PORTFOLIO TRADER

STRATEGY EXAMPLES

CONTENTS

ROTATION STRATEGY

This strategy was suggested by **kbeary33** on MultiCharts Forum (Link)**.**

STRATEGY DESCRIPTION

'Rotation Strategy' is a simple strategy that calculates a specific indicator by using every instrument in the portfolio. Positions are opened for those instruments which have the best indicator value(s).

Take for example the "% Change" indicator. This set of instruments is determined by the user in the Portfolio Trading application. The number of instruments to enter a Long position is configured by the "BuyBestX" input. Standard stop loss +profit target strategy is used to exit positions.

STRATEGY DEVELOPMENT

a) Portfolio_Rotation signal

This signal generates entry orders and calculates indicator values for all instruments in the portfolio.

Indicator formula is entered into the input field and it is calculated on every bar.

```
inputs:
       Formula(PercentChange(close, 14));

variables: formulaValue(0);

       formulaValue = Formula;
```

To further compare and decide to enter the position, the formula is then entered into global variables:

```
pmm_set_my_named_num("RotationalValue", formulaValue);
```

Entries for all instruments are generated:

```
buy("LE") next bar market;
```

b) Portfolio_Rotation

This strategy checks the indicator values for all instruments and manages opening positions.

The user sets the number of portfolio instruments for which positions are opened:

```
inputs: BuyBestX(10);
```

Rotation Strategy extracts the Indicator values of all instruments, then creates a sorted list of values at every calculation. To do this, we need 2 arrays: one for the indicator values and another for strategy indices.

```
variables: idx(0), strategyIdx(0), strategyValue(0);
       arrays: bestStrategies[](-1), bestValues[](0);

once begin
       emulate_dictionary__set_size(bestStrategies, bestValues,
BuyBestX);
```

```
        emulate_dictionary__set_size(worstStrategies, worstValues,
SellWorstY);
      end;
```

Entry order generation is disabled before every calculation:

```
pmms_strategies_deny_entries_all;
```

Arrays are cleared and then filled with indicator values and strategy indices.

```
emulate_dictionary__fill_defaults(bestStrategies, bestValues, -1, 0);
emulate_dictionary__fill_defaults(worstStrategies, worstValues, -1, 0);


for strategyIdx = 0 to pmms_strategies_count - 1 begin
       strategyValue = pmms_get_strategy_named_num(strategyIdx,
"RotationalValue");
       //print(currentbar:0:0, " For " , strategyIdx:0:0, " strategy
value = ", strategyValue:0:5);
       emulate_dictionary__insert_as_best(bestStrategies, bestValues,
strategyIdx, strategyValue, -1);
end;
```

Finally, we calculate how many strategies have an open position. The "BuyBestX" number of instruments should have open positions based on the indicator's best values:

```
variables: inLong(0);
arrays: strategiesLong[](-1);

inLong = pmms_strategies_in_long_count(strategiesLong);

for idx = 0 to BuyBestX - 1 begin
       if (not array_contains(strategiesLong, bestStrategies[idx]))
then begin
              pmms_strategy_allow_long_entries(bestStrategies[idx]);
       end;
end;
```

c) Function "emulate_dictionary__fill_defaults":

```
inputs:
       KeyArray[MaxSize1](NumericArrayRef),
       ValueArray[MaxSize2](NumericArrayRef),

       defaultKey(NumericSimple),
       defaultValue(NumericSimple);

variables: idx(0);

for idx = 0 to MaxSize1 begin
       KeyArray[idx] = defaultKey;
end;

for idx = 0 to MaxSize2 begin
```

```
        ValueArray[idx] = defaultValue;
end;
```

d) Function "emulate_dictionary__insert_as_best"

```
inputs:
        KeyArray[MaxSize1](NumericArrayRef),
        ValueArray[MaxSize2](NumericArrayRef),

        Key(NumericSimple),
        Value(NumericSimple),

        defaultKey(NumericSimple);

variables: idx(0);


for idx = 0 to MaxSize1 begin
        if KeyArray[idx] = defaultKey then begin
                KeyArray[idx] = Key;
                ValueArray[idx] = Value;
                break;
        end;

        if (Value > ValueArray[idx]) then begin
                emulate_dictionary__move_right(KeyArray, ValueArray,
idx);
                KeyArray[idx] = Key;
                ValueArray[idx] = Value;
                break;
        end;
end;
```

e) Function "emulate_dictionary__move_right":

```
inputs:
        KeyArray[MaxSize1](NumericArrayRef),
        ValueArray[MaxSize2](NumericArrayRef),

        FromIndex(NumericSimple);

variables: idx(0);


for idx = MaxSize1 downto FromIndex + 1 begin
        KeyArray[idx] = KeyArray[idx - 1];
        ValueArray[idx] = ValueArray[idx - 1];
end;
```

SPREAD TRADING STRATEGY

STRATEGY DESCRIPTION

Spread trading is a type of trading where instruments, divided into pairs, trade in opposite directions. This type of trading occurs when a Long Position is opened for one instrument, while another is opened simultaneously in the opposite direction (Short). Both of these positions open and close synchronously.

Here is an example. A portfolio has two pairs of instruments: QQQ vs SPY and KO vs PEP.

The strategy will enter into position when the spread deviation exceeds a Standard Deviation value for the last 20 bars. The Second Pair of Instruments enters synchronously into a position opposite the Main Instruments (First Pair).

STRATEGY DEVELOPMENT

a) Portfolio_SpreadTradingSystem.Master Signal

This signal is calculated on based on an instrument's data series. It contains opening and closing logic positions:

```
inputs: Ratio(c / c data2), Length(10), PercentOfEquity(10);

var: AvgRatio(0), StdDevRatio(0);
var: intrabarpersist cur_pos(0);

var: Contracts_(0);
Contracts_ = Portfolio_Equity * PercentOfEquity / 100;

if 1 < currentbar then begin
       if AvgRatio + StdDevRatio < Ratio then begin// short data1, long
data2
              if -1 <> cur_pos then begin
                     sellshort Contracts_ contracts this bar at c;
                     cur_pos = -1;
              end;
       end else if AvgRatio - StdDevRatio > Ratio then begin// buy
data1, short data2
              if 1 <> cur_pos then begin
                     buy Contracts_ contracts this bar at c;
                     cur_pos = 1;
              end;
       end else begin
              cur_pos = 0;
              sell this bar c;
              buytocover this bar c;
       end;
end;

AvgRatio = XAverage(Ratio, Length);
StdDevRatio = StdDev(Ratio, Length);
```

Other calculations require the strategy to be applied to a portfolio of symbols, so we need to check and see if that's the case:

```
if 1 = getappinfo(aiisportfoliomode) then begin
// code
end;
```

For the basic strategy, we need to return the strategy index of the second instrument and check if it has been applied:

```
var: slave_idx(pmms_strategies_get_by_symbol_name(symbolname data2));
once if 0 > slave_idx then
       raiseruntimeerror(text("specified slave trader on instrument ",
doublequote, symbolname data2, doublequote, " not found"));
```

To synchronize the capital invested into positions for both instruments, we need to send the price of the current position of the main instrument to the pair strategy:

```
       value22 = absvalue(cur_pos*Contracts_) * c * bigpointvalue;
       if 0 < value22 then
              value22 = pmms_to_portfolio_currency(value22);

       pmms_set_strategy_named_num(slave_idx, "MPMoney", -cur_pos *
value22);
```

b) Сигнал Portfolio_SpreadTradingSystem.Slave Signal

This signal "b)" is calculated for the second instrument of the pair. It monitors all entries and exits generated by the previous signal "a)" for the main instrument of the pair and trades in the opposite direction. Firstly, all synchronization is done when «MPMoney» variable returned by master strategy changes .

```
value1 = pmms_from_portfolio_currency( pmm_get_my_named_num("MPMoney")
);
```

We extract this variable and convert it from portfolio currency into instrument currency. Then, based on its value, we calculate the number of contracts for potential entry positions:

```
value33 = c;
if marketposition <> 0 then
       value33 = entryprice;

master_mp = IntPortion( value1 / ( value33 * bigpointvalue) );
```

The instrument's current position:

```
my_mp = currentcontracts*marketposition;
```

Now we will check to see if its position is unsynchronized. If that's the case, then we will synchronize it with the main strategy:

```
if sign(my_mp) <> sign(master_mp) then begin
…
end;
```

We'll check if the main instrument's position has closed:

```
if 0 = value1 then begin // need close position
       if my_mp > 0 then
              sell all contracts this bar c
       else
              buytocover all contracts this bar c;
       #return;
end;
```

If it has closed, we'll close the position for the second instrument as well. If the main instrument has an open position, then we will determine the position's direction for the second instrument:

```
if 0 < value1 then begin // we must to buy
if 0 < value1 then begin // we must to buy
```

Value1 > 0 means that to synchronize the positions we should buy. There can be two cases:

1. The current flat or short position should change to long, i.e., the master strategy has reversed its position or has entered a long position from the flat state.
2. The current position is already long which means that the first instrument partially closed its short position, signifying that we need to partially close the second instrument's position.

```
       if Sign(master_mp) <> Sign(my_mp) then
              buy absvalue(master_mp) contracts this bar c
       else
            buytocover value1 contracts this bar c;
```

In the opposite case:

```
end else begin
       if Sign(master_mp) <> Sign(my_mp) then
              sell short absvalue(master_mp) contracts this bar c
       else
              sell absvalue(value1) contracts this bar c;
end;
```

Value1 < 0 means that we need to sell to synchronize the positions; there also can be two cases,

1) The current flat or long position should change to short, i.e., the master strategy has reversed its position or has entered a short position from the flat state.

2) The current position is already short which means that the first instrument partially closed its long position, signifying that we need to partially close the second instrument's position.

APPENDIX

Portfolio signals scripts are added to MultiCharts and MultiCharts64 by default.

RANK STRATEGY

This strategy can be considered a modification of the Rotation Strategy.

This strategy was suggested by Angelos Diamantis.

STRATEGY DESCRIPTION

This strategy is based on calculating that one indicator which is applied to every instrument in the portfolio. Once all indicators' values have been determined, they are organized based on high to low values. Long positions are opened for instruments with best indicator values, while short positions are opened for instruments with worst indicator values.

Let's take an example of a portfolio consisting of 35 stocks with 5-minute resolution used for trading. The same indicator (% Chg) with the following formula: "(close – close[1]) / close" is calculated on a 1-day resolution for every instrument. For 5 instruments with the highest indicator values we enter long a position. For 5 instruments with the lowest indicator values we enter a short position.

Trade size is set as either a fixed number of contracts for all instruments or a percentage of the total portfolio capital.

STRATEGY DEVELOPMENT

a) Portfolio Rank Signal Base

This signal calculates the value of the specified indicator for all instruments contained in the portfolio and saves these values using the instrument strategies' indices.

Indicator formula and data series number that will be used for its calculation are set by the user:

```
inputs:
        BasedOnData(2),
        Formula( (close - close[1]) / close ),
        TraceOutput(false);
```

We will need to add some restrictions to our signal so it can be used only for portfolio trading; the data series used for its calculation should be available to start the calculation:

```
// *** restrictions
once if barstatus(BasedOnData) < 0 then raiseruntimeerror("Portfolio
Rank Signal Base needs datastream " + numtostr(BasedOnData, 0));
once if 1 <> getappinfo(aiisportfoliomode) then
raiseruntimeerror("Portfolio Rank Signal Base can be applied for
MCPortfolio application only.");
// ***************
```

Now we will calculate our indicator using the formula and save the value for each instrument:

```
BarN = BarNumber of data(BasedOnData);

if BarN > BarN[1] then begin
       R = Formula of data(BasedOnData);
       pmm_set_my_named_num("RankStrategyR", R);
end;
```

To trade a percentage of portfolio capital instead of fixed number of lots each instrument should return the cost of each contract:

```
begin
       var: MoneyCostForInvestPerCtrct(0), otential_entry_price(close);
       MoneyCostForInvestPerCtrct =
       pmms_calc_money_cost_for_entry_per_cntrct(potential_entry_price,
       Portfolio_GetMarginPerContract)
       +
       pmms_calc_money_cost_for_entry_per_cntrct(potential_entry_price,
       Portfolio_GetMaxPotentialLossPerContract);

       if 0 > MoneyCostForInvestPerCtrct then
            raiseruntimeerror( text("Error! Price = ",
            potential_entry_price:0:6, "PMargin = ",
            Portfolio_GetMarginPerContract, "PMaxPLoss = ",
            Portfolio_GetMarginPerContract) );

       // MoneyCostForInvestPerCtrct in symbol's currency. Convert it
to portfolio currency ...
       pmm_set_my_named_num("MoneyCostForInvestPerCtrct",
       pmms_to_portfolio_currency(MoneyCostForInvestPerCtrct));
end;
```

Finally, we will generate Long and Short Entry orders. After a money management signal calculation, only a few of them will be sent (based on the strategy's logic):

```
buy next bar market;
sellshort next bar market;
```

b) Сигнал Portfolio Rank MM Signal

This signal is used for money management. It organizes all indicator values into a list and manages opening positions for the instruments based on said list. Below are user inputs which manage trade size and number of instruments for which the position will be opened:

```
inputs:
       ContractsNumber(10),
       IgnoreContractsNumberUsePcnt(false),
       PortfolioBalancePercent(1),
       BuyBestN(10),
       SellWorseN(10),
      TraceOutput(false);
```

Let us apply some restrictions to the signal: a) it can be used only in Portfolio Trading, b) portfolio size should not be higher than 10 000 instruments and c) the number of instruments should correspond to user inputs that determine the number of entries:

```
once if 1 <> getappinfo(aiisportfoliomode) then
raiseruntimeerror("Portfolio Rank Monem Management Signal can be
applied for MCPortfolio application only.");

once if pmms_strategies_count() > 10000 then
raiseruntimeerror("Portfolio Rank Monem Management Signal too much
intruments, max value = " + numtostr(100000, 0));

once if pmms_strategies_count() < BuyBestN + SellWorseN then
raiseruntimeerror("Portfolio Rank Monem Management Signal, please check
inputs, BuyBestN + SellWorseN should be less or equal to tradeable
Instruments number");
```

Save the number of traded instruments in the portfolio to a variable, and forbid opening positions to all instruments:

```
once begin
       portfolioStrategies = pmms_strategies_count();
       array_setmaxindex(BaseR, portfolioStrategies);
       array_setmaxindex(ContractsForEntry, portfolioStrategies);
end;

pmms_strategies_deny_entries_all;
```

Extract indicators' values for every instrument:

```
for idx = 0 to portfolioStrategies - 1 begin
       BaseR[idx] = pmms_get_strategy_named_num(idx, "RankStrategyR");
end;
```

Strategy indices and values are stored in the array so we can open positions for those instruments with appropriate indices after all instruments have been sorted.

Then the strategy calculates the number of contracts to open a position for every instrument. After that, the indicator values array is sorted in ascending order:

```
for idx = 0 to portfolioStrategies - 1 begin
	Value_Idx[1, idx + 1] = BaseR[idx];
	Value_Idx[2, idx + 1] = idx;

	if IgnoreContractsNumberUsePcnt then begin
		ContractsForEntry[idx] =
pmms_calc_contracts_for_entry(PortfolioBalancePercent, idx);
	end
	else
		ContractsForEntry[idx] = ContractsNumber;
end;

Sort2DArray(Value_Idx, 2, portfolioStrategies, 1 {from high to low});
```

For instruments with the highest indicator values Long Entry for the specified number of contracts is allowed:

```
variables: inLong(0), inShort(0);
array: strategyIndexes[](0);

inLong = pmms_strategies_in_long_count(strategyIndexes);
for idx = 1 to BuyBestN - inLong begin
	strIdx = Value_Idx[2, idx];
	pmms_strategy_set_entry_contracts(strIdx,
	ContractsForEntry[strIdx]);
	pmms_strategy_allow_long_entries(strIdx);

	if TraceOutput then
		print("CurrentBar = ", currentbar:0:0, ". Allow LONG for
symbol ", pmms_strategy_symbol(strIdx), ", Contracts = ",
ContractsForEntry[strIdx]);
end;
```

For instruments with the lowest indicator values Short Entry for the specified number of contracts is allowed:

```
inShort = pmms_strategies_in_short_count(strategyIndexes);
for idx = portfolioStrategies downto portfolioStrategies - SellWorseN +
inShort + 1 begin
	strIdx = Value_Idx[2, idx];
	pmms_strategy_set_entry_contracts(strIdx,
	ContractsForEntry[strIdx]);
	pmms_strategy_allow_short_entries(strIdx);

	if TraceOutput then
	     print("CurrentBar = ", currentbar:0:0, ". Allow SHORT for
	     symbol ", pmms_strategy_symbol(strIdx), ", Contracts = ",
	     ContractsForEntry[strIdx]);
end;
```

Other instruments are not traded on the current calculation.

APPENDIX

Portfolio signals scripts are added to MultiCharts and MultiCharts64 by default.

Original strategy description by Angelos Diamantis:

With regards to the rank strategy here is a short but generic description.
Assume a new class of indicators applied to the whole universe e.g. AvgReturn= (R1+R2+R3+...+R500)/500; Sdev= Standard Deviation of AvgReturn;
where Ri = Day Return of i Stock i=1 to 500 if our universe is 500 stocks of S&P
Then based on this indicator and the data this is applied to for instance Data2= Daily, Data1=5min Bars
Rank all Stocks from Highest to Lowest.

```
Vars= BarNo2(0),MyIndicator(0),R(0);
BarNo2= BarNumber of data2;

If BarNo2>BarNo2[1] then Begin
R = (C of data2 - C[1] of data2) / C[1] of data2;
MyIndicator= (R - AvgReturn ) / Sdev
end;

{Retrieve MyIndicator Rank. Rank is from 1 to 500 since our universe is 500 Stocks}

If Rank<=10 then Buy 200 contracts next bar at O; {Go Long the best 10 stocks}
Else If Rank>=490 then SellShort 200 contracts next bar at O; {Go Short the Worse 10 stocks}
```

The above is a classic case of Stocks Relative Performance Trading
MyIndicator should be generic, meaning that the user should be able to change this Ranking Indicator as he wishes. Another Example of Ranking Indicator might be MyIndicator = ADX of data2; Then allow trading only in those stocks that have the highest ADX