTEST

Entering and exiting the market

2 categories of instruction allow you to enter and exit the market: position entry and exit instructions and stop instructions.

1) Position entry and exit commands

Different instructions are used depending on the type of position:

- Long positions
  - **BUY** enter long position instruction (buy securities)
  - **SELL** exit long position instruction (sell securities)
- Short positions
  - **SELLSHORT** enter short position instruction (short sell securities)
  - **EXITSHORT** exit long position instruction (buy back shorted securities)

ProBacktest does not allow simulation of "hedging", meaning taking simultaneous short and long positions on the same security. For this reason, the SELLSHORT command will first close any open long position before opening a short position and the BUY command will first close any short positions before opening a long position. It is advised to use the recommended commands to close positions (SELL for closing long positions and EXITSHORT for closing short positions).

Each command above may be followed by one or more of the following elements which we will describe:

SELLSHORT "Number" "Mode" AT "TYPE" "DATE/TIME"

a. Number

This is the quantity you want to buy or sell.

Note: It is possible to not insert a number. In this case, the program will consider the quantity to be 1 stock or one lot.

b. Mode

You can define the mode in which to buy or sell in absolute or relative terms.

- **SHARES** transaction defined in number of stocks or contracts
- **CASH** transaction defined in cash (ex: n € or $ worth of the security)
- **% CAPITAL** transaction defined in percent of capital (ex : 10% of capital)
- **% LIQUIDITY** transaction defined in percent of remaining liquidity

Example :

Buy (enter long) for 10% of capital when the RSI is oversold (RSI < 30) and the price is below the lower Bollinger band.

Sell (exit long) when the RSI is overbought (RSI > 70) and price is above the upper Bollinger band.

```
IF RSI[14](Close) < 30 AND Close < BollingerDown[25](Close) THEN
        BUY 10 %CAPITAL AT MARKET
ENDIF
IF RSI[14](Close) > 70 AND Close > BollingerUp[25](Close) THEN
        SELL 10 %CAPITAL AT MARKET
ENDIF
```

c. Type

Three order types are available:

- **AT MARKET** : The order will be executed at market price
- **AT** (price) **LIMIT** : The order will be executed at the indicated price
- **AT** (price) **STOP** : The order will be executed at the indicated price

Example : Volatility Breakout

This strategy is based on volatility.

On each bar, a BUY AT LIMIT and a SELLSHORT AT STOP order are placed.

The BUY order is placed on the close of the previous bar plus 50% of the range of the previous bar (Range = High – Low).
The SELLSHORT order is placed at the close of the previous bar minus 50% of the range of the previous bar.

```
REM Volatility Breakout
BuyLimit = Close[1] + (Range[1] * 50 / 100)
SellLimit = Close[1] - (Range[1] * 50 / 100)
BUY 1 SHARES AT BuyLimit Stop
SELLSHORT 1 SHARES AT SellLimit Stop
```

d. Date/time of execution

By default, AT MARKET orders are executed on the open of the **next bar**. In the case of AT MARKET orders, it is possible to set the time of execution using the following commands. Parenthesis and brackets are not used with these commands.

- **NextBarOpen** : places the order on the open of the next bar (**default**)
- **NextBarClose** : places the order on the close of the next bar
- **ThisBarOnClose** : places the order on the close of the current bar
- **TodayOnClose** : places the order on the close of the current day (relevant only if used with an intraday timeframe)
- **TomorrowOpen** : places the order on the open of the next day (relevant only if used with an intraday timeframe)
- **TomorrowClose** : places the order on the close of the next day (relevant only if used with an intraday timeframe)
- **RealTime** : places the order in real-time (on the current tick)

The "Date/time of execution" commands are usable only when the instruction "**AT MARKET**" precedes them.

Example : Channel breakout

We define the resistance and support of the channel as the highest and lowest points of the two first bars of the trading day. If before 16H00, price breaks the resistance of the channel, we open a long position with 70% of our capital. We close any long positions and open a short position if price beaks the support of the channel before 16H00.

```
REM Close of the second bar (IntradayBar index=1)
IF IntradayBarIndex = 1 THEN
        Resist = Highest[2](High)
        Support = Lowest[2](Low)
ENDIF
REM Buy / Short on breakout if before 16H00:00 (Local market time)
IF IntradayBarIndex > 1 AND Time < 160000 THEN
        REM Resistance breakout
        IF Close > Resist THEN
                BUY 70 %CAPITAL AT MARKET THISBARONCLOSE
        ENDIF
        REM Support breakout
        IF Close < Support THEN
                SELLSHORT 70 %CAPITAL AT MARKET THISBARONCLOSE
        ENDIF
ENDIF
```

2) STOP commands

It is possible to manually program STOPs in your ProBacktest strategy.

In addition to the 4 types of predefined stops in section 2 in the ProBacktest window (see chapter one), you can also insert stops which you program yourself. The command to create a stop in the program is:

In addition to the 4 types of predefined stops in section 2 in the ProBacktest window (see the "Money Management" section), can also insert stops which you program yourself.

The command to create a stop in the program is:

SET STOP (price)

where the constant "price" designates the level at which the position will be closed.

Example: Price and Parabolic SAR cross strategy

The following strategy will place a buy (or shortsell) order when the price crosses over (or under) the SAR.

A trailing stop is set to exit the position when the price hits a certain level (called cut in the program).

```
Indicator1 = Close
Indicator2 = SAR[0.02,0.02,0.2]
REM Buy
c1 = (Indicator1 Crosses Over Indicator2)
IF c1 THEN
        BUY 1 SHARES AT MARKET
ENDIF
REM Sell
c2 = (Indicator1 Crosses Under Indicator2)
IF c2 THEN
        SELLSHORT 1 SHARES AT MARKET
ENDIF
REM placing the trailing stop
IF Lowest[5](Close)< (1.2 * Low) THEN
        IF Lowest[5](Close) >= Close THEN
                Cut = Lowest[5](Close)
        ELSE
                Cut = Lowest[20](Close)
        ENDIF
ENDIF
SET STOP Cut
```

Note the difference between **STOP** commands:

- **AT (price) STOP**, is used to enter a short position when a certain price level is reached.
- **SET STOP (price), is used to define a protection stop (to exit an open position).**

Position verification commands

1) Commands verifying the type of positions open

ProBacktest allows you to create conditions for placing an order based on whether or not you currently have open long or short positions in the current ProBacktest simulation.

Here are the commands that let you check the status of your current positions:

- **ONMARKET** : checks if there are positions open
- **LONGONMARKET** : checks if there are long positions open
- **SHORTONMARKET** : checks if there are short positions open

They are used without parenthesis or brackets and are **usually preceded by the IF command**.

The commands checking the state of positions open are particularly interesting if you want to cumulate or pyramid positions (see page 29). These positions must be in the same direction (long or short). It is not possible to cumulate positions in different directions during a backtest simulation.

Here is an example of how these commands might be used.

Example: MACD Strategy (use of LONGONMARKET and SHORTONMARKET) :

This strategy is based on the changes in the sign of the MACD histogram (positive or negative state). It will take a certain number of positions which will be progressively closed. This progressive advance has the goal of locking in gains and limiting risk.

```
REM We give the value of the MACD to the variable Indicator1
Indicator1 = MACD[12,26,9](Close)
REM We observe the changes in sign of the MACD
c1 = (Indicator1 Crosses Over 0.0)
REM Buy : If we do not have an open long position and MACD > 0, we buy 3 lots
IF NOT LONGONMARKET AND c1 THEN
        BUY 3 SHARES AT MARKET ThisBarOnClose
        Long = 0
        Entry = Close
ENDIF
REM Sell : of our 3 lots, we sell successively at 7, 15 and 25% profit if possible.
REM We close the remaining positions when MACD crosses under 0.
IF LONGONMARKET AND Long = 0 AND Close > (Entry * 1.07) THEN
```

```
        SELL 1 SHARES AT MARKET ThisBarOnClose
        Long = 1
ELSIF LONGONMARKET AND Long = 1 AND Close > (Entry * 1.15) THEN
        SELL 1 SHARES AT MARKET ThisBarOnClose
        Long = 2
ELSIF LONGONMARKET AND Long = 2 AND Close > (Entry * 1.25) THEN
        SELL 1 SHARES AT MARKET ThisBarOnClose
        Long = 3
ENDIF
REM SHORT: If we do not have an open short position and MACD < 0, we sell short 3 lots
IF NOT c1 AND NOT SHORTONMARKET THEN
        SELLSHORT 3 SHARES AT MARKET ThisBarOnClose
        Short = 0
        Entry = Close
ENDIF
REM EXIT SHORT: Of the 3 lots, we successively buy back at profits of 7, 15 and 25% profit
REM if possible. We close the remaining positions when MACD crosses under 0.
IF SHORTONMARKET AND Short = 0 AND Close < (Entry * 0.93) THEN
        EXITSHORT 1 SHARES AT MARKET ThisBarOnClose
        Short = 1
ELSIF SHORTONMARKET AND Short = 1 AND Close < (Entry * 0.85) THEN
        EXITSHORT 1 SHARES AT MARKET ThisBarOnClose
        Short = 2
ELSIF SHORTONMARKET AND Short = 2 AND Close < (Entry * 0.75)  THEN
        EXITSHORT 1 SHARES AT MARKET ThisBarOnClose
        Short = 3
ENDIF
```

2) Position Counters

The following commands allow users to build strategies that take into account the number of positions entered since the beginning of the simulation (long or short).

- **COUNTOFPOSITION** : number of positions taken since the beginning of the backtest
- **COUNTOFLONGSHARES** : number of LONG positions since the beginning of the backtest
- **COUNTOFSHORTSHARES** : number of SHORT positions since the beginning of the backtest

Similar to commands verifying the type of positions open, these commands are **usually preceded by the IF command**.

Below is an example Backtest using COUNTOFLONGSHARES and COUNTOFSHORTSHARES.

Example: Inverse Fisher Transform applied to RSI.

This Backtest is based on the "Inverse Fisher Transform RSI" to place buy or sell orders.

The system enters long when Inverse Fisher Transform RSI crosses over 50 and exits long when Inverse Fisher Transform RSI crosses under 80.

It enters short when Inverse Fisher Transform RSI crosses under 50 and exits short when Inverse Fisher Transform RSI crosses over 20.

This backtest was designed for use with futures in a 1h view or stocks in a daily view.

```
REM Inverse Fisher Transform Applied to RSI
REM Parameters : n = Number of bars for the calculation of the RSI
n = 10
Ind=RSI[n](Close)
x = 0.1 * (Ind - 50)
y = (EXP (2 * x) - 1) / (EXP (2 * x) + 1)
z = 50 * (y + 1)
myInverseFisherTransformsRSI = z[7]
IF (myInverseFisherTransformsRSI Crosses Over 50) THEN
        BUY 1 SHARES AT MARKET
ENDIF
IF (myInverseFisherTransformsRSI Crosses Under 80) THEN
        SELL COUNTOFLONGSHARES SHARES AT MARKET
ENDIF
IF (myInverseFisherTransformsRSI Crosses Under 50) THEN
        SELLSHORT 1 SHARES AT MARKET
ENDIF
IF (myInverseFisherTransformsRSI Crosses Over 20) THEN
        EXITSHORT COUNTOFSHORTSHARES SHARES AT MARKET
ENDIF
```

3) ENTRYINDEX

The command **ENTRYINDEX[n]** allows you to access the index of the bar of the nth previous transaction.

The command has the same characteristics as **BarIndex** and **IntradayBarIndex** (introduced in the ProBuilder manual) :

- The bars are numbered from the first to last loaded (left to right)
- The first bar has an index of 0. Example: If ENTRYINDEX[0] has a value of 3; it means that the last order created was done on the fourth bar since the beginning of the historical data loaded.

The syntax is like the syntax for a constant:

ENTRYINDEX[nth previous order]

Note: It is possible to use ENTRYINDEX without brackets following it. In this case, the program will consider the BarIndex of the last order created.

Example : Inside bar strategy

The following example is a strategy based on a common price pattern called "Inside Bar" based on 2 candlestick forms:

- The first form occurs if the range of the 2nd candle preceding the current candle is greater than the range of the candle preceding the current candle. The candle preceding the current candle must be white (close > open). In this case, a long position is taken.
- The second form occurs if the range of the 2nd candle preceding the current candle is lower than the range of the candle preceding the current candle and the candle preceding the current one is black (close < open). In this case we take a short position.

The position exit in this system is systematically 3 bars after the position is opened.

```
Condition1 = (High[2] >= High[1] AND Low[2] <= Low[1])
Condition2 = (High[2] <= High[1] AND Low[2] <= Low[1])
Condition3 = (Close[1] > Open[1])
Condition4 = (Close[1] < Open[1])
IF (Condition1 AND Condition3) THEN
        BUY 10 %CAPITAL AT MARKET NextBarOpen
ENDIF
IF LONGONMARKET AND (BarIndex - ENTRYINDEX) = 3 THEN
        SELL 10 %CAPITAL AT MARKET ThisBarOnClose
ENDIF
IF (Condition2 AND Condition4) THEN
        SELLSHORT 10 %CAPITAL AT MARKET NextBarOpen
ENDIF
IF SHORTONMARKET AND (BarIndex - ENTRYINDEX) = 3 THEN
        EXITSHORT AT MARKET ThisBarOnClose
ENDIF
```

4) ENTRYQUOTE

The command **ENTRYQUOTE[n]** allows you to call the price at which the nth previous transaction was executed. This is particularly useful when the length of time between trades is short (intraday strategies).

The syntax is as follows:

ENTRYQUOTE[nth previous order]

As for all constants, you can specify within the brackets the order which you are referring to. If you do not specify a number in brackets after ENTRYQUOTE, the price of the previous order is called.

Example : Creation of a take profit stop

We define 2 conditions:

- No open positions
- Low RSI(<30)

We buy when these conditions are true and the price crosses above the 10-period moving average.

We close the position when the price exceeds 15% above the ENTRYQUOTE in real-time (using the limit command).

```
IF NOT ONMARKET AND RSI < 30 THEN
        IF Close > AVERAGE[10](Close) THEN
                BUY 100 %CAPITAL AT MARKET
        ENDIF
ENDIF
SELL 100 %CAPITAL AT ENTRYQUOTE * 1.15 LIMIT
```

5) PreviousTrade

Some traders refer to the performance of their previous trade in the construction of their trading strategies. The command PreviousTrade(n) allows the construction and backtesting of this type of strategy.

The command returns the performance in % of the nth previous trade.

PreviousTrade(nth previous trade)

PreviousTrade(1) = Performance of the last trade executed. Note: the parenthesis are required.

Example :

Here is an example based on the crossings of stochastic and RSI lines. We first create a buy at market order based on a bullish exponential moving averages crossing and then we create another position :

- Buy (pyramiding) if :
  - The first trade has a positive performance
  - RSI is less than 30
- Sell if:
  - RSI > 70
  - Stochastic %K crosses under Stochastic %D

```
ONCE StochPeriod = 14
ONCE KPeriod = 3
ONCE DPeriod = 3
LineK = Stochastic[StochPeriod, KPeriod](Close)
LineD = Average[DPeriod](LineK)
//We place the first order if there is a bullish EMA crossing
IF ExponentialAverage[12](Close) Crosses Over ExponentialAverage[20](Close) THEN
        BUY AT MARKET
ENDIF
//We place the second order
IF LineK Crosses Over LineD THEN
        IF RSI < 30 THEN
                REM Buy if the previous trade has a positive result
                IF PreviousTrade(1) > 0 THEN
                        BUY 100 %LIQUIDITY AT MARKET
                ENDIF
        ENDIF
ENDIF
IF RSI > 70 AND LineK Crosses Under LineD THEN
        SELL 100 %CAPITAL AT MARKET
ENDIF
```

CHAPTER III: PRACTICAL APPLICATIONS

Indicator Strategies

1) Heiken Ashi strategy

This system generates a buy signal when a green Heiken Ashi candle appears after a red one. A sell signal is given if a red Heiken Ashi candle appears after a green one.

This backtest reconstructs the Heiken Ashi view from normal candlesticks. It must be applied to a chart using the normal candlestick style.

```
ONCE PreviousStatus = 0
IF BarIndex = 0 THEN
        XClose = TotalPrice
        XOpen = (Open + Close) / 2
ELSE
        XClose = TotalPrice
        XOpen = (XOpen[1] + Xclose[1]) / 2
ENDIF
IF XClose >= XOpen THEN
        IF PreviousStatus = -1 THEN
                BUY 1 SHARES AT MARKET
        ELSE
                PreviousStatus = 1
                IF PreviousStatus = 1 THEN
                        SELLSHORT 1 SHARES AT MARKET
                        PreviousStatus = -1
                ENDIF
        ENDIF
ENDIF
```

2) ZigZag Strategy

This is a backtest based on the zigzag to determine what would have been the best buy and sell opportunities. The excellent results of this strategy on both stocks and futures are related to the non-predictive character of the ZigZag. The signals are recalculated after the fact and as a result do not always give valid signals in real-time.

The reason the results of this system are interesting is that they give nearly ideal results that can be compared to other systems.

```
// Periods of the zigzag could be a variable to optimized
c11 = (myZigZag > myZigZag[1])
c12 =  (myZigZag < myZigZag[1])
IF c11 AND (SHORTONMARKET OR NOT LONGONMARKET) THEN
        EXITSHORT COUNTOFSHORTSHARES SHARES AT MARKET
        BUY 50 %CAPITAL AT MARKET
ENDIF
IF c12 AND (LONGONMARKET OR NOT SHORTONMARKET) THEN
        SELL COUNTOFLONGSHARES SHARES AT MARKET
        SELLSHORT 50 %CAPITAL AT MARKET
ENDIF
```

3) Range breakout with trailing stop

This is a Breakout strategy. The signals are generated by breakouts of highest high levels calculated over a certain number of periods.

This system only takes long positions and includes a trailing stop protection. The number of periods should be declared as a variable to be optimized by ProBacktest.

```
REM Period = Optimizable variable (from 2 to 20 by steps of 1)
ONCE MMentry = 5
ONCE Period = 14
REM Enter Long:
Condition = High > Highest[Period](High)[1]
IF Condition AND Summation[Period](Condition) = 1 THEN
        BUY 1 SHARES AT MARKET
ENDIF
c2 = Lowest[10](Low[1])
StopLoss = Highest[MMentry](High)[BarIndex - ENTRYINDEX + 1] / Average[20](High / Low)
SET STOP MAX(StopLoss,(c2 - 0.01))
```

4) Smoothed Stochastic Strategy

This strategy is based on the smoothed stochastic applied to median price and on moving averages.

When the indicator is above its exponential moving average, the system will exit short positions and enter a long position.

The system exits long positions and enters a short position when the indicator is below its moving average.

```
REM Variable Definitions
Indicator1 = SmoothedStochastic[9,9](MedianPrice)
Indicator2 = ExponentialAverage[9](Indicator1)
StopLimit = 10
c1 = (Indicator1 >= Indicator2)
REM Buy
IF c1 THEN
        BUY 1 SHARES AT MARKET RealTime
ENDIF
IF LONGONMARKET THEN
        SELL AT ENTRYQUOTE * (1 + StopLimit / 100) Limit
ENDIF
IF SHORTONMARKET THEN
        EXITSHORT AT ENTRYQUOTE / (1 + StopLimit / 100) Limit
ENDIF
REM Sell
IF NOT c1 THEN
        SELL AT MARKET RealTime
ENDIF
REM Short
IF NOT c1 THEN
        SELLSHORT 1 SHARES AT MARKET RealTime
ENDIF
REM Exit short
IF c1 THEN
        EXITSHORT AT MARKET RealTime
ENDIF
```

5) Swing Trading, ADX and Moving Averages

This backtest uses the ADX indicator and its position with regard to the level 30 since at least 5 to 10 bars, with the goal of reducing false signals and minimizing risk.
The strategy has many conditions that limit the number of trading opportunities.

```
MyADX12 = ADX[12]
ADXperiods = 5
MyMM20 = Average[20](Close)
IsLow30 = 0
FOR Count = 0 TO ADXperiods DO
        IF MyADX12[Count] < 30.0 THEN
                IsLow30 = 1
                BREAK
        ENDIF
NEXT
// LONG
// ADX 12 is greater than 30 since at least 5 to 10 bars
Condition1 = NOT IsLow30
// If the 20-period moving average of the current period is between the high and low of the current
// period and the moving average of the previous period is between the high and low of the previous period
Condition2 = High > MyMM20 AND Low < MyMM20 AND High[1] < MyMM20[1] AND Low[1] < MyMM20[1]
// If the high of the current day is higher than the high of the previous day
Condition3 = Dhigh(0) > Dhigh(1)
IF Condition1 AND Condition2 AND Condition3 THEN
        BUY 1 SHARES AT MARKET ThisBarOnClose
ENDIF
// SHORT
// ADX 12 is greater than 30 since at least 5 to 10 bars
Condition4 = NOT IsLow30
// If the 20-period moving average of the current period is between the high and low of the current
// period and the moving average of the previous period is between the high and low of the previous period
Condition5 = High > MyMM20 AND Low < MyMM20 AND High[1] > MyMM20[1] AND Low[1] > MyMM20[1]
// If the low of the current day is less than the low of the previous day
Condition6 = Dlow(0) < Dlow(1)
IF Condition4 AND Condition5 AND Condition6 THEN
        SELLSHORT 1 SHARES AT MARKET ThisBarOnClose
ENDIF
```

Money management strategies

A backtest's result can be improved by using advanced money management strategies.

These strategies are often formalized in “martingales”. They are aimed at optimizing the mathematical expectancy of a strategy. The expectancy is the average win or loss for each transaction if many transactions are done. This implies being able to estimate the probability of a transaction being winning and the probable amount of profit or loss.

In order to implement a martingale, it can be very useful to have stop loss, take profit and inactivity orders directly coded in a strategy, so that they are fully customizable, and to have sub-strategies allowing us to dynamically manage a position's size.

1) Stop loss

The code below allows you to program a stop directly in your strategy. Don't forget to define the conditions of your stop called StopLossLong and StopLossShort in this code.

```
ONCE Level1 = 0.05
REM Determines the loss threshold above which the position will be closed (0.05 = 5%).
REM If we are long, we close the position as soon as the price moves Level1 % below the entry price.
IF LONGONMARKET AND (Close - ENTRYQUOTE) / (ENTRYQUOTE) < Level THEN
        SELL AT MARKET StopLossLong
ENDIF
REM If we are short, we close the position when the price has increased Level1 % above the entry price.
IF SHORTONMARKET AND (Close - ENTRYQUOTE) / (ENTRYQUOTE) > Level THEN
        EXITSHORT AT MARKET StopLossShort
ENDIF
```

2) Take profit stop

The code below allows you to set a fixed take profit stop. Don't forget to define the conditions of your take profit stop, called TakeProfitLong and TakeProfitShort in this code.

```
ONCE Level2 = 0.05
REM Determines the gains threshold above which the position will be closed (0.05 is equivalent to 5%).
REM If we are long, we close the position as soon as the price moves Level2 % above the entry price.
IF LONGONMARKET AND (Close - ENTRYQUOTE) / (ENTRYQUOTE) > Level THEN
        SELL AT MARKET TakeProfitLong
ENDIF
REM If we are short, we close the position when the price has decreased Level2 % from the entry price.
IF SHORTONMARKET AND (Close - ENTRYQUOTE) / (ENTRYQUOTE) < Level THEN
        EXITSHORT AT MARKET TakeProfitShort
ENDIF
```

3) Inactivity Stop

The following code allows you to use an inactivity stop in your strategy. Don't forget to define the conditions of your stop, here called InactivityStopLong and InactivityStopShort. In the following example, the stop is triggered after 10 bars.

```
ONCE Count = 10
REM Choice of the number of bars after which the position will be systematically closed.
IF ONMARKET AND (BarIndex - ENTRYINDEX + 1) > Count THEN
    IF LONGONMARKET THEN
        SELL AT MARKET InactivityStopLong
    ENDIF

    IF SHORTONMARKET THEN
        EXITSHORT AT MARKET InactivityStopShort
    ENDIF
ENDIF
```

4) Pyramiding a position

To pyramid positions, first check “Cumulate positions” in the backtest's “Capital management” window. Pyramiding means placing several successive orders in the same directions to increase the position size. An example of coding pyramiding in a backtest is shown in this example:

```
REM BUY when RSI > 30
IF RSI[14](Close) < 30 THEN
    BUY 1 SHARES AT MARKET
ENDIF
REM If there is a long position open and Open is greater than Previous close, buy 1 additional stock/lot.
IF LONGONMARKET AND Open > Close[1] THEN
    BUY 1 SHARES AT MARKET
ENDIF
REM If price crosses under simple moving average, sell the entire position.
IF Close Crosses Under Average[14](Close) THEN
    SELL 100 %CAPITAL AT MARKET
ENDIF
```

5) Dynamic position size management

To dynamically vary a position size without pyramiding, you can use a variable indicating the amount of stocks/lots to purchase when an order is placed as shown in this example.

```
ONCE OrderSize = 1
REM The variable order size is initially set at 1.  We buy OrderSize shares.
BUY OrderSize SHARES AT MARKET
REM The position is closed after 2 bars systematically.
IF BarIndex - ENTRYINDEX >= 2 THEN
        SELL AT MARKET
ENDIF
REM IF RSI is less than 30, we increase OrderSize by 1 each bar.
REM OrderSize may not be greater than 50.
IF RSI[14](Close) < 30 THEN
        OrderSize = MAX(OrderSize + 1, 50)
ENDIF
REM If RSI is more than 70, we decrease OrderSize by 1 each bar.
REM OrderSize may not be less than 0.
IF RSI[14](Close) > 70 THEN
        OrderSize = MIN(OrderSize - 1, 0)
ENDIF
```

6) Taking into account previous performance

By using the PreviousTrade(n), we can modify a strategy based on its performance. By taking the previous backtest, we can make it more intelligent by increasing position size when previous trades were winning or decreasing position size when previous trades were losing.

```
ONCE OrderSize = 1
REM The variable order size is initially set at 1.  We buy OrderSize shares.
BUY OrderSize SHARES AT MARKET
REM The position is closed after 2 bars systematically.
IF BarIndex - ENTRYINDEX >= 2 THEN
        SELL AT MARKET
ENDIF
REM IF RSI is less than 30, we increase OrderSize by 1 each bar.
REM OrderSize may not be greater than 50.
IF RSI[14](Close) < 30 THEN
        OrderSize = MAX(OrderSize + 1, 50)
ENDIF
REM If RSI is more than 70, we decrease OrderSize by 1 each bar.
REM OrderSize may not be less than 0.
IF RSI[14](Close) > 70 THEN
        OrderSize = MIN(OrderSize - 1, 0)
ENDIF
REM Application of behavior modification depending on past performance.
REM We analyze successively the 3 previous trades.
REM If a trade was losing, the position size decreases by 1.  Otherwise, it increases by 1.
FOR n = 1 TO 3 DO
        IF PreviousTrade(n) >= 0 THEN
                OrderSize = MAX(OrderSize + 1, 50)
        ELSIF PreviousTrade(n) < 0 THEN
                OrderSize = MIN(OrderSize - 1, 0)
        ENDIF
NEXT
```

With these tools, we can now use different martingales in ProBacktest strategies. Here are some examples of capital management techniques that can be used with other strategies.

7) The classic martingale

The classic martingale doubles the position size when it loses in order to make up for the loss if the next trade is a winner. The disadvantage of a strategy like this is that successive losses make it more and more difficult (or impossible) to double your position. Starting with 1000€ for example, if you lose 5 times in a row, you would need 1000 x 2^5 or 32000€ to continue with this strategy.

As a result, strategies with the martingale may be more adapted to trading stocks than Futures or Forex because the initial capital required to trade may be much larger in these 2 types of markets.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
//********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        OrderSize = OrderSize * 2
        REM If the last trade was a losing trade, we double the position size.
ELSIF PreviousTrade(1) > 0 THEN
        OrderSize = 1
        REM If the last trade was a winning trade, we go back to a position size of 1.
ENDIF
//********************//
REM In the backtest, the position size must be set using the variable OrderSize.
```

8) The great martingale

The great martingale is similar to the classic martingale, except that in addition to doubling the position size after each loss, we add one additional unit.

This is more risky than the classic martingale in case of successive losses but it allows significantly increasing gains otherwise.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
//********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        OrderSize = OrderSize * 2 + 1
        // If the last trade was a losing trade, we double position size and add one.
ELSIF PreviousTrade(1) >= 0 THEN
        OrderSize = 1
        // If the last trade was a winning trade, we go back to a position size of 1.
ENDIF
//********************//
REM In the backtest, the position size must be set using the variable OrderSize.
```

9) The Piquemouche

The Piquemouche is another variant of the classic martingale. In case of loss, we increase the position size by 1 if there are less than 3 consecutive losses. If there are more than 3 consecutive losses, we double the position. A gain resets the position size to 1 unit.

This strategy is less risky than the 2 previous ones because the position size is not exponentially increased until 3 successive losses are attained.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
ONCE BadTrades = 0
// We initiate the counter of losing trades.
//*********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        BadTrades = BadTrades + 1
                IF BadTrades < 3 THEN
                        // If the last trade was losing and there are less than 3 consecutive losses, we
                        // increase the size of the position by 1.
                        OrderSize = OrderSize + 1
                ELSIF PreviousTrade(1) < 0 AND BadTrades MOD 3 = 0 THEN
                        // If the last trade was losing and there were more than 3 consecutive losses,
                        // we double the size of the next position.
                        OrderSize = OrderSize * 2
                ENDIF
ELSIF PreviousTrade(1) >= 0 THEN
                // If the last trade was a winning trade, we go back to a position size of 1.
                OrderSize = 1
                BadTrades = 0
ENDIF
//*********************//
REM In the backtest, the position size must be set using the variable OrderSize.
```

10) The Whittacker

In a Whittacker, when there is a loss, we set the position size to the sum of the two previous position sizes. In case of a gain, the position size is set to 1 unit.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
//********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        OrderSize = OrderSize + OrderSize[1]
ELSIF PreviousTrade(1) >= 0 THEN
        OrderSize = 1
        // If the last trade was a winning trade, we go back to a position size of 1.
ENDIF
//********************//
REM In the backtest, the position size must be set using the variable OrderSize.
```

11) The D'Alembert Pyramid

This martingale was made famous by d'Alembert, a French 18th century mathematician. In case of loss, the position size is increased by 1 unit, in case of gain it is decreased by 1 unit.

This technique of position size management is relevant only if we suppose that successive gains reduce the probability of winning again and successive losses reduce the probability of losing again.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
//********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        OrderSize = OrderSize + 1
ELSIF PreviousTrade(1) >= 0 THEN
        OrderSize = MAX(OrderSize -1, 1)
ENDIF
//********************//
REM In the backtest, the position size must be set using the variable OrderSize.
```

12) The contre d'Alembert

This is the reciprocal strategy of the D'Alembert Pyramid. We decrease the position size in case of a loss and increase the position size in case of a gain.

This technique is relevant if we believe that a past loss increases the probability of a future loss and a past gain increases the probability of a future gain.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the Strategy*********//
ONCE OrderSize = 1
REM We start with a position size of 1.
//********************//
//***********Code to insert just after closing a position**********//
IF PreviousTrade(1) < 0 THEN
        OrderSize = MAX(OrderSize - 1, 1)
ELSIF PreviousTrade(1) >= 0 THEN
        OrderSize = OrderSize + 1
ENDIF
//********************//
```

ANNEX: CAPITAL MANAGEMENT SETUP

The "capital management" window is available in the Money Management section of the ProBacktest programming window.

Capital management is a key element that can make strategy results vary widely. For example, reducing brokerage fees or different risk management rules can significantly increase net backtest results.

This window is made of 5 customizable sections :

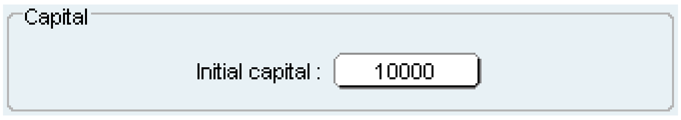
- Capital
- Brokerage Fees
- Risk Management
- Orders Management (Position management)
- Round the number of securities to buy/sell

Different rules apply for the brokerage fees depending on what type of security you are working with such as stocks, futures or Forex. We will explain how to setup brokerage fee settings for each of these types of securities at the end of this annex.

Capital

In this section, you just enter the amount you want for the starting capital of the trading strategy.

Note : except in special cases (see the section on risk management), ProBacktest will not take any more positions if the initial capital is all lost.

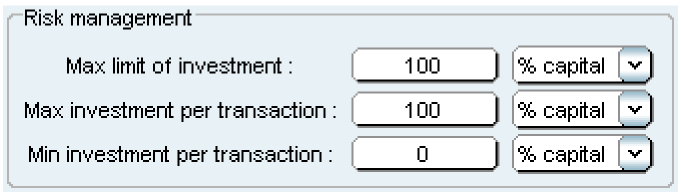

Capital

Initial capital : 10000

If your strategy is not generating orders, increase your initial capital to make sure it is sufficient to place an order.

Risk Management

Risk management allows you to setup these 3 parameters:


Risk management

Max limit of investment :	100	% capital
Max investment per transaction :	100	% capital
Min investment per transaction :	0	% capital

1) Maximum limit of investment

This field is very useful to limit your losses or manage leverage. Begin by selecting how you want to set your max limit of investment. You can choose % of capital, % of liquidity or absolute amount.

2) Max investment per transaction and leverage

The max investment per transaction allows you to limit the amount allowed for each order and works in the same way as the previous field.

By combining this feature with the max limit of investment, you can manage leverage. Let's look at an example:

- Max limit of investment: 500 %Capital
- Max limit per transaction: 500 % Capital

With this configuration, you will be able to use a leverage of 5 in your backtest.

If you limit yourself to 100% of capital, the maximum loss possible would be the total amount of your backtest portfolio.

3) Minimum investment per transaction

The use of minimum investment per transaction is to avoid an investment which is too small implying brokerage fees which will be proportionately too large compared to the possible gain from trading.

(Example: The purchase of one stock X at a price of 5 with a brokerage fee of 5€/order meaning that the brokerage fee represents 100% of the invested capital and that your loss would immediately be -100% or -200% if you then sell the stock at the same price at which you bought it).

Orders management

This section allows you to check a maximum of 3 options.

"Reinvest profit" allows you to decide how you will manage your profits: reinvest them or not. By default, the system does not increase Initial Capital with the gains obtained. If you check this box, the gains will increase the capital available for the backtest.

"Cumulate positions" allows you to define whether to cumulate unclosed positions in case the condition to enter a position is true for several successive bars. By default, only one one position is taken at a time except in the case of explicit definition in your code. In this case, checking this option or not will have no effect on your backtest.

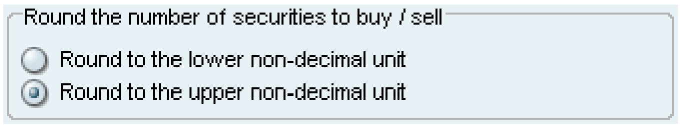
If in the interface, you have setup predefined STOPs, you may associate them to your positions with the following parameters:

- One stop for all positions (if cumulate positions is checked)
- One stop for each position

This section becomes particularly interesting if you use trailing stops. A trailing stop is a stop that follows the price and is usually defined as a distance in % from the last price. In the case of cumulated positions, the the system will have a choice of using only one trailing stop (based on the average entry price) or use one trailing stop per position.

Round the number of securities to buy/sell

This section is relatively simple to fill in. You decide how the number of securities to buy or sell will be rounded. This is applicable for example if you decide to buy a certain % of your available capital worth of shares. Just choose one of the following options:


Round the number of securities to buy / sell

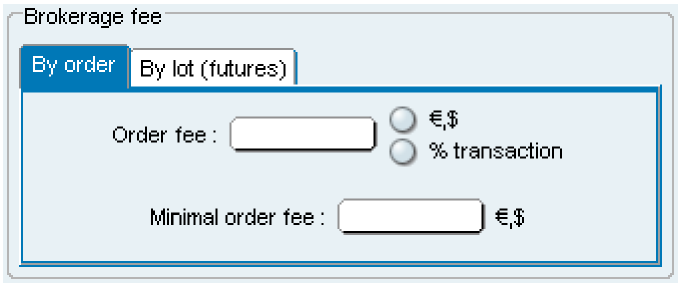
○ Round to the lower non-decimal unit
[✓ Round to the upper non-decimal unit]

1) Brokerage fees for stocks

To apply a ProBacktest to stocks or similar securities (ex: warrants, …) you need to choose the tab "by order", in the brokerage fees section. As you can see in the figure below, its possible to setup the brokerage fees as an absolute amount of cash or in %.

The brokerage fees are applied to one order. Opening and then closing a position (ex: buy and then selling a stock) will charge you 2 times the brokerage fee (one charge for each order) in the ProBacktest simulation.

The "minimum order fee" allows you to define a minimum brokerage fee per order for brokerage fees calculated in % of the transaction.


Brokerage fee

By order | By lot (futures)

Order fee : ○ €,$ ○ % transaction

Minimal order fee : €,$

Ex : If the "minimum order fee" is 15€ and the % of your transaction is only 10€, then the total brokerage fee for the order taken into account by ProBacktest will be 15€ for that order.

2) Brokerage fees for futures

Begin by choosing the tab "by lot (futures)" in the section "Brokerage fees".

You must then define :

- The commission by lot
- The deposit by lot
- The value of one point

a. Commission by lot

In the futures market, the commission by lot is considered the equivalent of the brokerage fee for stocks. Each broker will have its own commission rate which you can fill in this section to simulate trading with a commission at that rate.

b. Deposit by lot

The deposit by lot is the deposit you put down with your broker. It allows you to manage your leverage which can be large with certain futures.

Example :

You have $500 and you are authorized to trade with $1000 by your broker. The initial $500 is the deposit you put down with the broker. With $1000 to trade, you have a leverage of 2. If we want to have a simulation as close to reality as possible, we must consider that the broker will not wait for you to have lost $1000 to liquidate your position. We can assume for this example that the broker would close your positions as soon as your deposit is no longer enough to cover losses (so at 500$ of loss).

The deposit by lot varies depending on the future you are working with and your broker. To find out the exact amount, contact your broker.

If you want to simulate a certain leverage amount, here is the formula that will allow you to calculate your deposit by lot with regards to this leverage:

Deposit par lot = Initial Capital / Leverage

Suppose you have an initial capital of $1000 and you want to simulate a leverage of 5, your deposit will be 1000/5= $200.

c. The value of a point (one point worth)

This value indicates the leverage applied to realized gains or losses. The value to input depends on the future you are applying the backtest on and is expressed in the monetary units of the future. The calculation formula is as follows:

1 point worth = Monetary value of one tick / Size of one tick

The tick in this case is the smallest unit of price variation authorized in the market (also called "ticksize").

Here are the values for the main futures contracts:

Future Contract	Value of 1 tick	Size of 1 tick	Value of 1 point
FCE CAC 40	5€	0,5	10€
DAX	12,5€	0,5	25€
DJ Eurostoxx 50	10€	1	10€
BUND	10€	0,01	1000€
Euro FX	12,5$	0,0001	125000$
Mini S&P 500	12,5$	0,25	50$
Mini Nasdaq 100	5$	0,25	20$
Mini Dow	5$	1	5$

3) Brokerage fees for Forex

Begin by clicking on the tab "by lot (futures)" in the section "Brokerage fees".

Fill in the sections following these directions:

a. Commission by lot / Spread

The commission by lot for one order in the the FOREX market is equal to the spread times the pip value divided by 2 (the spread * the pip value is the cost of entering and exiting a position). A pip in the Forex market is the minimum unit of price variation of a currency pair. The value of a pip depends on the lot size

It can be strange to apply the word commission in the Forex market, normally the term SPREAD is used instead. The spread is equal to the difference between the selling (bid) and buying (ask) price:

SPREAD = (Bid – Ask)

Commission by Lot=Spread*Pip Value / 2

Each Forex broker has different spreads. In addition, the spreads of a given broker can vary depending on time or market conditions. For this reason, its **usually not possible for you to calculate what the spread of your broker will be** all the time when you trade currencies and the results of backtests on these types of securities may not be interpreted as completely realistic for this reason.

For these reasons, we advise you to enter 0 in this field.

If you want to simulate the spread of your broker as a given value, then you could enter (the spread*the pip value)/2 as the commission by lot.

To simulate a spread equal to 3 pips (on EURUSD), you could enter in Commission by lot: 3 * $10 / 2 = $15.

b. Deposit by lot

The deposit by lot is the deposit you put down with your broker. It allows you to manage your leverage which can be large with certain currency pairs such as JPY pairs.

Example :

You have $500 and you are authorized to trade with $1000 by your broker. The initial $500 is the deposit you put down with the broker. With $1000 to trade, you have a leverage of 2. If we want to have a simulation as close to reality as possible, we must consider that the broker will not wait for you to have lost $1000 to liquidate your position. We can assume for this example that the broker would close your positions as soon as your deposit is no longer enough to cover losses (so at $500 of loss).

The deposit by lot varies depending on the future you are working with and your broker. To find out the exact amount, contact your broker.

If you want to simulate a certain leverage amount, here is the formula that will allow you to calculate your deposit by lot with regards to this leverage:

Deposit par lot = Initial Capital / Leverage

Suppose you want to trade on EURUSD (lots of $100000) with a leverage of 100, your deposit by lot would be: $100000/100 = $1000.

c. The value of a point (one point worth)

This value indicates the leverage applied to realized gains or losses. The value to input depends on the currency pair you are applying the backtest on and is expressed in the monetary units of the pip value (second currency in the pair). The calculation formula is as follows:

1 point = Pip value / Pip size

Example :

Consider EURUSD which has a pip size of 0.0001. We know that 1 pip has a value of $10 for this pair.

For EURUSD and a standard lot size of $100 000:

1 point = 10/0.0001 = 100 000

GLOSSARY

A

CODE	SYNTAX	FUNCTION
Abs	Abs(a)	Mathematical function "Absolute Value" of a
AccumDistr	AccumDistr(close)	Classical Accumulation/Distribution indicator
ADX	ADX[N]	Indicator Average Directional Index or "ADX" of n periods
ADXR	ADXR[N]	Indicator Average Directional Index Rate or "ADXR" of n periods
AND	a AND b	Logical AND Operator
AroonDown	AroonDown[N]	Aroon Down indicator of n periods
AroonUp	AroonUp[N]	Aroon Up indicator of n periods
Atan	Atan(a)	mathematical function "Arctangent" of a
AS	RETURN Result AS "ResultName"	Instruction used to name a line or indicator displayed on chart. Used with "RETURN"
AT	AT (price)	Associates a command to a price
Average	Average[N](price)	Simple Moving Average of n periods
AverageTrueRange	AverageTrueRange[N](price)	"Average True Range" - True Range smoothed with the Wilder method

B

CODE	SYNTAX	FUNCTION
BarIndex	BarIndex	counts how many bars are displayed on all the data loaded
BollingerBandWidth	BollingerBandWidth[N](price)	Bollinger Bandwidth indicator
BollingerDown	BollingerDown[N](price)	Lower Bollinger band
BollingerUp	BollingerUp[N](price)	Upper Bollinger band
BREAK	(FOR...DO...BREAK...NEXT) or (WHILE...DO...BREAK...WEND)	Instruction forcing the exit of FOR loop or WHILE loop
BUY	BUY x SHARES	Instruction to open a long position

C

CODE	SYNTAX	FUNCTION
CALL	myResult = CALL myFunction	Calls a user indicator to be used in the program you are coding
CAPITAL	BUY x% CAPITAL	% of capital used in the position
CASH	BUY x CASH	Amount of cash used in the position
CCI	CCI[N](price) or CCI[N]	Commodity Channel Index indicator
ChaikinOsc	ChaikinOsc[Ch1, Ch2](price)	Chaikin oscillator
Chandle	Chandle[N](price)	Chande Momentum Oscillator
ChandeKrollStopUp	ChandeKrollStopUp[Pp, Qq, X]	Chande and Kroll Protection Stop on long positions
ChandeKrollStopDown	ChandeKrollStopDown[Pp, Qq, X]	Chande and Kroll Protection Stop on short positions
Close	Close[N]	Closing price of the current bar or of the n-th last bar
COLOURED	RETURN Result COLOURED(R,G,B)	Colors a curve with the color you defined using the RGB convention
COS	COS(a)	Cosine Function
COUNTOFLONGSHARES	COUNTOFLONGSHARES	Counts the number of open long shares or lots
COUNTOFPOSITION	COUNTOFPOSITION	Counts the number of open shares or lots
COUNTOFSHORTSHARES	COUNTOFSHORTSHARES	Counts the number of open short shares or lots
Crosses Over	a Crosses Over b	Boolean Operator checking whether a curve has crossed over another one
Crosses Under	a Crosses Under b	Boolean Operator checking whether a curve has crossed under another one
CUMSUM	CUMSUM(price)	Sums a certain price on the whole data loaded
CurrentDayOfWeek	CurrentDayOfWeek	Represents the current day of the week
CurrentHour	CurrentHour	Represents the current hour
CurrentMinute	CurrentMinute	Represents the current minute
CurrentMonth	CurrentMonth	Represents the current month
CurrentSecond	CurrentSecond	Represents the current second
CurrentTime	CurrentTime	Represents the current time (HHMMSS)
CurrentYear	CurrentYear	Represents the current year
CustomClose	CustomClose[N]	Term customizable in the settings window of the chart (default: Close)
Cycle	Cycle(price)	Cycle Indicator

D

CODE	SYNTAX	FUNCTION
Date	Date[N]	Reports the date of each bar loaded on the chart
Day	Day[N]	Reports the day of each bar loaded in the chart
Days	Days[N]	Counter of days since 1900
DayOfWeek	DayOfWeek[N]	Day of the week of each bar
Dclose	Dclose(N)	Close of the n-th day before the current one
DEMA	DEMA[N](price)	Double Exponential Moving Average
Dhigh	Dhigh(N)	High of the n-th bar before the current bar
DI	DI[N](price)	Represents DI+ minus DI-
DIminus	Diminus[N](price)	Represents the DI- indicator
Diplus	Diplus[N](price)	Represents the DI+ indicator
Dlow	Dlow(N)	Low of the n-th day before the current one
DO	See FOR and WHILE	Optional instruction in FOR loop and WHILE loop to define the loop action
Dopen	Dopen(N)	Open of the n-th day before the current one
DOWNTO	See FOR	Instruction used in FOR loop to process the loop with a descending order
DPO	DPO[N](price)	Detrented Price Oscillator

E

CODE	SYNTAX	FUNCTION
EaseOfMovement	EaseOfMovement[I]	Ease of Movement indicator
ELSE	See IF/THEN/ELSE/ENDIF	Instruction used to call the second condition of If-conditional statements
ELSEIF	See IF/THEN/ELSIF/ELSE/ENDIF	Stands for Else If (to be used inside of conditional loop)
EMV	EMV[N]	Ease of Movement Value indicator
ENDIF	See IF/THEN/ELSE/ENDIF	Ending Instruction of IF-conditional statement
EndPointAverage	EndPointAverage[N](price)	End Point Moving Average of a
ENTRYINDEX	ENTRYINDEX(x)	Indicates the index of the bar on which a previous order was executed
ENTRYQUOTE	ENTRYQUOTE(x)	Indicates the price of a previous order executed

EXITSHORT	EXITSHORT x SHARES	Instruction to close a short position
Exp	Exp(a)	Mathematical Function "Exponential"
ExponentialAverage	ExponentialAverage[N] (price)	Exponential Moving Average

F-G

CODE	**SYNTAX**	**FUNCTION**
FOR/TO/NEXT	FOR i=a TO b DO a NEXT	FOR loop (processes all the values with an ascending (TO) or a descending order (DOWNTO))
ForceIndex	ForceIndex(price)	Force Index indicator (determines who controls the market (buyer or seller)

H

CODE	**SYNTAX**	**FUNCTION**
High	High[N]	High of the current bar or of the n-th last bar
Highest	Highest[N](price)	Highest price over a number of bars to be defined
HistoricVolatility	HistoricVolatility[N](price)	Historic Volatility (or statistic volatility)
Hour	Hour[N]	Represents the hour of each bar loaded in the chart

I-J-K

CODE	**SYNTAX**	**FUNCTION**
IF/THEN/ENDIF	IF a THEN b ENDIF	Group of conditional instructions without second instruction
IF/THEN/ELSE/ENDIF	IF a THEN b ELSE c ENDIF	Group of conditional instructions
IntradayBarIndex	IntradayBarIndex[N]	Counts how many bars are displayed in one day on the whole data loaded

L

CODE	SYNTAX	FUNCTION
LIMIT	BUY AT x LIMIT	Instruction introducing a limit order
LinearRegression	LinearRegression[N](price)	Linear Regression inidcator
LinearRegressionSlope	LinearRegressionSlope[N](price)	Slope of the Linear Regression inidcator
LIQUIDITY	BUY x %LIQUIDITY	Designates % of available liquidity to use when opening a position
Log	Log(a)	Mathematical Function "Neperian logarithm" of a
LONGONMARKET	LONGONMARKET	Indicates whether you have open long positions or not
Low	Low[N]	Low of the current bar or of the n-th last bar
Lowest	Lowest[N](price)	Lowest price over a number of bars to be defined

M

CODE	SYNTAX	FUNCTION
MACD	MACD[S,L,Si](price)	Moving Average Convergence Divergence (MACD) in histogram
MACDline	MACDLine[S,L](price)	MACD line indicator
MARKET	BUY AT MARKET	Designates an order at market price
MassIndex	MassIndex[N]	Mass Index Indicator applied over N bars
Max	Max(a,b)	Mathematical Function "Maximum"
MedianPrice	MedianPrice	Average of the high and the low
Min	Min(a,b)	Mathematical Function "Minimum"
Minute	Minute	Represents the minute of each bar loaded in the chart
Mod	a Mod b	Mathematical Function "remainder of the division"
Momentum	Momentum[N]	Momentum indicator (close – close of the n-th last bar)
MoneyFlow	MoneyFlow[N](price)	MoneyFlow indicator (result between -1 and 1)
MoneyFlowIndex	MoneyFlowIndex[N]	MoneyFlow Index indicator
Month	Month[N]	Represents the month of each bar loaded in the chart

N

CODE	SYNTAX	FUNCTION
NEXT	See FOR/TO/NEXT	Ending Instruction of FOR loop
NextBarClose	AT MARKET NextBarClose	Designates an order to be executed on the next bar's close
NextBarOpen	AT MARKET NextBarOpen	Designates an order to be executed on the open of the next bar
NOT	NOT A	Logical Operator NOT

O

CODE	SYNTAX	FUNCTION
OBV	OBV(price)	On-Balance-Volume indicator
ONCE	ONCE VariableName = VariableValue	Introduces a definition statement which will be processed only once
Open	Open[N]	Open of the current bar or of the n-th last bar
OpenOfNextBar	OpenOfNextBar[N]	Open of the bar following the n-th last bar
OR	a OR b	Logical Operator OR

P-Q

CODE	SYNTAX	FUNCTION
Previous Trades Performance	PreviousTrade(x)	Indicates the percent of gain or loss of a previous trade
PriceOscillator	PriceOscillator[S,L](price)	Percentage Price oscillator
PositiveVolumeIndex	PriceVolumeIndex(price)	Positive Volume Index indicator
PVT	PVT(price)	Price Volume Trend indicator

R

CODE	SYNTAX	FUNCTION
R2	R2[N](price)	R-Squared indicator (error rate of the linear regression on price)
Range	Range[N]	calculates the Range (High minus Low)
RealTime	AT MARKET RealTime	Designates a order to be executed in real-time
REM	REM comment	Introduces a remark (non prise en compte dans le code mais facilitant une relecture)
Repulse	Repulse[N](price)	Repulse indicator (measure the buyers and sellers force for each candlestick)
RETURN	RETURN Result	Instruction returning the result
ROC	ROC[N](price)	Price Rate of Change indicator
RSI	RSI[N](price)	Relative Strength Index indicator
Round	Round(a)	Mathematical Function "Round a to the nearest whole number"

S

CODE	SYNTAX	FUNCTION
SAR	SAR[At,St,Lim]	Parabolic SAR indicator
SARatdmf	SARatdmf[At,St,Lim](price)	Smoothed Parabolic SAR indicator
SELL	SELL x SHARES	Instruction to close a long position
SELLSHORT	SELLSHORT x SHARES	Instruction to open a short position
SET	SET	Determines the type of order: either limit or stop
SET STOP	SET STOP price	Instruction to define a protection stop
SHARES	BUY x SHARES	Designates the number of shares to buy or sell
SHORTONMARKET	SHORTONMARKET	Indicates whether there are open short positions or not
Sin	Sin(a)	Mathematical Function "Sine"
Sgn	Sgn(a)	Mathematical Function "Sign of" a (it is positive or negative)
SMI	SMI[N,SS,DS](price)	Stochastic Momentum Index indicator
SmoothedStochastic	SmoothedStochastic[N,K] (price)	Smoothed Stochastic
Square	Square(a)	Mathematical Function "a Squared"
Sqrt	Sqrt(a)	Mathematical Function "Squared Root" of a

STD	STD[N](price)	Statistical Function "Standard Deviation"
STE	STE[N](price)	Statistical Function "Standard Error"
Stochastic	Stochastic[N,K](price)	%K Line of the Stochastic indicator
STOP	AT price STOP	Instruction to create a stop order
Summation	Summation[N](price)	Sums a certain price over the N last candlesticks
SuperTrend	SuperTrend[STF,N]	Super Trend indicator

T

CODE	**SYNTAX**	**FUNCTION**
Tan	Tan(a)	Mathematical Function "Tangent" of a
TEMA	TEMA[N](price)	Triple Exponential Moving Average
THEN	See IF/THEN/ELSE/ENDIF	Instruction following the first condition of "IF"
ThisBarOnClose	AT MARKET ThisBarOnClose	Designates an order to be executed on the close of the current bar
Time	Time[N]	Represents the time of each bar loaded in the chart
TimeSeriesAverage	TimeSeriesAverage[N](price)	Temporal series moving average
TO	See FOR/TO/NEXT	Directional Instruction in the "FOR" loop
Today	Today[N]	Date of the bar n-periods before the current bar
TodayOnClose	AT MARKET TodayOnClose	Designates an order to be executed on the close of the current day
TomorrowClose	AT MARKET TomorrowClose	Designates an order to be executed on the close of the next day
TomorrowOpen	AT MARKET TomorrowOpen	Designates an order to be executed on the open of the next day
TotalPrice	TotalPrice[N]	(Close + Open + High + Low)/4
TR	TR(price)	True Range indicator
TriangularAverage	TriangularAverage[N](price)	Triangular Moving Average
TRIX	TRIX[N](price)	Triple Smoothed Exponential Moving Average
TypicalPrice	TypicalPrice[N]	Represents the Typical Price (Average of the High, Low and Close)

U

CODE	SYNTAX	FUNCTION
Undefined	a = Undefined	Sets a the value of a variable to undefined

V

CODE	SYNTAX	FUNCTION
Variation	Variation(price)	Difference between the close of the last bar and the close of the current bar in %
Volatility	Volatility[S, L]	Chaikin volatility
Volume	Volume[N]	Volume indicator
VolumeOscillator	VolumeOscillator[S,L]	Volume Oscillator
VolumeROC	VolumeROC[N]	Volume of the Price Rate Of Change

W

CODE	SYNTAX	FUNCTION
WeightedAverage	WeightedAverage[N](price)	Represents the Weighted Moving Average
WeightedClose	WeightedClose[N]	Average of (2*Close), (1*High) and (1*Low)
WEND	See WHILE/DO/WEND	Ending Instruction of WHILE loop
WHILE/DO/WEND	WHILE (condition) DO (action) WEND	WHILE loop
WilderAverage	WilderAverage[N](price)	Represents Wilder Moving Average
Williams	Williams[N](close)	%R de Williams indicator
WilliamsAccumDistr	WilliamsAccumDistr(price)	Accumulation/Distribution of Williams Indicator

X

CODE	SYNTAX	FUNCTION
XOR	a XOR b	Logical Operator eXclusive OR

Y

CODE	SYNTAX	FUNCTION
Year	Year[N]	Year of the bar n periods before the current bar
Yesterday	Yesterday[N]	Date of the day preceeding the bar n periods before the current bar

Z

CODE	SYNTAX	FUNCTION
ZigZag	ZigZag[Zr](price)	Represents the Zig-Zag indicator introduced in the Eliott waves theory
ZigZagPoint	ZigZagPoint[Zp](price)	Represents the Zig-Zag indicator in the Eliott waves theory calculated on Zp points

Other

CODE	FUNCTION
+	Addition Operator
-	Substraction Operator
*****	Multiplication Operator
/	Division Operator
=	Equality Operator
<>	Difference Operator
<	Strict Inferiority Operator
>	Strict Superiority Operator
<=	Inferiority Operator
>=	Superiority Operator
//	Introduces a commentary line

IT-FINANCE

Information Technology for Finance