LAMBRECHT Grégoire

Y-Intercept-Test

1 Data Processing

In the given data file there is a csv file with values of prices and volumes for different shares. There are 248 different shares. Data for each share are given at different dates, on 2994 different days. The first date 2013-01-04. To simplify we suppose that the volume and the price of a share at a date t_1 are equal to the last volume and last price recorded before t1. By this process we get data for all tickers over the 2994 days.

In the file, the function plotData(int i) displays the price and volume over the 2994 days of the ith share. The list last contains 248 lists containing the different prices for the 248 shares. The same for the list volume.

2 Naive Strategy

2.1 Strategy for a Stock Composed by One Share

To start, we try to implement a strategy to decide when buy and sell a unique share according to the evolution of its price and its volume. In this strategy every day we take into account market values over the previous T days, whit $T > 0$. Let's considering a single share.

The strategy is the following :

1. If the volume is lower than its average value of the previous T days, we don't do anything. We consider that price moves made on low volume may be said to "lack conviction" and are viewed as being less predictive of future returns.
2. Else
   (a) If the price is lower than its average value of the previous T days, we sell shares in a proportion p
   (b) Else we buy new shares

In the file strategy.py we set up the model. In the first function SimpleStrategy(tick, x, dayConsider, buy, propSale, end) we implement the strategy. According to which share the startegy is applied, results are variable. Two different examples are illustrated in the file comparaison : one with positive results, the other with negative results.

2.2 Strategy for a Stock Composed by Several Shares

We assume that the price of shares are decorated (which is clearly not true), and we do not favor any action over another. We apply the previous strategy simultaneously to each share. In the file strategy.py we set up the model. In the function PortFolioSS(initInvest,dayConsider,propSale,end) we implemented the strategy. We run PortFolioSS(10000,3,1/2,1000) in the file comparaison.py and the strategy seems to work according to the graph.

3 Black and Scholes Model

3.1 The model

In this strategy we use the Black and Scholes model. Let $(S_t)_{t \in \mathbb{R}*}$ the price of a share at the time $t \in \mathbb{R}$. Assume that there exist $(\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_+$ such that

$$S_t = s_0 exp((\mu - \frac{\sigma^2}{2})t - \sigma W_t)$$

with s0 the initial price at time t=0, and (W_t) a brownian motion.
The expected value is $\mathbb{E}(S_t) = s_0 \times e^{\mu t}$. So if μ is non negative, the model predicts that the price will tend to rise. So we should invest.
Then in this strategy we try to obtain the value of μ to decide if we should buy or sell shares.

Let $t_0 = 0 < t_1 < ... < t_N$ a sequence of time such that for all i $t_{i+1} - t_i = \Delta t$. Let l_i the real price observed at time t_i. We know that

$$\boxed{\Delta S_i = log(S_{t_{i+1}}) - log(S_{t_i}) \stackrel{Law}{=} \mathcal{N}((\mu - \frac{\sigma^2}{2}) \times \Delta t, \sigma^2 \times \Delta t)}$$

(Normal law of expected value $(\mu - \frac{\sigma^2}{2}) \times \Delta t$ and variance $\sigma^2 \Delta t$)

At the i-th time t_i we introdue

$$\Delta l_i = log(l_{i+1}) - log(l_i)$$

$$\bar{\Delta S_i} = \frac{1}{i} \sum_{j=0}^{i-1} \Delta l_j$$

$$V_i = \frac{1}{i} \sum_{j=0}^{i-1} (\Delta l_j - \bar{\Delta S_i})^2$$

By the law of large number we know that $\bar{\Delta S_i}$ and V_i are estimators of $(\mu - \frac{\sigma^2}{2}) \times \Delta t$ and $\sigma^2 \times \Delta t$. So we approximating μ by $\frac{\bar{\Delta S_i} + \frac{V_i}{2\Delta t}}{\Delta t}$. Notice that the more time passes the more our model is faithful because we have more and more data.

3.2 Strategy for a Stock Composed by One Share

Now that we know how approximate μ, we have to erect a strategy : when to buy and when to sell ?

Assume that we take a moove every time $(t_{T \times i})$ with $T \in \mathbb{R}_+$. Let $\bar{l_i} = \frac{1}{i} \sum_{j=0}^{i-1} l_j$ the emperical mean of the prices observed at time t_i. Let $p \in (0, 1)$ the proportion of sale, $n_s \in \mathbb{R}_+$ an amount of money to invest at each moove. We adopt the folowing strategy at each time $(t_{T \times i})$:

1. If $\Delta \bar{S}_{Ti} > 0$ we buy shares for n_s dollars
2. Else if $\Delta \bar{S}_{Ti} < 0$ We sell in p proportion our shares.

In the file BlackScholes.py we set up the model. In the first function BlackScholes(tick,x,begin, end,mooveTime,buy,propSell) we implement the strategy. We run two different examples in comparaison.py. On these examples the strategy seems to work according to graphs.

One of the possible improvements would be to take volume into account.

3.3 Strategy for a Stock Composed by several shares

We assume that the price of shares are decorated (which is clearly not true), and we do not favor any share over another. We apply the previous strategy simultaneously to each share. In the file BlackScholes.py we set up the model. In the function PortFolioBS(initInvest,begin,end,mooveTime,buy,propSell) we implement the strategy. We run an example in comparaison.jl, and the strategy seems to work according to the graph.

3.4 Defaults of the Strategy

This model presents a lot of defaults. Firstly it doesn't consider the volume. It does not react to the sudden movement of the market. Next it does not take into account the correlation between stocks. So the strategy is risky because we take the risk of having an undiversified portfolio. However this risk is reduced by the fact that we invest uniformly in all stocks.

The comparaison made in the file comparaison.file prove that our strategy using the model of Black and Scholes is better that our naive strategy.