Testing Kandle-based strategies using an ocaml simulator

Mangrove Research

October 26, 2024

You need to install ocaml.
Eg via the opam package manager

1 Setup price series

To run a test we first need to get a price series.
There are two options.
We can use a csv file of historical data (see `h` function):

```
let h
~number_of_lines:nb_lines (* nb_lines to read after header *)
~filename:filename        (* path to csv file *)
~column:col               (* index of price column in file *)
~splitchar:c              (* split char in the csv file *)
= ...
```

Or we can use a BM or GBM with parameters:

```
let gbrowmo
~initial_value:ival       (* initial price value *)
~drift:drift              (* drift of BM *)
~volatility:vol           (* volatility of BM *)
~timestep:dt              (* integration step *)
~duration:duration        (* total duration *)
= ...
```

Both methods output a price series as an array of floats.

2 Setup Kandle

To define a Kandle instance one needs:
1) a (relative) price grid
2) an initial allocation of capital (in quote),
3) and the fraction α of the said quote to be swapped to base (to provide Kandle's asks)

The relative price grid is defined by picking any 2 of the following 3 parameters:
- (half) number of points,
- range multiplier,
- gridstep (aka ratio)

The initial price (or entry price) used to center the grid is taken from the price series (hence the name 'relative' price grid).

3 Setup simulation

A simulation is the interaction of a Kandle with a price series.

The output of a simulation run is a 6-uple:
- the return
- the number of up crossings and up exits (above range)
- the number of down crossings and down exits (below range)
- the final price of the price series

To set up a run of the simulation, we combine both sets of parameters (using here range and gridstep for the price grid).

Inputs:

```
let sim
~rangeMultiplier:rangeMultiplier (* pmax/p0 = p0/pmin *)
~gridstep:gridstep               (* ratio of price grid *)
~quote:qB                        (* total budget in quote *)
~cashmix:alpha                   (* 0 <= alpha <= 1 *)
~start:start                     (* entry in position *)
~duration:duration               (* number of price moves *)
~price_series:price_series       (* series of price *)
```

Outputs:

```
(mtmf /. mtmi -. 1.),
!rebu,
!cross_above,
!rebd,  !cross_below,
price_series.(start + duration - 1)
```

Fig. 1 shows an example of a 1000 runs, 10000 steps each.
The syntax is

```
./a.out 10_000 10 1.25
```

with arguments (in this order): number of repetitions, number of points on the grid, and ratio of the grid.
The compiled code will generate a csv file `return_vs_final_price_scatter_1.25_10_stopped.csv`.
The csv file can be visualised using a python script called `hist.py` with similar syntax `python hist.py 10 1.25`.

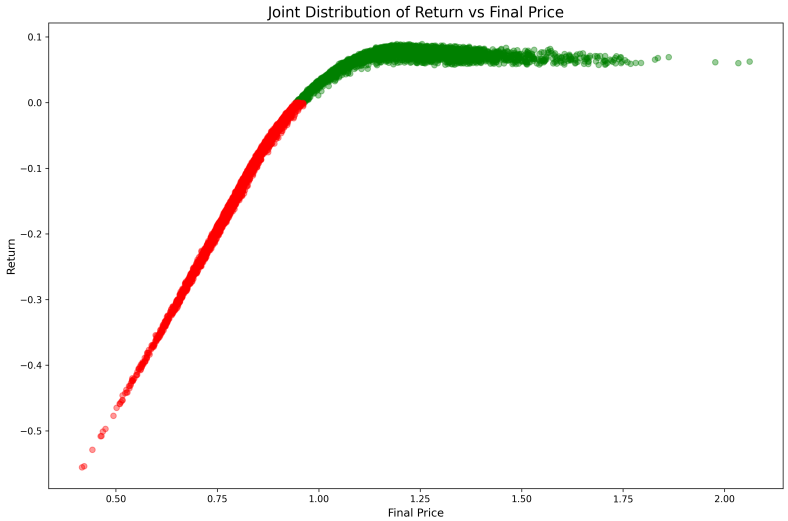

Figure 1: *On the x-axis the final price, on the y-axis the return. In red (green) realisations with a negative (positive) return.*

4 Setup simulation with exit

There is a variant, called `sim_stopped`, that will stop if the price down exits the range. So no need for specifying a duration. Either the simulation will stop (an return the time of exit and price), and it will run all the way to the end of the price series.

Inputs:

```
let sim_stopped
 ~rangeMultiplier:rangeMultiplier
 ~gridstep:gridstep
 ~quote:qB
 ~cashmix:alpha
 ~start:start
 ~price_series:price_series
```

Outputs:

```
mtmf /. mtmi, (* growth rate, because more compositional *)
!rebu, !upper_exit,
!rebd, !lower_exit,
!current_index,
!current_price
```

NB1 - `sim_stopped` returns the growth rate `mtmf /. mtmi` not the return `mtmf /. mtmi` -1.

Fig. 2 shows an example of a 1000 runs, 10000 steps each (under 2sec of execution, using the native compiler ocamlopt). We see the green and read walls at the ends of the range near the $\pm 20\%$ mark.

5 Setup simulation with reset

There is also a `sim_reset` variant where each time the price exits the range, the strategy re-enters with the same grid translated to the new current price.

Figure 2: *On the x-axis the final price, on the y-axis the return. The Kandle grid has 4 points (on each side of the entry price of* 100*), with a ratio of* 1.1*.*