Randomised Signal Acceptance

Probability of Signal Acceptance. We introduce an important parameters, this is called the confidence parameters, since we have to make random choices which may be influenced by our confidence, we can vary these parameters. Let P_b and P_s, be our respective confidence buying and selling parameters. With the help of randomization we can also reduce the number of trades! Which means we are charged way less on transaction fees, but the frequency of your trading also depends on the type of strategy we have too.

We generate a probability from a uniform distribution. We then assign our values. These values can be considered as the acceptance rate for the buy or sell actions. The code is available in Simulations.py, it is the "BackTestGoingLongRandomised" function. You can have a play around with it to to see how these values impact the trading frequency too.

Simulation of Randomized strategy:

1. Initialize Starting amount, and initialize starting point. (Point should ensure that you can work with the maximal amount of points, or from a common point if you are comparing strategies). Initialize your Sell Signal as 0, Buy Signal as 1. Assign your value in between 0 and 1 for your buy and sell threshold value.
2. While the index is less than the size of the total number of prices, pass in data up till that index to your strategy function.
3. Generate a random a number in between 0 and 1.
4. Strategy function should either return 0,1,-1. 0 means no Signal was generated. 1 means a buy signal was generated, 0 means a sell signal was generated.
5. Act upon that action signal and check if the random number generated is below the assigned thresholds.
6. For every transaction charge a %0.005 fee of the total amount involved in the transaction. So when selling, the amount you receive will be %0.005 less, when buying the amount you buy with will be %0.005 less.
7. Continue, until we run out of points.

$1 - \alpha$ is the probability that the signal generated will be accepted. If α is a very large value, then signals are more likely to be acted on when they occur in high frequencies. In the simulation, when using moving averages of relatively small windows on hourly time frames there were a lot of signals being produced, and thus I needed a way to filter high quality signals and hence an easy way to do that was to randomly choose signals.

This concept of accepting signals is somewhat relevant to fields such as reinforcement learning and other randomization algorithms.

I will most likely extend this algorithm into something more meaningful later on.