A Brief Introduction to Monte-Carlo Permutation Tests of Trading Systems

This document presents an intuitive description of how we might use a Monte-Carlo Permutation Test to evaluate the performance of a market trading system. It shows only one of many possible approaches, and this description does not address any of the pitfalls of the test. The purpose of this document is to serve as an easily understood introduction to this family of tests. It is not intended to be comprehensive or rigorous.

The scenario under which this particular test might be employed is as follows: We have developed a market trading system by experimenting on a historical dataset of market prices. Once satisfied with our obtained performance, we procure a new dataset which has not played a role in the development of the trading system. Typically, this new dataset begins at a date later than the last date in the dataset used to devise the trading system.

We choose a test statistic which will measure the performance of our trading system. This may be total return, mean return per trade, Sharpe ratio, Ulcer Index, profit factor, or any other statistic which we favor. We then apply our trading system to the new dataset and compute this performance statistic.

A naive experimenter would look at the computed performance figure and, if it is impressive, put the trading model to work. But there is an aspect of the performance that is every bit as important as its magnitude: the probability that a truly worthless system could have done as well by virtue of good luck. No matter how impressive the trading result, if a system that determines trades by flipping a coin has a significant probability of doing just as well, we would be crazy to bet money on the product of our endeavors. Good luck never lasts. If the ups and downs of the market can be predicted with our trading system, its continued good performance is likely. But if the sequential changes in the market do not follow a pattern that is predictable with our system, we will soon lose our investment.

Here is one way to handle this situation. Suppose we randomly permute the market changes in the test period and recompute the performance statistic. If this value is less than that obtained from the raw, unpermuted data, we are happy for this small bit of evidence that the trading system recognized predictable patterns in the market. But it's not very convincing evidence. If it were incapable of detecting these patterns, there would still be a 50-50 chance of observing this result. So we need to test more random permutations.

If we test nine random permutations, and the performance statistic of the original data exceeds all of them, we have more convincing evidence. In particular, if our system had no talent for capitalizing on patterns present only in real markets, there is a 1/10 chance that good luck would have placed it at the top. After all, in this situation, any of these ten orderings of the changes (one of them being the original order) has an equal shot at being the best.

What if the original performance statistic is the second best of the ten? There is a 2/10 probability that it will land in the best or second best slot. So, suppose we had decided in advance that if the original performance statistic is at least the second best, we would confidently conclude that our system honestly sees predictable patterns. If in truth it is worthless, we would have a twenty percent chance of being fooled by good luck.

Suppose we decide in advance to conclude that our system is worthy of trust if its performance statistic on the original data has at least a specified rank among all permutations. It should be apparent that there is a simple formula for computing the probability of this event under the scenario that the system is truly worthless.

Let *m* be the number of random permutations tested (not counting the original), let *k* be the number of these random permutations (again, not counting the original) whose performance statistic equals or exceeds that of the original. Then, the probability that the original performance statistic will achieve this exalted position or better is $(k+1) / (m+1)$. You can understand this formula if you visualize the $m+1$ statistics (original plus *m* permuted) lined up in order. Note that the original statistic has equal probability of occupying any of these $m+1$ slots.

A traditional statistical test of the null hypothesis that our trading system is unable to capitalize on real market data, versus the alternative that it is able to do so, would be performed as follows: Decide in advance what level of error probability you are willing to accept. This error, often called the *alpha level*, is an upper bound for the probability that you will erroneously reject the null hypothesis. Here, this error is concluding that our system has the power to predict market moves when in fact it doesn't. Choose a large value of *m*, and compute *k* from the formula above. Then perform the random replications and count how many of them have a performance statistic that equals or exceeds that of the original data. If *k* of them or fewer do so, we can reject the null hypothesis. If the null hypothesis is true (the system is worthless), we will make this erroneous rejection with probability at most our specified alpha.

This permutation test is not perfect. For example, serial correlation in the market data can have an adverse impact on the test's reliability, although much empirical evidence indicates that the impact is minimal. Other weaknesses are possible. The current state of the art in Monte-Carlo Permutation Tests of market trading systems is not very advanced. Hopefully, some talented graduate students will take up the challenge and base their dissertations on this subject.

There are several other ways in which a Monte-Carlo Permutation Test can be used to evaluate a market trading system. The method described in this document assumes that we have enough control over the experimental process that we can re-run the trading algorithm on a large number of permutations of the market changes. In real life we will often not be this fortunate. We may have available only a single set of trades, perhaps given to us by an outsider. In this case, we must keep these trades fixed in time and permute the market around them. This is a relatively easy procedure, but it has numerous pitfalls. This method, along with its advantages and disadvantages, is described in considerable detail in the accompanying document *Monte-Carlo Evaluation of Trading Systems.*

We can avoid the need for holding out a virgin test period of data by repeatedly permuting the market changes and applying the training algorithm to each permutation. This is wonderful when we can do it, as we enjoy the benefit of being able to use the entire available history for both training and testing! But because the entire training process must be repeated for each permutation, this method requires that the training algorithm be both automated and fast. That's a lot to ask.

For those situations in which the predictions are numeric, and we base trade decisions on where the prediction lies relative to one or more fixed thresholds, we can use a very simple version of this test to shuffle data around the threshold(s).

Finally, there are Monte-Carlo Permutation Tests that are applicable to situations in which we generate several competing models and choose the best from among them as our final model. These tests automatically account for the selection bias inherent in choosing the best.

All of these variations are described in *Monte-Carlo Evaluation of Trading Systems*, available elsewhere on this site.