A high performance pair trading application

Conference Paper · May 2009

DOI: 10.1109/IPDPS.2009.5161147 · Source: DBLP

Camilo Rostoker
University of British Columbia - Vancouver
9 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Alan Wagner
University of British Columbia - Vancouver
56 PUBLICATIONS **469** CITATIONS
SEE PROFILE

A High Performance Pair Trading Application

Jieren Wang
Department of Mathematics
University of British Columbia
Vancouver, British Columbia
cwang@math.ubc.ca

Camilo Rostoker and Alan Wagner
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
{rostokec,wagner}@cs.ubc.ca

***Abstract*— This paper describes a high-frequency pair trading strategy that exploits the power of MarketMiner, a high-performance analytics platform that enables a real-time, market-wide search for short-term correlation breakdowns across multiple markets and asset classes. The main theme of this paper is to discuss the computational requirements of model formulation and back-testing, and how a scalable solution built using a modular, MPI-based infrastructure can assist quantitative model and strategy developers by increasing the scale of their experiments or decreasing the time it takes to thoroughly test different parameters. We describe our work to date which is the design of a canonical pair trading algorithm, illustrating how fast and efficient backtesting can be performed using MarketMiner. Preliminary results are given based on a small set of stocks, parameter sets and correlation measures.**

I. Introduction

Pair trading is a popular quantitative method of statistical arbitrage that has been widely used in the financial industry for over twenty years [1]. The essence of pairs trading is to exploit pairs of stocks whose co-movement are related to each other. When the co-movement deteriorates, the strategy is to buy the under-performer and sell the over-performer, anticipating that the co-movement will recover and gains can be made. If the co-movement does recover, the positions are reversed yielding arbitrage profits from the spread of the two stock prices.

In the last few years the trading industry has seen an explosive growth in low-latency networks and infrastructure. While this has enhanced many aspects of the trading lifecyle, it has also led to an increase in data volume and frequency, posing additional challenges for processing and analyzing data in the context of designing and backtesting trading strategies. Software platforms that assist in the process of data acquisition and management, quantitative analysis, backtesting and deployment are broadly referred to as *alpha generation* platforms [2]. There are a number of sophisticated alpha generation platforms in the market [3], [4], [5], with one notable open-source project [6], but the problem with all these solutions is that they are not inherently parallel and thus their ability to scale across a large number of securities or over a large amount of data, particularly intra-day tick data, is limited. Financial engineering practitioners and researchers have begun to address this problem by parallelizing computationally intensive aspects of the analysis pipeline such as monte carlo for options pricing [7], [8] or by developing more generic analysis frameworks [9]. Given the current meltdown in the financial markets, we should expect the next generation of models and strategies to be faster, smarter, and have the ability to take into account market-wide dependencies.

This paper will describe how a canonical intra-day pure statistical pair trading strategy can be quickly and efficiently backtested using the MarketMiner analytics platform [10]. Since we are backtesting a strategy based on correlation, we wanted to compare various correlation measures to determine which one performs better and under what circumstances. To eliminate the potential bias of selecting specific pairs, our goal is to take a brute-force approach by backtesting over as many pairs as possible to determine the relative performance of the strategy under different correlation measures. We'll elaborate the challenges we've encountered while designing and backtesting the strategy in Matlab, even when the correlation matrices had been computed beforehand, and then describe how we plan to integrate the strategy directly into MarketMiner in order to speed up the backtesting process by overcoming the main bottleneck, the computation of all pair-wise correlations. Section II provides an overview of the MarketMiner system, followed by Section III which provides details on our canonical pair trading strategy. Sections IV describes the usual backtesting process, and how ours differs given we want to analyze more data and more pairs than ever before, while Section V will provide some preliminary performance results. Section VI concludes the paper and suggests interesting avenues for future work.

II. Overview

Pair trading can be broadly categorized into three forms: fundamental, statistical, and risk. A *fundamental pair* is a pair that has been highly correlated over a historical period, usually a few years or more, and often belong to the same industry or sector. A few well-known pairs are Exxon/Chevron, UPS/Fedex and Wal-Mart/Target. A *risk pair* occurs when a company is about to merge with or acquire another one, and thus the two securities will become highly correlated in anticipation of the adjusted price levels. A *statistical pair* refers to a pair that may or may not be fundamental linked, but have been found to be highly correlated over a given historical period, with a high degree of statistical certainty. Intra-day statistical pairs trading is high-turnover strategy that uses only very recent data to determine correlations (e.g., the last few hours or days at most). This means that the strategy makes no assumptions that a pair will remain correlated next year or month, but has a certain level of confidence that the pair will remain correlated for the next few hours or days. Generally speaking, we would expect the correlations to remain stable for approximately the same amount of time used in the correlation calculation.

The usual routine for a fundamental pair trader is to first identify a number of candidate pairs. Each pair is then back-tested over a given set of data and parameter sets before being promoted to a live trading environment. The exact method used to identify and backtest pairs differs from trader to trader. Some traders may employ a rigorous statistical analysis, while others simply "eye-ball" two charts to determine the degree of correlation. In live trading, the number of pairs monitored per trader can range from a few to a thousand or more; once the number of pairs exceeds what a human can watch, software for monitoring the pairs must be utilized.

In today's fast-paced trading environments, it is increasingly true that to out-compete the competition, we must out-compute them [11]. The explosive trend toward automated trading and the availability of tick data at sub-millisecond rates introduces new demands and opportunities which require quick online analysis and decision processing. MarketMiner is an ongoing research project that addresses this data analysis problem by supporting the computational workload associated with performing market-wide backtesting of trading strategies. The original design of MarketMiner was a basic MPI-enabled pipeline for processing quote data [12], and has since been extended to support arbitrary directed acyclic graph (DAG) stream processing workflows. One of the strengths of MPI is that it is the de-facto standard for messaging-passing parallel programming and there are a large number of high quality open-source numerical libraries available.

Given the requirements of a pair trading strategy, the enabling feature of MarketMiner is its ability to handle a large amount of market-wide, high frequency "tick" data from a live feed or from a historical database, and use this data to produce large correlation matrices in an online fashion. One obvious challenge in working with high-frequency data is due to its sheer volume - a single day's worth of uncompressed Trade and Quote (TAQ) data [1] typically consumes over 50 gigabytes of disk space! While research using high-frequency data appears to be gaining momentum, the ability to incorporate the data in a market-wide backtesting context has been limited due to the lack of a scalable solution for processing and analyzing such data.

It is well-known that the quality of high-frequency real-time stock quote data is low and difficult to use in measuring correlation. Due to its frequent nature it may contain a large proportion of transmission or human errors. Indeed, traditional correlation measures are quite sensitive to outliers and this presents a major challenge. Traditionally, traders use a variety of data filters to filter out the 'bad' data, and then use the standard Pearson definition to find correlation. This way correlation is somewhat more robust and reliable. However, this approach still has potential bias due to choice of filter. The MarketMiner system has the ability to use a robust correlation measure, Maronna correlation, which is much less sensitive to outliers and smooths the underlying timeseries used for computing correlations [13]. Despite these advantages, the robust method is computationally expensive and thus not commonly used in statistical software packages, especially those that operate on real-time data. The MarketMiner system overcomes this difficulty by implementing a parallel algorithm for computing robust correlation matrices [14]. The original work investigated its scalability as an offline algorithm, and more recently in an online setting [12].

[1]TAQ data is a consolidated dataset of all equity trades and quotes from the NYSE, NASDAQ and AMEX.

III. A Canonical Pair Trading Strategy

Our high-frequency pair trading strategy exploits the power MarketMiner to perform a real-time search for short-term correlation divergences. Unlike other pair trading approaches described in the literature [15], [16], [17], [18], we are able to take a brute-force approach looking over all possible pairs and combinations of parameters. Table I describes the strategy parameters and typical values we use within our experimental framework. Where these parameters arise in our pairs trading algorithm is discussed below. All time-based parameters are in time units, defined by the time window Δ_s and indexed by $s = \{0, \ldots, s_{max}\}$, where s_{max} defines the total number of Δ_s intervals in the analysis. For example, there are exactly 23400 seconds in a typical trading day, and if $\Delta_s = 30$ seconds, then there will be $s_{max} = \frac{23400}{30} = 780$ intervals. We let K denote the set of parameter sets under consideration, and use k to index a particular parameter set. Thus, for instance $\{\Delta_s = 30, C_{type} = Pearson, A = 0.1, M = 100, W = 60, Y = 10, d = 0.01, \ell = 2/3, RT = 60, HP = 30, ST = 20\}$ is one element of the set K. Each unique combination of parameters gives rise to a unique pair trading strategy. Using the MarketMiner system we are able to backtest a trading strategy for each pair $p \in \Phi$, with Φ denoting the set of all pairs under consideration, and for each parameter vector $k \in K$ over the given time period. In our high-frequency analysis we use the bid-ask midpoint (BAM) as an approximation to the stock price, and from that calculate the 1-period return. The *bid price* is the highest price someone is willing to pay for a stock, and the *ask price* is the lowest price someone is willing to sell a stock. We choose to use the BAM instead of just the actual price as it allows for a closer approximation to the actual price level between trades (e.g., as opposed to using regression), which is especially useful for stocks which trade infrequently. Quote data is much higher in frequency and volume than trade data, which makes processing and analyzing the data more challenging, and thus a particularly well-suited problem for an HPC solution. A small sample of intra-day quote data is shown in Table II. Raw tick TAQ data contains every raw quote, not just the best offer, so there can be many spurious ticks originating from various sources, some human typing errors but mainly from electronic trading systems generating test quotes (e.g., when testing a new feature) or far-out limit orders which have little probability of getting filled. Raw data, whether from a database or a live stream, needs to be cleaned before being analyzed and used in

Parameter	Description	Values					
Δ_s	Time window	30 sec					
C_{type}	Type of correlation measure	Pearson		Maronna		Combined	
A	Minimum correlation for trading	0.1					
M	Time window for correlation calculation	50		100		200	
W	Time window of average correlation calculation	60			120		
Y	Time window over which divergences from the correlation average are considered	10		20		40	
d	Divergence level from correlation average required to trigger a trade	0.01%	0.02%	0.03%	0.04%	0.05%	0.10%
ℓ	Retracement level for determining when to reverse a position	1/3					
RT	Time window for measuring the spread level (used in calculating retracement level)	60					
HP	Maximum holding period for any position	30					
ST	Minimum time before market close required to open a new position	20					

TABLE I

STRATEGY PARAMETER DESCRIPTIONS AND VALUES

Timestamp	Symbol	Bid Price	Ask Price	Bid Size	Ask Size
09:30:04	NVDA	16.38	20.1	3	3
09:30:04	NVDA	18.23	18.26	3	3
09:30:04	NVDA	18.24	18.26	1	4
09:30:04	ORCL	19.56	19.59	2	104
09:30:04	ORCL	19.58	19.62	1	1
09:30:04	SLB	82.81	83.11	1	1
09:30:04	TWX	14.01	14.2	18	5
09:30:04	TWX	14.01	14.65	2	6
09:30:04	BK	41.11	42.1	41	1
09:30:04	BK	41.13	41.5	1	1
09:30:04	BK	41.11	42.1	38	1
09:30:04	BK	41.13	41.5	3	1

TABLE II

SAMPLE DATA FROM THE NYSE TAQ DATASET.

a financial model or strategy. There are many techniques used in practice to clean tick data [19], [20], each having its own advantages and disadvantages. The exact method of cleaning will vary depending on the particular task at hand, and trade-offs between the quality of cleaning and delay need to be managed; e.g., in a real-time environment cleaning process needs to be fast and efficient. Our approach is to use a very simple but effective TCP-like filter to eliminate prices that are more than a few standard deviations from their corresponding moving average and deviation. The remaining outliers will be gracefully down-weighted by the robust correlation method implemented in MarketMiner.

The enabling aspect of this market-wide strategy is the ability to quickly compute a large correlation matrix using a sliding window of recent data points. The input to each pair-wise correlation calculation at time s are two vectors $X_i(s)$ and $X_j(s)$, containing the last M log-returns for stocks i and j respectively. Each element $x_i \in X_i(s)$ is defined as $x_i = log(r_i(s))$, where $log(\cdot)$ is the natural logarithm operator, and $r_i(s) = \frac{P_i(s)}{P_i(s-1)})$ is the 1-period return with $P_i(s)$ and $P_j(s)$ the prices of stocks i and j at time s. The reason for using log-returns instead of the raw prices is twofold: taking the difference of the returns yields a stationary process, while taking the log of the differences results in a (log) normal distribution; both results are necessary in order to utilize statistics which assume stationarity and normality.

The following pseudo-code outlines a canonical statistical pair trading strategy, defined by the particular set of stocks Φ and parameter set $k \in K$ over a given trading day t. We want to choose Φ as the full set of stocks which may potentially be chosen for backtesting, so as to optimize the strategy to perform well under that set of stocks. If there are n stocks then $|\Phi| = \frac{n(n-1)}{2}$. If our goal was to backtest over all US stocks, of which there are approximately 8000, this would require our strategy to support backtesting on over 32 million pairs! While many stocks are not liquid enough (too few trades) to be considered in our style of pair trading, the number of potential pairs is still so large that a parallel algorithm is essential for real-time trading.

1) At time s, calculate the average correlation over the last W time intervals as
$$\bar{C}_{i,j}(s) = \frac{\sum_{\sigma=s-W+1}^{s} C_{i,j}(\sigma)}{W},$$
where $C_{i,j}(\sigma)$ is the correlation coefficient calculated using the log-return vectors $X_i(\sigma)$ and $X_j(\sigma)$.
2) Check to see if $\bar{C}_{i,j}(s)$ is greater than threshold A, and the current correlation coefficient at time s has diverged more than d% from $\bar{C}_{i,j}(s)$ within the last Y time intervals. We refer to d as the divergence threshold. Note that typical divergence levels for pair traders with longer time horizons tend to be larger, due to the fact that the volatility will also be greater. With our intra-day strategy we use a smaller divergence level to account for lower volatility.
3) If no divergence is detected or $\bar{C}_{i,j}(s) \leq A$, move on to the next pair. If a divergence is detected, trigger a pair trade. Go long on the stock that has "under-performed" and short the one which has "over-performed". The over-performer is simply the one which has a higher W-period return relative to the other.
4) To choose a long/short ratio, we choose a ratio that keeps us as close to cash-neutral as possible, but just slightly on the long side. For example, if we are buying MSFT at \$30 and selling IBM at \$130, a ratio of 5:1 would give us an allocation of \$150 long and \$130 short. To be more specific, suppose we have two prices $P_i > P_j$, and we want to long stock i and short j, then we want the ratio of long/short shares for stocks i and j to be 1:x, where
$$x = \lfloor \frac{P_i}{P_j} \rfloor$$

Similarly, if we short i, and long j, then

$$x = \lceil \tfrac{P_i}{P_j} \rceil$$

5) The next step is to decide when to reverse the positions. We reverse the position when we have reached a retracement level L, or if a given amount of time has elapsed since we entered the position. The *retracement level* is calculated in the following way. Let S_l, S_h and $\bar{S}$ be the high, low and average of the spread during the last M time intervals, and S_e be the spread of the two stock prices at the time we opened the position. If $S_e \leq \bar{S}$, then

$$L = S_l + \ell(S_h - S_l),$$

and if $S_e \geq \bar{S}$, then

$$L = S_h - \ell(S_h - S_l)$$

where $1 > \ell > 0$ is the retracement parameter. For example, if the high of a MSFT-IBM spread is \$100, and the low \$80, and we opened the position when the spread was around \$80, and $\ell = \frac{1}{3}$, then we reverse when the spread has reached the retracement level

$$L = \$80 + \tfrac{1}{3}(\$100 - \$80) = \$80 + \tfrac{1}{3}\$20 = \$86.67.$$

Similarly, if we opened the position when the spread was around \$100, then

$$L = \$100 - \tfrac{1}{3}(\$100 - \$80) = \$100 - \tfrac{1}{3}\$20 = \$93.40$$

and we will reverse the position when the spread is lower than L. We also need to add a time-based reversal trigger in case the retracement level is never reached. Therefore, we choose not to hold a position longer than HP time periods. Thus after HP time periods the position is reversed, regardless of the situation. Finally, we should reverse all positions at the end of the trading day. We note here that the key to a good strategy is to mitigate losses and control risk. Thus, we point out, but do not consider any further, several other reversal conditions. The first is an absolute stop-loss: If the spread continues to drop rapidly, we want to exit and minimize our loss. The second is correlation reversion: If the correlation returns within the average range (i.e., $[\bar{C}(1-d), \bar{C})$, then we reverse the positions. The reasoning behind correlation reversion is that the prices may have adjusted to new levels and watching for spread reversion may not give us this information.

6) Once the position is reversed, we calculate the return $R_{i,j}$ for pair of stocks over both the long and short positions:

$$R_{i,j} = \tfrac{\pi_{i,j}}{P_i N_i + P_j N_j}$$

where $\pi_{i,j}$ is the profit/loss of the trade (in dollars), P_i and P_j are the prices and N_i and N_j the number of shares held for stock i and j respectively. For example, suppose a trade was to long MSFT at \$30 and short IBM at \$130 with the ratio of MSFT to IBM 5:1. If when we reverse the position MSFT is \$29 and IBM is \$120, then we profit $(\$29 - \$30)5 + (\$120 - \$130(-1)) = \$5$ from this trade. The total cost, not including transaction costs, is $5(\$30) + 1(\$130) = \$280$, and thus the return is $\$5/\$180 = 2.8\%$.

IV. Backtesting of Trading Strategies

The next natural question is to ask which configuration of parameters results in the best performance. One way to compare them is to test on historical data and measure the performance of each. This procedure is called by *backtesting*. Backtesting a pair trading strategy on a particular pair of stocks involves choosing a suitable set of historical data H, running the strategy on H and noting wins and losses of each trade and computing some measure of performance, such as cumulative returns. For comparison, one can do backtesting on alternative configurations of a given pair trading strategy on the same data H and compare the relative performance results. This basic procedure can be done across a variety of strategies, pairs, sets of historical data and performance measures to help identify the best overall trading strategy. In our experiments we focused on testing the performance of trading strategies where the major difference was in the method of correlation.

The raw data used in the experiments are TAQ bid-ask data for 61 highly liquid US stocks frequently traded by professional pair traders. Since we examine all pairs for a given set of stocks, the results presented here are based on $\binom{61}{2} = 1830$ pairs. Our strategy works on high-frequency time frames, and thus the total dataset we consider here is limited to one month (March 2008) which consists of 20 trading days. While designing our market-wide pair trading strategy we performed some preliminary experiments using Matlab to get a feel for the different parameters and range of values they would take. These values are given in Table I. While this approach worked reasonably well for a small dataset of 61 stocks, we are aware that this solution will not scale. In the following paragraphs we briefly describe our initial experiments using Matlab and how the need for scalability motivated us to consider a design more tightly integrated into MarketMiner.

Approach 1: Using Matlab to read in MarketMiner's pre-computed correlation matrices. In this approach we turn sets of pre-computed correlation matrices $\Omega^t(s)$ generated by MarketMiner into 1830 time series $\{C_p^t(s) : M \leq s \leq s_{\max}\}$ for each trading day t, by picking out the relevant entry of each correlation matrix. We soon abandoned this approach as we were unable to read in multiple matrices due to memory constraints. Each matrix $\Omega^t(s)$ is 61×61, and when we use $\Delta_s = 30$ and $M = 100$, we need read in 680 such matrices in order to define a particular time series $\{C_p^t(s) : M \leq s \leq s_{\max}\}$, and that is for just one day t out of 20 under consideration!

Approach 2: Using Matlab to re-create the correlation timeseries, not utilizing MarketMiner correlation matrices. In this approach we did not use MarketMiner's correlation matrices, but rather re-created all correlation timeseries in Matlab. The caveat here is that calculating the Maronna correlation coeffficients independently no longer assures the resulting matrix is positive semi-definite (PSD). Nonetheless, using Matlab to generate our correlations directly proved to be more efficient than reading the pre-computed matrices into Matlab. We were able to produce a daily return vector $R_p^{t,k}$ for a given pair p, day t and parameter vector k in approximately 2 seconds, depending on the specific pair and parameters, using an Open SUSE Linux PC with a dual core Intel Pentium 4 2.80 GHz processor. Even when using a Sun Grid Engine scheduler (SGE) to distribute jobs, it is clear the computations are prohibitively slow. With the need to produce 1830 (number of pairs) $\cdot$ 20 (number of business days in March, 2008) $\cdot$ 42 (number of parameter sets) daily return vectors to track returns over a given month, a rough estimate for the computation time on a single computer is 854 hours. Using this same scenario but backtesting over a year would take about 445 days, and even worse, scaling up to 1000 pairs over just one month would take an estimated 19425 days, or 53 years! We were able to reduce the computation time by creating scripts which sent out independent Matlab jobs to a Sun Grid Engine scheduler. This solution still has problems as the matrices are still not PSD, and more importantly does not allow for a tight interaction between independent pairs throughout the course of a trading day, which can be used to optimize certain aspects of the strategy.

Approach 3: The integrated MarketMiner solution. Given the challenging task of analyzing market-wide correlation matrices, it seems apparent that a custom implementation integrated directly with the MarketMiner platform is necessary to achieve the desired scale and timing objectives. Figure 1 illustrates a potential pair trading system using MarketMiner to power a pair trading strategy with a particular set of parameters. The advantage of a tight integration with MarketMiner is that the outputs from each strategy (trades decision) can be gathered by a master process to perform additional tasks such as risk management and liquidity provisioning. Also, aggregating the results into a single basket, as opposed to many individual trade orders, allows the trading system to send utilize a sophisticated list-based algorithm to optimize the actual execution of the trades.

Evaluating a Trading Strategy

The approach in which an intraday strategy is evaluated differs from strategies which make trades only occasionally (e.g., every few days or even just once a month, in the case of a pension or mutual fund). Since we have many trades each day, we want to evaluate how the strategy performs within the day, but also over multiple days through a given period of time. We adapt some of the trading model evaluation measurements from the high frequency finance literature [21]. In a given trading day t, for each for pair p and parameter vector k, a set $R_p^{t,k}$ of returns is generated. Therefore the total set of returns for the trading periods is just the union of each days returns:

$$R_p^k = \bigcup_{t=1}^{T} R_p^{t,k}. \tag{1}$$

The following analysis uses three key performance metrics commonly used to assess the performance of a trading strategy: cumulative returns, maximum draw-down and win-loss ratio. These performance measures can be defined either over a given pair p and parameter set k, or summarized over all pairs or over all parameter sets. Each of the three variants provides a different view of the results. For example summarizing the results over all pairs but for a given parameter set indicates which parameters are most effective, while summarizing over all parameter sets but with a given pair indicates that the pair may be a particular good candidate for pair trading and less sensitive to choice of parameters. The formulas for each of the three performance measures is given below.

1) *Cumulative Returns*: Cumulative returns measures the equity growth of a particular strategy. This measure is appropriate when we assume that the strategy always reinvests the total available capital at the start of each period. The *daily cumulative return* for pair p and parameter vector k on day t is defined as

$$r_p^{t,k} = \prod_{q=1}^{|R_p^{t,k}|} (r_{p,q}^{t,k} + 1) - 1 \tag{2}$$

where $r_{p,q}^{t,k}$ is the qth return on day t. The *total cumulative return* r_p^k over the entire trading period, again for pair p and parameter set k, is calculated as

$$r_p^k = \prod_{t=1}^{T} (r_p^{t,k} + 1) - 1. \tag{3}$$

Both the daily and total cumulative returns can be further summarized by aggregating the returns over all pairs using a given parameter set, or over all parameter sets but for a particular pair. These measures can be used to test the effects of pairs choice and parameter choice on returns of trading strategies, and thus help with refining trading strategies. For example, the total cumulative return over all pairs on day t using parameter set k is

$$r^{t,k} = \prod_{p\in\Phi} (r_p^{t,k} + 1) - 1 \tag{4}$$

and similarly, the total cumulative return for pair p on day t over all parameter sets is

$$r_p^t = \prod_{k\in K} (r_p^{t,k} + 1) - 1. \tag{5}$$

The same summary calculations can be applied to daily cumulative returns.

2) *Maximum Drawdown*: Maximum drawdown is a measure of the riskiness of a trading strategy. It can be interpreted as the "worst peak to valley drop", for the pair p:

$$MDD_p = \max_{k\in K} (r_{p,q_a}^k - r_{p,q_b}^k : q_a, q_b \in R_p^k, q_a \leq q_b), \tag{6}$$

where r_{p,q_a}^k and r_{p,q_b}^k are the total returns for pair p using parameter set k from trade number 1 to q_a and q_b, respectively. Note that we could also define maximum drawdown MDD^k for a given parameter set k. Moreover, we can define the two variants of maximum

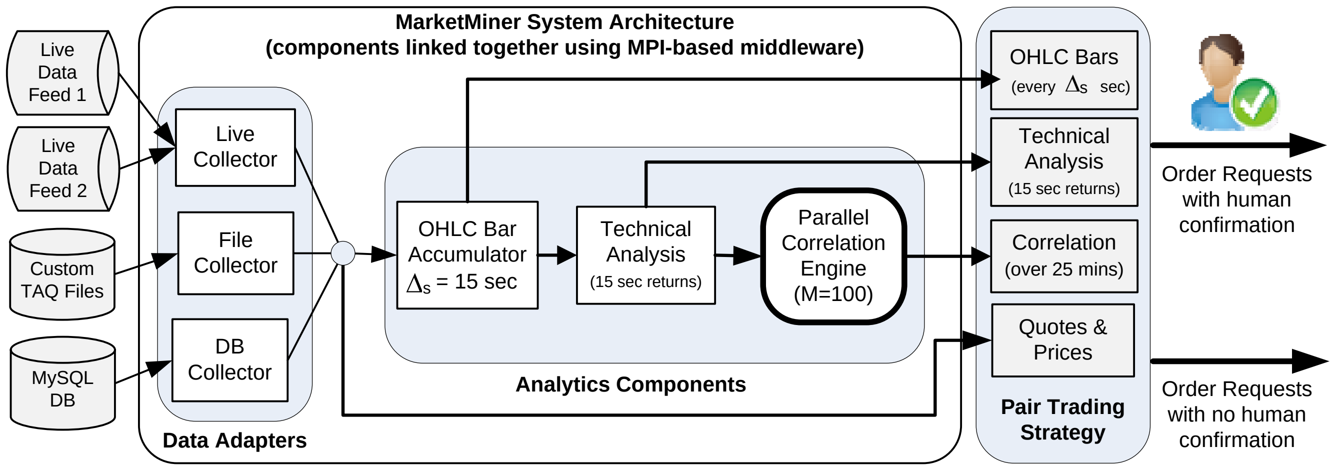


Fig. 1. MarketMiner enabling rapid backtesting or live execution of a pair trading strategy

drawdown on a daily basis for pair p and parameter set k, which is:

$$MDD_p^k = \max(r_{p,t_a}^k - r_{p,t_b}^k : t_a, t_b \in T, t_a \leq t_b). \quad (7)$$

3) *Win-Loss Ratio*: The win over loss trades ratio provides information on the type of strategy used by the model. Its definition is

$$\frac{W_p^k}{L_p^k} = \frac{|\{r_{p,q}^k : r_{p,q}^k > 0, q \in R_p^k\}|}{|\{r_{p,q}^k : r_{p,q}^k < 0, q \in R_p^k\}|}, \quad (8)$$

where $\{r_{p,q}^k : r_{p,q}^k > 0)\}$ is a set of trades that gives positive returns, and $\{r_{p,q}^k : r_{p,q}^k < 0)\}$ is the set of trades with negative returns. The numerator corresponds to the number of winning trades and the denominator is the number of losing trades over the same period. If we are interested in the difference of the performance of the strategies with different parameters values, we can use

$$\frac{W^k}{L^k} = \frac{|\{r_{p,q}^k : r_{p,q}^k > 0, p \in \Phi, q \in R_p^k\}|}{|\{r_{p,q}^k : r_{p,q}^k < 0, p \in \Phi, q \in R_p^k\}|}, \quad (9)$$

where again Φ denotes the set of all pairs under consideration.

V. Results

The results presented here focus on some preliminary performance data from trading 61 stocks using our Matlab implementation. The purpose is to demonstrate how a wide range of parameters can be backtested to find configurations that result in different strategies which can be matched to particular risk profiles.

Strategy Performance Results

Performance comparisons of two different trading strategies can be done across several dimension: the type of correlation measure used, the choices of parameters and pairs, etc. We focus attention on differences in performance arising from different choices of correlation type. With a large set of returns data and their corresponding performance measures we may ask whether this information can help to shed some light on which strategies are more effective - those using Pearson, Maronna or Combined correlation. We analyze three performance measures - cumulative monthly return as defined in Equation (3), maximum daily draw down (7), and the win-loss ratio (9) - and aggregate the data by taking an average over different parameter sets.

Here are the specific details for our analysis. We may consider Pearson, Maronna and Combined correlations as our *treatments*, which are applied to our 1830 pairs of stocks, with other *factors* (not considered part of our treatment) consisting of the remaining elements in our parameter sets: $\{\Delta_s, A, M, W, Y, d, \ell, RT, HP, ST\}$. We run the experiments on different levels of these factors to account for bias of choosing any one level. Each pair of stocks receives each treatment at each level of the remaining factors.

The response from each treatment is one of our three performance measures - cumulative monthly return, maximum daily draw down, and win-loss ratio. We discuss in detail the case of cumulative monthly returns, but the other cases are similar. Recall our notation that r_p^k is the total cumulative return of pair p using parameter vector k over the period of one month. To highlight the fact that there are three treatments we let $r_p^{C_{type},k'}$ denote the return with a specified correlation type C_{type} with $k' \in K'$ representing the 14 different parameter vectors of the form $\{\Delta_s, M, W, d, \ell, RT, HP, ST, Y\}$. Thus there are 14 levels of non-treatment factors, and each pair has a response $r_p^{C_{type},k'}$ for each of these levels. Our approach is to average these responses over the different factor levels to get a single estimate of the performance of pair p using correlation type C_{type}. Thus, the sample observations from our populations are *average* cumulative returns over the month:

$$\bar{r}_p^{C_{type}} = \frac{\sum_{k' \in K'} r_p^{C_{type},k'}}{|K'|} + 1$$

where the average is over the set of alternate parameter vectors K'. We see that $\bar{r}_p^{C_{type}}$ is a measure of returns for pair p when using C_{type} as the type of correlation. We define average maximum daily drawdown and win-loss ratio for each pair of stocks and each correlation measure analogously, again where the average is over the 14 different levels of the non-treatment factors $\{\Delta_s, A, M, W, Y, d, \ell, RT, HP, ST\}$. Tables III, IV and V contain descriptive statistics for each performance measure with respect to the different correlation types. The "best" value for each measurement is shown in bold. In Table III we also show the Sharpe ratio, which is a measure of *risk-adjusted return* and defined as

$$S_R = \frac{\bar{r}}{\sqrt{\hat{\sigma}^2}}$$

where $\bar{r}$ is the average return and $\hat{\sigma}^2$ is the variance of the return around its mean.

	Correlation type: C_{type}		
	Maronna	**Pearson**	**Combined**
Mean	1.1473	**1.1521**	1.1098
Median	1.1204	**1.1278**	1.0979
Standard Deviation	0.1235	0.1085	**0.0747**
Sharpe Ratio	9.2899	10.6184	**14.8568**
Skewness	**2.8484**	1.9281	1.4871
Kurtosis	16.6541	9.4091	**7.1706**

TABLE III

AVERAGE CUMULATIVE MONTHLY RETURNS

	Correlation type: C_{type}		
	Maronna	**Pearson**	**Combined**
Mean	1.6662%	**1.5433%**	1.5666%
Median	1.2446%	**1.1533%**	1.1702%
Standard Deviation	1.5481	**1.4606**	1.4668
Skewness	**3.4443**	3.5005	3.889
Kurtosis	21.5922	**21.5295**	27.3131

TABLE IV

AVERAGE MAXIMUM DAILY DRAWDOWN

	Correlation type: C_{type}		
	Maronna	**Pearson**	**Combined**
Mean	1.2697	1.2724	**1.2787**
Median	1.2652	1.2688	**1.2689**
Standard Deviation	**0.1263**	0.1269	0.1356
Skewness	0.2897	0.2521	**0.3002**
Kurtosis	3.0781	**3.0665**	3.0991

TABLE V

AVERAGE WIN-LOSS RATIO

In addition to these tables, box plots are included to give a qualitative appreciation of the data. See Figure 2(c). On each box, the central mark is the median of the distribution, the edges of the box are the 25th and 75th percentiles (or first and third quartiles), the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. From these plots we see that the distributions contain a significant number of outliers with some abnormally high values.

The data presented here leads to several interesting observations. For simplicity, we will call any trading strategies using correlation type C_{type} a C_{type} strategy. First, Pearson strategies have higher mean cumulative returns than Maronna and Combined, but it also has higher standard deviation than Combined. Therefore, the Sharpe ratio of Combined strategies are much higher than both Maronna and Pearson.

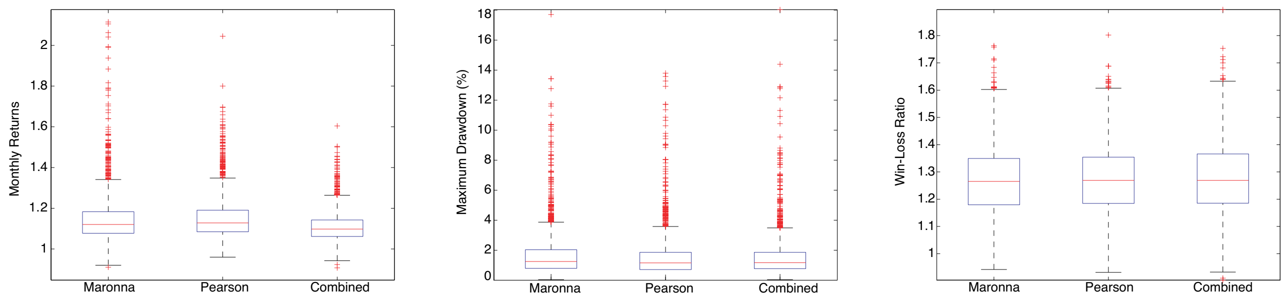
The average maximum draw down of Pearson strategies are lower than both of Maronna and Combined. This means if we use the automated trading strategies using the three correlation measures to trade for a month, Pearson strategies will have the least average "worst-peak-to-valley" drop. Indeed, Combined and Pearson strategies have the same average absolute "peak-to-valley" drop value, but the Pearson has higher peak return value which results in less of a maximum drop down. Since we would prefer to have a low maximum draw down it appears that Maronna strategies are a less favorable choice according to this performance measure.

As for win-loss ratios, we find that the results for all three strategies are fairly similar, with Combined strategies having a small advantage in both in terms of mean and standard deviation. Related to the fact that the Combined strategies make much less cumulative returns over the month on average, this suggests that Combined correlation is more conservative but generates lower returns, whereas Pearson can generate higher returns, but bears more risk. Maronna strategies are somewhere in between.

Also included are measures of skewness and kurtosis, which are the third and fourth moments of a distribution respectively. Generally speaking, skewness measures the lack of symmetry of a distribution around its mean. A better strategy would have a higher degree of skew to the right (large positive skewness statistic), which means there are more proportion of samples that are greater than the mean than those less than the mean. Kurtosis measures the degree to which a distribution is more or less "peaked" than a normal distribution. A greater kurtosis means there is a relatively greater probability of an observed value being either close to the mean or far from the mean. From the summary data in the tables we find that the cumulative returns of Maronna trading strategies are more skewed to the right and has fatter tails than the others, which suggests more trades yield unusually high returns, compared to say Pearson trading strategies. This can also be clearly seen in the box plots. This suggests that Maronna strategies yield high returns for select pairs. Identifying which pairs perform well is worthy a further investigation.

We should also note that the mean is affected by the skewness of the distribution, and so we also give a robust estimate of central tendency, the median, which is less affected by lack of symmetry of the population distribution. Notice that by a simple comparison of medians we draw identical conclusions to when we used means.

It is important to stress that all of these simple comparisons between values in the tables need to be examined on a more rigorous standard of *statistical significance* in order to be truly meaningful. To do so we may consider a few simple inferential statistical tests. We discuss the ideas of a basic scheme for designing tests on cumulative monthly returns as one example of the type of analysis we are interested in, and the other performance measures can be analyzed in a similar fashion. To be clear about the underlying statistical



(a) Average cumulative monthly returns (b) Average maximum daily drawdown (c) Average win-loss ratio

Fig. 2. Box plots for the three performance metrics

model on which we are basing our analysis we can consider our tests with respect to three populations. One population is cumulative monthly returns of pairs averaged over the 14 different parameter sets using Pearson correlation in the trading strategy amongst *all* ‘highly’ correlated pairs in the market. The other populations are similarly defined, where instead we use Maronna and Combined correlations. The averaged cumulative monthly returns of the 1830 pairs yields 1830 sample data points per population.

Details of this more rigorous statistical approach are not be included this paper, and will be the subject of further studies. The end goal of these future studies would be to give a clearer picture of which correlation measure performs best under different scenarios. These initial results shed some light on the question and lead the way to further explorations.

VI. CONCLUSIONS

To design a pair trading strategy using high-frequency data, and to backtest and compare with existing strategies, requires intensive computational resources. This paper demonstrates the limitations of using Matlab to meet this computational challenge, and the promise of using parallel systems like that of MarketMiner. MarketMiner can accelerate the data analysis process to consider real-time trading across the whole market, and gives the promise of fast and accurate backtesting across many alternate strategies.

We have also presented some preliminary work on comparing pair trading strategies using three different measures of correlation - Maronna, Pearson and Combined. Preliminary results suggest that there are some important differences among these measures for trading, each with different strengths and weaknesses in terms of their risk vs. return profiles. These preliminary results motivate further investigation into determining the characteristics of each correlation measure. Further experiments will include considering more parameter sets, identification of optimal parameter sets for a given correlation measure, longer time frames (more than one month) and a larger universe of stocks.

Future studies would also benefit from considering various “implementation shortfalls” that occur in practice such as transaction costs, moving the market (on big orders) and lost opportunity (inability to fill an order).

Acknowledgments: We would like to thank Darren Clifford of PairCo for his valuable feedback and suggestions on our pair trading strategy.

REFERENCES

[1] E. Gatev, W. N. Goetzmann, and K. G. Rouwenhorst, “Pairs Trading: Performance of a Relative Value Arbitrage Rule,” *SSRN eLibrary*, 2006.
[2] “The world according to quants: Enter alpha generation platforms,” 2008.
[3] “ClariFi ModelStation.” [Online]. Available: http://w.clarifi.com
[4] “Alpacet Discovery.” [Online]. Available: http://ww.alphacet.com
[5] “Openquant.” [Online]. Available: http://www.smartquant.com/openquant.php
[6] “Marketcetera Trading Platform.” [Online]. Available: http://www.marketcetera.com/
[7] K. Kola, A. Chhabra, R. K. Thulasiram, and P. Thulasiraman, *A Software Architecture Framework for On-Line Option Pricing*. Springer Berlin / Heidelberg, 2006, pp. 747–759.
[8] G. Pauletto, “Parallel monte carlo methods for derivative security pricing,” in *In Computing in Economics, Finance 2000*. Springer-Verlag, 2000, pp. 650–657.
[9] M.-P. Leong, C.-C. Cheung, C.-W. Cheung, P. P. M. Wan, I. K. H. Leung, W. M. M. Yeung, W.-S. Yuen, K. S. K. Chow, K.-S. Leung, and P. H. W. Leong, “Cpe: A parallel library for financial engineering applications,” *Computer*, vol. 38, no. 10, pp. 70–77, 2005.
[10] “Marketminer analytics platform,” 2008, Scalable Analytics Inc. [Online]. Available: http://www.scalableanalytics.com
[11] “Third annual high performance computing users conference report,” 2006, council on Competitiveness.
[12] C. Rostoker, A. Wagner, and H. H. Hoos, “A parallel workflow for real-time correlation and clustering of high-frequency stock market data,” in *IPDPS*. IEEE, 2007, pp. 1–10.
[13] R. Maronna, “Robust m-estimators of multivariate location and scatter,” *Annals of Statistics*, vol. 4, no. 1, pp. 51–67, 1976.
[14] J. Chilson, R. Ng, A. Wagner, and R. Zamar, “Parallel computation of high-dimensional robust correlation and covariance matrices,” *Algorithmica*, vol. 45, no. 3, pp. 403–431, 2006.
[15] B. Do, R. Faff, and K. Hamza, “A new approach to modeling and estimation for pairs trading,” 2006. [Online]. Available: www.fma.org/Stockholm/Papers/PairsTrading_BinhDo.pdf
[16] P. Nath, “High Frequency Pairs Trading with U.S. Treasury Securities: Risks and Rewards for Hedge Funds,” *SSRN eLibrary*, 2003.
[17] M. S. Perlin, “M of a Kind: A Multivariate Approach at Pairs Trading,” *SSRN eLibrary*, 2007.
[18] ——, “Evaluation of Pairs Trading Strategy at the Brazilian Financial Market,” *SSRN eLibrary*, 2007.
[19] H. Green, B. Schmidt, and K. Reher, “Algorithms for filtering of market price data,” *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, pp. 227–231, Mar 1997.
[20] T. N. Falkenberry, “High frequency data filtering,” 2002. [Online]. Available: http://www.tickdata.com/FilteringWhitePaper.pdf
[21] M. M. Dacorogna, R. Gencay, U. Muller, R. B. Olsen, and O. V. Olsen, *An Introduction to High Frequency Finance*. Academic Press, New York, 2001.