Choosing trading strategies in electronic execution using importance sampling

Quants often need to develop algorithmic decision rules for selecting execution strategies based on trade characteristics. While A-B testing is commonly used to evaluate potential decision rules, it can be inefficient and limited in comparing multiple rules. In this paper, Stuart Baumann demonstrates how importance sampling can provide a more efficient method for comparing and fine-tuning decision rules

Electronic execution desks at banks offer a large number of trading algorithms to clients. These algorithms may differ in the venues they trade on, the speed at which they trade, the aggressiveness of the prices and volumes quoted, and the extent to which they cross the spread. To improve their performance, brokers will often try to select the best algorithm to execute a given order. The best algorithm will generally differ for different types of trades. For instance, high-liquidity stocks should be traded differently from small caps. Buy-side execution desks face a similar problem: different brokers offer different algorithms and have access to different venues, and the optimal broker will differ for different orders. In addition, buy-side execution desks may also need to select the algorithm that their chosen broker should use (unless they delegate that choice to the broker itself). These choices are a central factor in the execution market, with buy-side execution desks constantly monitoring the performance of brokers and choosing which orders, and how many, to send them. On the sell-side, execution desks are always trying different ways of improving their performance so as to receive more order flow from the buy-side. One way this can be done is by choosing strategies that are better tailored to each incoming order.

In general, both buy-side and sell-side execution teams need to choose decision rules. For each order, a strategy (which might be an algorithm, a broker or a venue, depending on the context) must be selected. It can be difficult to assess the performance of all the decision rules that are possible to implement. The simplest method – running each decision rule in parallel and comparing the average realised slippage from each, known as A-B testing – has three key disadvantages. The first is that the necessity of running each decision rule means that only a limited set of decision rules can be evaluated. The second is that assessing decision rules using disjoint sets of orders leads to more noise than necessary. We can achieve greater statistical efficiency by estimating the slippage of all decision rules using all orders. In addition, if we assess decision rules on the same set of orders, then outlier orders affect all decision rules equally and are thus less of a factor when making comparisons. In this sense, A-B testing is inefficient. The third disadvantage is that all ideas to be trialled need to be established before trading takes place. This paper illustrates how these problems can be overcome using importance sampling.

In particular, we assess the performance of each decision rule using a technique known in the reinforcement learning literature as off-policy policy evaluation (Doroudi *et al* 2018; Metelli *et al* 2020; Sutton & Barto 2018). This technique allows us to estimate the expected rewards from undertaking certain actions given a data set that was the result of undertaking potentially different actions. Applying this to the context of trade execution, we can estimate the expected slippage of a potential decision rule without having run this decision rule in a production environment. We use the same data set of historically executed orders to estimate expected slippage from many possible decision rules.

Reinforcement learning has been increasingly applied to problems in finance in recent years (for a review see Ritter (2017)). In particular, it has been applied to the hedging of option positions (Kolm & Ritter 2019), to mean-reversion trading strategies (Kolm & Ritter 2020) and to trade execution. In the field of trade execution, it has been applied to learn the best way to post limit orders in both equity and cryptocurrency markets (Nevmyvaka *et al* 2006; Schnaubelt 2022). There are many more such studies.

While these reinforcement learning strategies are often innovative and can perform well, they can also be difficult for some organisations to implement. Banks typically have a codebase that is highly complicated, connects to many venues and has been built over many years, and they may be reluctant to make large changes to this established infrastructure. There is also a significant learning curve associated with reinforcement learning, which may hinder widespread adoption of these methods within a large organisation such as an investment bank. By contrast, the importance-sampling technique that we suggest does not require infrastructure changes to be implemented. The technique can be implemented by a single quant researcher without changes being made to the production code. The only requirement is that the likelihood of a particular strategy being selected has been appropriately recorded in the data set of historical trades.

In this short paper, we first briefly recount the essential concepts of importance sampling. Next, a simple Monte Carlo example is given, both to present a representative importance-sampling calculation and to illustrate the efficiency gains that are possible. We then discuss how importance sampling is a preferable technique for estimating the expected slippage of a decision rule, rather than estimating slippage as a function of order attributes for each strategy and then integrating over the result. Functional approximation methods can be a useful technique for creating a decision rule, however, so in the final section we show an example of how importance sampling and functional approximation can be used in a realistic setting.

Importance sampling

The key equation for importance sampling is:

$$E[\theta \mid p, X] = \int \theta p(\theta \mid X)\,\mathrm{d}\theta \mid X$$
$$= \int \theta \frac{p(\theta \mid X)}{q(\theta \mid X)} q(\theta \mid X)\,\mathrm{d}\theta \mid X \qquad (1)$$

where θ is the quantity we are integrating to get the expectation (in an execution case, this might be slippage), $p(\theta \mid X)$ is the probability density of

$\theta \mid X$, $p(\theta \mid X)/q(\theta \mid X)$ is the likelihood ratio and $q(\theta \mid X)$ is a sampling distribution that is potentially different from $p(\theta \mid X)$. Importance sampling is useful because we can sample from $q(\theta \mid X)$ and then use the likelihood ratio to calculate the value of the integral as if we had sampled under $p(\theta \mid X)$.[1] This basic idea has been applied in several areas of finance. In derivatives pricing, it appears in the form of a change in probability measure with the Radon–Nikodým derivative. In Monte Carlo simulations (such as those for XVA pricing or the pricing of exotic derivatives), this technique is used as a variance-reduction method.

Our goal is to choose a trading strategy for each trade that we are going to execute. Put more formally, we seek a 'decision rule', which is a function $\boldsymbol{X} \rightarrow \boldsymbol{F}$, where $\boldsymbol{F}$ is a multinomial distribution that gives the probability that a given strategy will be chosen and $\boldsymbol{X}$ is a vector of the order attributes that we use to assign a strategy to a trade. These attributes might include spread, volatility, anticipated alpha, average daily volume, order size and anything else that might inform how the order should be traded. We assume we have chosen such a set of attributes that form the $\boldsymbol{X}$ vector, and we further assume that we have a set of trading strategies to choose from, which constitutes the domain of the possible $\boldsymbol{F}$ distributions.

We can adapt the general importance-sampling equation (1) into a format that can be used to calculate slippages from a set of N historical orders:

$$E[\text{Slippage} \mid p] = \frac{1}{N}\sum_{i-1}^{N} \text{Slippage}_i \frac{\Pr(a_i \mid X_i, p)}{\Pr(a_i \mid X_i, h_i)} \tag{2}$$

where Slippage_i, a_i and X_i denote, for order i, the slippage, the strategy chosen and the observables with which the strategy was chosen; h_i is the historical decision rule used to choose the algorithm for order i; and p is the decision rule for which we want to estimate slippage.

Before presenting a concrete case in which (2) is implemented, it is worth highlighting that, in order for this equation to be identified, we need $\Pr(a_i \mid X_i, h_i) > 0$.[2] This is akin to the requirement in importance sampling that the two distributions (p and q in the notation of (1)) agree on which draws have a zero likelihood.

In the sections that follow, we enumerate potential strategies with Roman numerals (so we talk of strategy I, strategy II, etc). We use the convention that slippage resulting from more adverse prices is negative, and hence higher (or less negative) slippage is desirable. We use the notation $S(x_i, a_i) = E[\text{Slippage} \mid x_i, a_i]$ to give the expected slippage using strategy a_i for an order with attributes x_i. Functional approximations are marked with a hat symbol, $\hat{}$, and, where necessary, a subscript detailing the method used to approximate the function. As an example, an approximation of the slippage of strategy a_i as a function of x using ordinary least squares (OLS) would be written as $\hat{S}_{\text{OLS}}(x, a_i)$.

A simple Monte Carlo example

In this section, we consider four strategies that we want to choose between using a decision rule. To make this first example as simple as possible, we will choose strategies based on only one uniformly distributed observable $x \in \text{UD}[0, 1]$.[3] We assume a data generation process (DGP) whereby each strategy has an expected slippage given by:

$$E[\text{Slippage} \mid x, a] = S(x, a) = -\alpha_a[\sqrt{x} + (x - \beta_a)^2]$$

where a is the index for each strategy. We have $\alpha_a = 5, 10, 15, 20$ and $\beta_a = 0.2, 0.4, 0.6, 0.8$ for each of the four strategies, respectively. These expectations are the true expectations that are not observable by us.

The total slippage of an order we observe is given by:

$$\text{Slippage} = S(x, a) + N\left(0, \frac{d\sigma}{2}\right) \tag{3}$$

where the noise term $N(0, d\sigma/2)$ is roughly what we would expect from trading through time at a uniform rate (ie, a time-weighted average price execution) of a stock with volatility σ over duration d. For simplicity, we will consider that all stocks have the same σ, but the duration d of orders is $d \sim \text{UD}[0, 1]$.

Finally, we have a data set of orders that have been previously executed. The decision rule that generated the orders has the logic that if $x < 0.5$, then the probability of each algorithm being chosen is $[0.625, 0.125, 0.125, 0.125]$, and if $x \geqslant 0.5$, then the probability of each algorithm being chosen is $[0.125, 0.125, 0.625, 0.125]$. We will call this the production decision rule. Given our assumed DGP, we can work out that the true expected slippage (which we will seek to estimate) is -9.01.

We are not sure if this previous decision rule is useful and want to estimate slippage if we had merely chosen all four strategies with equal probability (the 'equal weight' decision rule).[4] We also want to see the probability that the previous decision rule actually outperforms the equal weight decision rule.

We can use importance sampling to do this, and for exposition's sake we will do this with the small data set in table A. The probabilities in column '$\Pr(a_i \mid x_i, h_i)$' are the same probabilities as implied by the production decision rule because this was the decision rule used to generate them. This implies that, for the production decision rule, the likelihood ratios will be 1 for all these orders.

For the equal weight decision rule, however, the likelihood ratios will not be 1. Instead they will be $[2, 2, 2, 0.4, 0.4]$ for the five orders in the table.[5] By applying (2), we can calculate the expected slippage of the equal weight decision rule as:

$$\frac{2(-5.49 - 0.32 + 0.86) + 0.4(-1.87 + 0.03)}{5}$$

A. Example data for importance-sampling calculations

X	Duration	a_i	$\Pr(a_i \mid x_i, h_i)$	Slippage_i
0.27	0.53	IV	0.125	−5.49
0.37	0.68	II	0.125	−0.32
0.57	0.38	IV	0.125	0.86
0.91	0.95	III	0.625	−1.87
0.2	0.12	I	0.625	0.03

[1] *As well as importance sampling, this approach is also known as inverse probability weighting.*

[2] *If this is not true, it might suggest a data issue, as it implies that a strategy was chosen, while our data indicates that its selection was a zero-probability event.*

[3] *The restriction to one dimension is not essential. It simplifies the mapping* $\boldsymbol{X} \rightarrow \boldsymbol{F}$ *so is easier for exposition, but it has no impact on the importance-sampling logic.*

[4] *Given our assumed DGP, we can work out that the true expected slippage (which we will seek to estimate) is* -10.

[5] *The first three entries come from dividing* 0.25 *(the probability of a strategy being chosen under equal weighting) by the historical probability of* 0.125. *The fourth and fifth entries come from dividing* 0.25 *by* 0.625.

which gives −2.13. As a result of the likelihood ratios all being 1, the expected slippage of the production decision rule is the simple arithmetic average of the five slippage values in the table, which is −1.36.

With similar calculations, we can estimate the slippage we would get from other possible decision rules. We will consider the preceding two decision rules as well as a third (the 'distant' decision rule) that has the logic that if $x < 0.5$, then the probability of each algorithm being chosen is $[0.125, 0.125, 0.125, 0.625]$, and if $x \geqslant 0.5$, then the probability of each algorithm being chosen is $[0.125, 0.625, 0.125, 0.125]$.[6] We call this decision rule 'distant' as the probabilities with which each strategy is chosen are very different from the probabilities from the production decision rule.

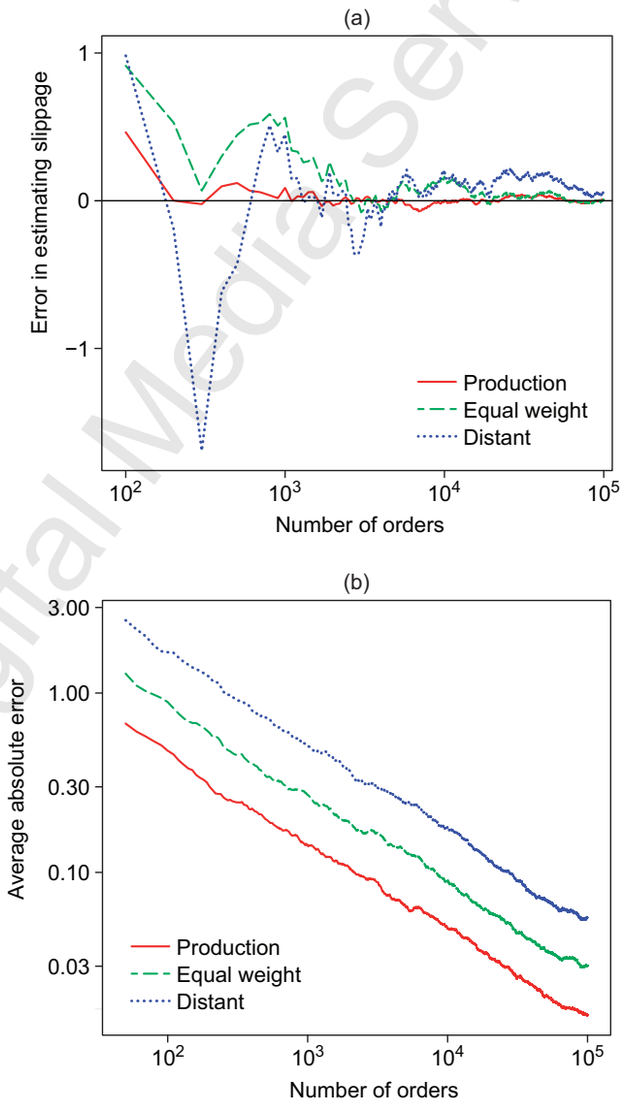
In figure 1(a), we plot one path showing the convergence of each decision rule to its true expectation. It is notable that the distant decision rule has a more erratic path towards convergence. This occurs because this decision rule is substantially different from the production decision rule that was used to generate the data set. This means that the likelihood ratios are generally far from 1, and hence certain orders can be weighted highly in calculating expected slippage under this decision rule. This slower convergence can be seen more clearly in part (b), where we average over 200 paths to see the average error.

We can use these importance-sampling techniques to calculate the probability that a given decision rule has a better expected slippage than another. In a standard A-B testing case, a t-test would be appropriate for establishing statistical differences in means; however, that approach will not work with importance sampling. The reason is that, while (1) works to calculate expectations, it does not imply that the standard deviation of θ will be comparable to the standard deviation of $\theta p(\theta \mid X)/q(\theta \mid X)$. What we can do instead to infer statistical significance is to bootstrap the difference between $E[\text{Slippage} \mid p]$ and $E[\text{Slippage} \mid q]$ based on our data set; the probability that $E[\text{Slippage} \mid p] > E[\text{Slippage} \mid q]$ is the fraction of bootstrapped differences that are positive.

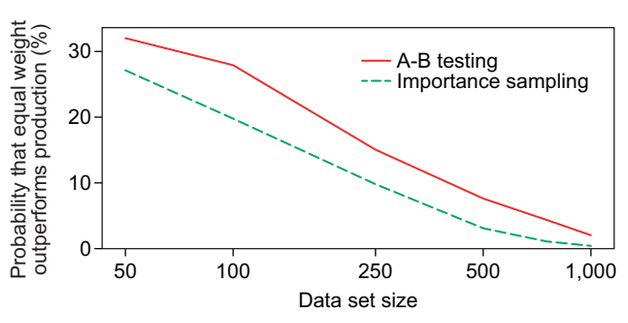
This is done to create figure 2. Specifically, we calculate the probability that the expected slippage from the equal weight decision rule is higher (ie, less negative) than the expected slippage from the production decision rule. We do this for data sets of various sizes, with 200 different data sets generated for each size. The results are averaged together and can be seen in the figure.

It can be seen that the probability of the equal weight decision rule outperforming the production decision rule approaches zero for large data sets, reflecting that it is a worse decision rule. Using importance sampling, we arrive at this conclusion with less data than A-B testing would need. This method has several other advantages over A-B testing. The first is that each slippage value is calculated from twice the number of orders, as we do not need to split our sample into two to run an A-B test. The second is that, because the orders used to assess each decision rule are the same, the impact of outliers gets cancelled out to some extent when we calculate the difference.[7] These factors outweigh the advantage of A-B testing, which is that orders are generated from the appropriate sampling distribution and hence all likelihood ratios are 1, which lowers the variance of the expected slippage estimate.

1 (a) The errors in estimating convergence to the true expectation along one path, and (b) the average error (over 200 paths) at each given number of orders



2 Probability of one decision rule outperforming another under each testing method



[6] *Given our assumed DGP, we can work out that the true expected slippage (which we will seek to estimate) is* −11.5.

[7] *This, however, is a secondary effect in this Monte Carlo example, since there are no extreme outliers as a result of us assuming a normally distributed price diffusion in (3). In real data sets, where stocks do jump over the course of a trading day, this is a larger factor.*

Functional approximation methods

Another way in which we might find the expected slippage of a decision rule is to rewrite it as:

$$E[\text{Slippage} \mid x, p] = \sum_{a \in A} S(x, a)\, p(a \mid x) \tag{4}$$

where A is the set of possible strategies and $p(a \mid x)$ describes each of the strategies assigned by decision rule p given each set of order attributes x. We can then approximate $\hat{S}(x, a)$ as a function of x for each strategy and use these approximations and (4) to approximate:

$$\hat{E}[\text{Slippage} \mid x, p] = \sum_{a \in A} \hat{S}(x, a)\, p(a \mid x)$$

as a function of x. Finally, we can compute the integral:

$$E[\text{Slippage} \mid p] \approx \int_{x \in D(x)} \hat{E}[\text{Slippage} \mid x, p] d(x)\, \mathrm{d}x \tag{5}$$

where $D(x)$ is the domain of order attributes and $d(\cdot)$ is the probability density function of order attributes x. Equation (5) estimates $E[\text{Slippage} \mid p]$ without the use of importance sampling and is an alternative to the importance-sampling method described in (2).

This approach has some practical issues, however, which limit its use in practice. The main one is that we do not know $S(x, a)$ and can only create a functional approximation $\hat{S}(x, a)$. There are many ways to do this, but for each, we will face the bias-variance trade-off (Hastie *et al* 2009). We could choose a linear model, for instance, which would have low variance but be biased to the extent that $S(x, a)$ in nonlinear in x. The use of a biased estimator could lead to bias in the estimate of the expected slippage of the decision rule. We could also choose a nearest neighbours approach, which would have little bias but high variance and be highly impacted by noise in the training set.

We also have the problem that, given estimates $\hat{S}(x, a)$ for each action, we may want to use these estimates to choose the best strategy to use at each point. So, in many practical cases, $p(a \mid x)$ will be affected by errors in estimating $\hat{S}(x, a)$, and as a result of this noise we will tend to overestimate the possible gains from the decision rule.

To see this, consider that at a point x_0 we are choosing between two strategies, I and II, each of which delivers equal slippage in expectation, so $S(x_0, \text{I}) \equiv S(x_0, \text{II})$. Our estimators $\hat{S}(x_0, \text{I})$ and $\hat{S}(x_0, \text{II})$ are unbiased but do have some variance, so that:

$$\hat{S}(x_0, a) = S(x_0, a) + \varepsilon_a \quad \text{for } a \in \{\text{I}, \text{II}\} \tag{6}$$

where $\varepsilon_a \sim N(0, \sigma_\varepsilon^2)$ is a normally distributed random draw (with some variance σ_ε^2) for each of the two strategies. This random draw reflects that $\hat{S}(x_0, \text{I})$ was trained on a different data set of executions (ie, the executions that were undertaken using strategy I) than $\hat{S}(x_0, \text{II})$.

We will choose the strategy that delivers the higher slippage, so:

$$p(\text{I} \mid x_0) = 1 \iff \hat{S}(x_0, \text{I}) > \hat{S}(x_0, \text{II})$$

In this setting, however, we get $\hat{S}(x_0, \text{I}) > \hat{S}(x_0, \text{II})$ only if $\varepsilon_\text{I} > \varepsilon_\text{II}$. Further, the expectation of the higher ε draw will be positive, as the first order statistic of two draws from a mean-zero normal distribution is positive. This is a bias that will be incorporated into the expected slippage of the decision rule. In addition, the more strategies that we choose between, the greater the extent of this bias.

As a final problem, in a realistic setting, we may have many order attributes that are informative as to which strategy should be chosen for each order. If we need to estimate $\hat{S}(x, a)$ when x is high-dimensional, however, we will run into the curse of dimensionality. We may not have enough data to reliably estimate $\hat{S}(x, a)$ in this setting.[8]

Put simply, using (4) and (5) to estimate the expected slippage of a decision rule requires additional assumptions relative to the importance-sampling method – we need to determine how to model the $S(x, a)$ functions. These assumptions can have a large impact on the accuracy of the expected slippage estimate.

Using functional approximation and reinforcement learning to find a decision rule

While the use of functional approximation to estimate expected slippage from a decision rule is problematic, it is a reasonable idea to inform the choice of a decision rule. This section shows a simple example of how decision rules may be chosen using functional approximation. We also show that, since importance sampling does not use the estimated functions, it can be used to measure in an unbiased way the performance of each decision rule constructed from functional approximation.

To explore this, we will undertake another Monte Carlo exercise, with expected slippages derived from the Rashkovich & Verma (2012) impact model. This model returns an expectation of slippage as a function of the duration of the spread, the volatility, the ratio of the order size to the average daily volume traded and the period over which the order is traded. We will use these four order attributes as the inputs to the decision rule that assigns to each strategy its probability of being chosen. For the variance of realised slippage around this expectation, we will use the same format as (3), which can be justified as coming from price movements when the order is executed at a uniform rate for the duration of the order.

To ensure our orders reflect the kinds of order that are encountered in an institutional setting, we will consider that durations are uniformly distributed between half a day and a full day,[9] and that order sizes are drawn from a uniform distribution between 1% and 5% of average daily volume. We will jointly sample volatilities and spreads from a daily data set of Standard & Poor's 500 (S&P 500) constituents in the period from January 1, 2003, to December 31, 2022.[10] Each order that we encounter is characterised by

[8] *For more than a few dimensions we would additionally find that it is faster to integrate under this function using Monte Carlo simulation rather than more direct numerical integration techniques. If we are going to select points of the $\hat{S}(x, a)$ function for a Monte Carlo simulation, however, we might as well skip the functional approximation step and calculate the expected slippage of a decision rule using (2) as well as the data that we have collected.*

[9] *Note that, while some buy-side institutions will need to buy or sell equities over a longer period, it is common for them to split the order, such that the order a particular broker will trade is rarely of a duration longer than a day.*

[10] *Specifically, this data set has an observation for every S&P 500 constituent on every trading day in this interval. Volatilities are daily volatilities calculated based on the previous 30 days, while spreads are measured from bid to ask. We filter out volatilities that are more than 100% or less than 1% and spreads that are less than 1 basis point or more than 500bp.*

having the four attributes mentioned above, which we will denote by:

$$X = \left\{\text{spread, volatility, } \frac{\text{order size}}{\text{average daily volume traded}}\text{, duration}\right\}$$

In this Monte Carlo setting, we assume that there are four possible strategies, each of which corresponds to one of the four inputs of the Rashkovich & Verma (2012) model. Specifically, we assume that each strategy effectively changes the inputs of the model such that one input is 10% less extreme and the other three are 10% more extreme.[11] The first strategy (strategy I) executes an order with expected slippage equal to what the Rashkovich-Verma model would return for an identical order with the spread 10% less extreme (lower) and the other three inputs 10% more extreme. Strategy II has less extreme volatility, strategy III has a less extreme ratio of the order size to the average daily volume traded, strategy IV has a less extreme order duration, and in each case the unmentioned inputs are more extreme. These are, in some senses, artificial assumptions, but they are sufficient for each strategy to be effective on a different set of orders, which provides a setting in which to examine how importance sampling can identify potential performance gains.

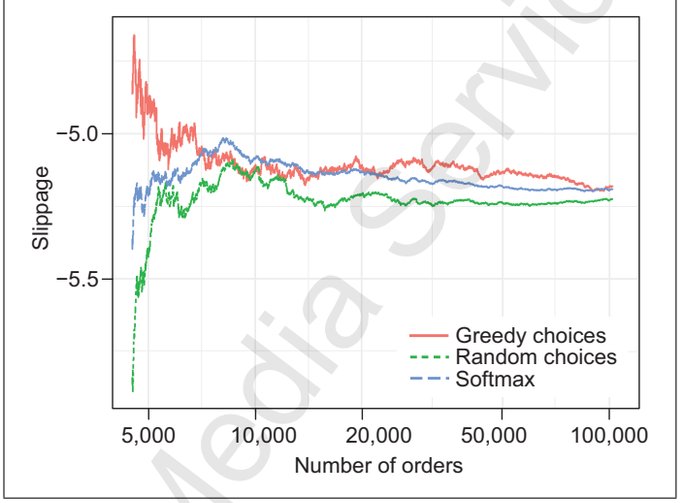
We assume that the four strategies have been running in production for an extended period. We initially execute 2,000 orders by randomly choosing from the four strategies. This ensures that we have around 500 observations for each strategy. We then implement a decision rule for the next 100,000 orders. For each order, with attributes x_i, we calculate $\hat{S}(x_i, \text{I})$ by taking all previous orders executed using strategy I and finding the average slippage of the k nearest neighbours to the point x_i (with k chosen as 200).[12] We similarly estimate the slippage from the other three strategies. Finally, we want to balance exploration and exploitation, so rather than choosing the best slippage from these four estimates, we instead use the softmax function to find the probability of sampling each strategy (Sutton & Barto 2018, (13.2)):

$$\Pr(a_i \mid X_i) = \frac{\exp(\hat{S}(x_i, a_i)}{\sum_{a_j \in \{\text{I,II,III,IV}\}} \exp(\hat{S}(x_i, a_j))} \tag{7}$$

The probability that we use a particular strategy comes from a 90% weight on the output of this function plus 2.5% for each strategy.[13]

These assumptions regarding the design of decision rules that have been run in production are not essential and will differ in every institutional setting. The two important factors are that the probabilities of each strategy being chosen need to have been recorded, and that the closer these historical probabilities are to the probabilities of hypothetical decision rules, the lower the variance of the estimates we calculate.

3 Performance of k-NN decision rule



As a result of this approach, the probability that we will use a particular strategy varies order to order. This happens both because different orders have different attributes, x_i, and also because, as time goes on, there is a larger data set that can be used to assign nearest neighbours to an order. We might also expect that this k nearest neighbours (k-NN) method will improve as the data set increases, as it will allow for closer neighbours. We can see the performance of this decision rule in figure 3. The realised performance can be seen in the 'softmax' line, while the 'greedy choices' line is the result of using importance sampling on our realised data to estimate the slippage we would have achieved if we had abandoned exploration and instead always chosen the strategy with the best predicted slippage. The 'random choices' line is the result of using importance sampling to predict the result from selecting each strategy with 25% probability.

It is encouraging that, despite the large amount of noise when the number of orders is small, the greedy decision rule does appear to outperform the random choice in figure 3, as this implies that our k-NN algorithm tends to prioritise better strategies.

While the fact that greedy choices lead to better slippage than random choices suggests that k-NN is useful in choosing strategies, we cannot tell if we can do any better than k-NN. One factor to consider is the bias-variance trade-off (Hastie *et al* 2009). While the nearest neighbours method does not impose a functional form, it does impose some bias, since some of the 200 neighbours it averages over may not be sufficiently close. There is also high variance in that only 200 neighbours are used, which may mean that our estimate is adversely influenced by outliers. So we may instead want to assess the performance of alternative decision rules. One simpler decision rule we can try is to not condition on the order attributes x_i but instead simply estimate slippage from the mean slippage of each strategy. We could also try an OLS regression (predicting slippage as a function of the four inputs, their squares and the cross products) or a decision tree.

Given each potential decision rule, we calculate the probability of choosing each strategy when using softmax probabilities or when greedily selecting the best strategy in each case. We then use (2) to estimate the expected slippage of these alternative decision rules. The results can be seen in table B.

We can see that, while our production decision rule does add value relative to randomly selecting strategies, it is outperformed by other techniques. In

[11] *By 'more extreme', we mean that it leads to higher slippage: the spread is wider, the volatility is higher, the ratio of order size to average daily volume traded is higher and duration is shorter.*

[12] *For this we use the* `NearestNeighbors` *package in* Julia. *We re-estimate the tree object used for efficiently finding nearest neighbours after every 1,000 observations.*

[13] *We do this to ensure that all sampling probabilities are at least 2.5%, so that, if we use importance sampling to determine the slippage of a random decision rule that chooses each strategy with 25% probability, the likelihood ratios will never exceed 40. We want to avoid large importance ratios because they can result in slow convergence of our estimate of decision rule slippage.*

B. Alternate decision rule

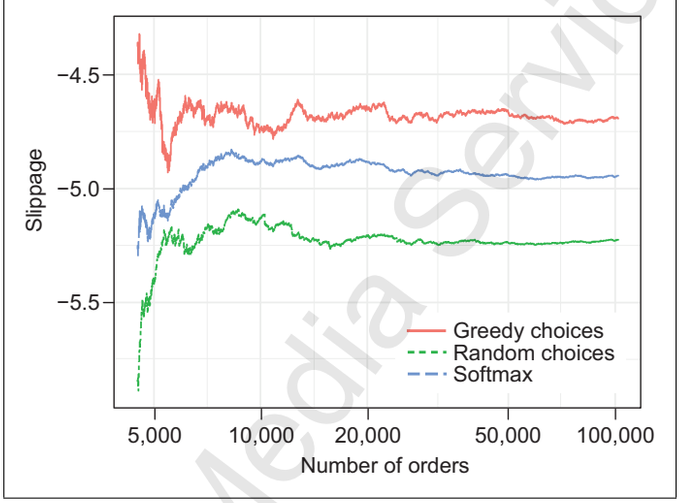
Decision rule	Greedy	Softmax	Random
k-NN	−5.1898 (0.0325)	−5.1921 (0.0137)	−5.2205 (0.0181)
Simple means	−5.0221 (0.0478)	−5.1651 (0.0183)	−5.2205 (0.0181)
Decision tree	−4.8334 (0.0362)	−4.9775 (0.0212)	−5.2205 (0.0181)
OLS	−4.6964 (0.0352)	−4.9414 (0.016)	−5.2205 (0.0181)

particular, in this case, the OLS method performs the best, and we managed to find this insight without needing to run this alternative decision rule in production.[14] We can additionally use importance sampling to recreate the scenario in figure 3 but with the greedy and softmax estimates of expected slippage reflecting the counterfactual case of using the OLS-driven decision rule. This is presented in figure 4. We can see that the OLS-informed greedy decision rule performs better relative to the random decision rule than we saw for the k-NN-driven greedy decision rule in figure 3.

Finding that in this case OLS outperforms k-NN is particularly useful, as it would be hard to arrive at this insight based purely on intuition. Both the OLS-driven and the k-NN-driven decision rules have intuitive appeal. The correct choice depends on which method chooses the best strategy most often, and the bias and variance of each method work against it choosing the best strategy. While OLS is generally a higher-bias method, we found that in this case it delivered better strategy recommendations than k-NN. Without using importance sampling, the only way to discover this insight would be to run both decision rules in parallel. This necessity would have placed a limit on how many decision rules could be trialled. (This use of importance sampling does not depend on the exact setting shown in this paper. In the appendix in the preprint version of this article (Baumann 2024), we show that if the exercise in the section titled 'Functional approximation and reinforcement learning to find a decision rule' is repeated with differing parameterisations, we get qualitatively similar results.)

[14] *OLS performs the best in this case since the true $S(x, a)$ used in generating the data is relatively smooth and without sharp changes. In other cases, a nonlinear method such as k-NN or a decision tree may be more appropriate. Thus, the take-away from this paper should not be that OLS performs well in making decision rules, but that importance-sampling methods can be used to find decision rules that perform well.*

4 Performance of counterfactual OLS decision rule



Conclusions

In conclusion, the advantages of importance sampling over A-B testing are significant. Importance sampling makes it possible to assess many decision rules without needing to start a new A-B test for every new idea. We can also use more data to assess the performance of each decision rule, which leads to greater statistical confidence in the results. Further, importance sampling is relatively easy to implement and does not require changes to the production code. ■

Stuart Baumann is a quantitative researcher who has worked on the buy side (statistical arbitrage in cash equities) and on the sell side (algorithmic trading of cash equities).
Email: Stuart@StuartBaumann.com.

REFERENCES

Baumann S, 2024
Choosing trading strategies in electronic execution using importance sampling
Preprint, SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5001783

Doroudi S, P Thomas and E Brunskill, 2018
Importance sampling for fair policy selection
Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18

Hastie T, R Tibshirani and J Friedman, 2009
The Elements of Statistical Learning: Data Mining, Inference and Prediction
Springer

Kolm P and G Ritter, 2019
Dynamic replication and hedging: a reinforcement learning approach
Journal of Financial Data Science 1(1), pages 159–171

Kolm P and G Ritter, 2020
Modern perspectives on reinforcement learning in finance
Journal of Machine Learning in Finance 1(1), pages 84–89

Metelli AM, M Papini, N Montali and M Restelli, 2020
Importance sampling techniques for policy optimization
Journal of Machine Learning Research 21, pages 1–75

Nevmyvaka Y, Y Feng and M Kearns, 2006
Reinforcement learning for optimized trade execution
Preprint, arXiv, https://arxiv.org/abs/2307.11685

Rashkovich V and A Verma, 2012
Trade cost: handicapping on PAR
Journal of Trading 7(4), pages 47–54

Ritter G, 2017
Machine Learning for Trading
Risk October, pages 84–89

Schnaubelt M, 2022
Deep reinforcement learning for the optimal placement of cryptocurrency limit orders
European Journal of Operational Research 296(3), pages 993–1,006

Sutton RS and AG Barto, 2018
Reinforcement Learning: An Introduction(2nd edition)
MIT Press