MS&E 448: Big Financial Data for Algorithmic Trading
High Frequency Trading
Final Report

Thibaud Bruyelle, Tom Morvan, Brian Lui, and Julius Stener

ABSTRACT

As the 2020 coronavirus pandemic introduces an unforeseen level of uncertainty to the global financial market, high frequency trading once again attracts market attention on whether it proves to be effective in defending assets against global economic downturn. This project leverages high-frequency data from the proprietary MayStreet simulator to explore two common algorithms to generate alpha on high-frequency data: making (1) directional bets with mean reversion and (2) relative bets with pairs trading. This paper provides empirical results and discussions on the details of each trading strategy.

INTRODUCTION

High-frequency trading is made possible only with high-frequency data at a much higher granularity than per-minute stock prices publicly accessible online. We are grateful to have access to the proprietary MayStreet simulator for this class project to not only analyze high-frequency data but also simulate real-world complexities that high-frequency trading strategies face.

MayStreet Simulator

The MayStreet simulator is a powerful tool for backtesting high frequency trading strategies in daily increments. The real utilities and advantages of the simulator over other backtesting platforms such as Quantopian is iteration speed, data granularity, data flexibility and integrated execution modelling. However, with these advantages, a considerable number of disadvantages exist, including notable reliability issues, a lack of multi-day backtesting support and an inflexible backtesting platform.

While many platforms, including Quantopian, similarly provide Python wrappers over existing C++ backtesting libraries, the ease of use of MayStreet's simulator far outweighs its competitors. The full extent of the simulator class' capabilities can be adequately learned within the span of a day given the appropriate documentation, and the high-level simplicity provides a large degree of customization of outputs as well as unintended and invisible errors at execution.

However, the small issues and idiosyncrasies learned from working with the simulator consistently for 6 weeks are dwarfed by the value the simulator provides in data granularity and execution simulation. Outside of in-house backtesting platforms used by high-AUM HFT firms, very few platforms exist which provide event-level granularity. Below are plots of AAPL bid/ask prices and returns on the 2015/01/21 trading day. The plot highlights just the first level of the order book, but the simulator afforded us access to every existing level of the order book. Understandably, this level of granularity was invaluable in our research.

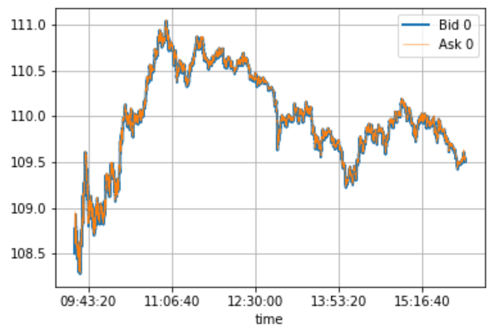

Figure 1: AAPL Bid/Ask

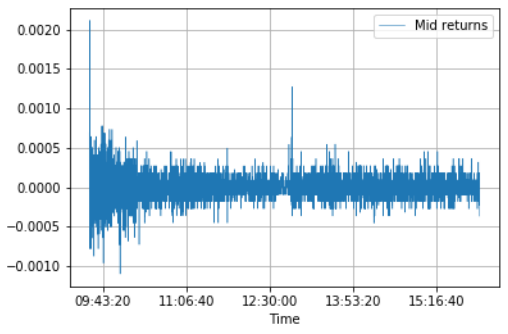

Figure 2: AAPL Mid returns

In addition, the simulator provides execution simulation through session.add_order which executed whatever order our strategy placed as if it were really in the real order book at the time it was placed.

Of note, the backtesting period available provided access to at most one trading day per backtest which limited the ability to concatenate multiple trading day's results to one another. In an effort to work around this, we built a custom backtesting wrapper in Python which tested over a given number consecutive trading days and concatenated the results. While this worked well, backtests do not run in linear time, partly because the number of orders on a given day is not uniform across trading days and partly because the simulator had a minimum "loading time" of about 20 second. Therefore, even a modest backtest over 3 days with roughly 10 traded tickers could take as long as 3 minutes. Of course, this is fast relative to the amount of information being processed, but could've been optimized with integrated multi-day backtests in the simulator.

Lastly on the simulator itself, it must be noted that the simulator experienced significant reliability issues. Over the 7 weeks between initial access and the time of this writing, the simulator was working less than five and half weeks. The outages included a solid week the penultimate week of our research and severely limited the optimization testing and hyperparameter tuning.

All in all, the power and utility of the simulator cannot be overlooked. Not only did it provide industry-leading quality data, but it provided an entire execution framework which saved our group multiple orders of magnitude more time than was lost due to the simulator intermittently not working. We are incredibly thankful and appreciative of MayStreet for allowing us to use their platform.

1. MEAN REVERSION

Mean Reversal relies on the assumption that longer term moving averages are more consistent than shorter term moving averages. Following that assumption, a traded asset is expected to return to its own mean with time, and one can utilize this assumption to generate a trading signal across a given time period. For what it captures, the encoding in pseudocode of this trading strategy is remarkably simple. Given a price for an asset on which to perform mean reversal, the short term mean, s_1, of length n_1 and the long term mean, s_2, of length l_2 can be calculated simply by taking the mean of the previous l_1 and l_2 prices, respectively, of that asset. Concurrently, an indicator variable, one_above_two, is kept which is 1 (True) if the short term mean was above the long term mean and 0 (False) otherwise. Then,

```
if one_above_two and s_1 < s_2:
    buy(asset)
elif not one_above_two and s_1 > s_2:
    sell(asset)
```

Given this simple strategy, the logical next steps to determine the validity of the strategy are (1) Investment Universe Selection, (2) Modeling, (3) Portfolio Construction and (4) Risk Assessment.

Investment Universe Selection

Over the course of our brief 7 weeks with the simulator, we explored several investment universes before selecting the current one. Due to the obvious need of high volume in any high frequency trading application, the initial explorations of Mean Reversion traded on the S&P 500.

In an effort to gain a statistical intuition and methodology for investment universe creation, we uilitze the Hurst exponent. The Hurst exponent, H, measures the long-term memory of a time series, characterising it as either mean-reverting, trending or a random walk. We have:

$$\mathbb{E}(\frac{R(n)}{S(n)}) = Cn^H$$

where $0 < H < 0.5$ characterizes mean reverting, $H = 0.5$ characterizes Brownian motion (random walk) and $0.5 < H < 1.0$ characterizes trending. Utilizing a randomized-trial Hurst exponent calculation over the chosen backtesting period, we found that the Russell 2000 (cleaned, as discussed below) has a Hurst exponent of

0.6249 and the S&P 500 had a Hurst exponent of 0.6581. While statistically both were trending, the Russell 2000 ”trended” less than the S&P 500. Note that we utilized randomized-trials because the extreme time cost of backtesting simulations prevented us from (1) testing on all of the tickers available and (2) testing on all the time periods available. Interestingly, although over significantly different backtesting periods and over very different interval lengths, these value for the Hurst exponents are almost flipped to those of Precaido and Morris in the 2008 paper *The Varying Behavior of U.S. Market Persistence* in which they evaluate similar statistics.

Following these results, the Russell 2000 became our preferred investment universe. Unfortunately, finding an quality open-source list or csv of the Russell 2000 is rather difficult, and we were forced to scrape and collate the tickers from a variety of websites.

However, the Russell 2000 had some additional undesirable qualities. Namely, due to the nature of small caps, a much smaller percentage ($< 75\%$) traded over the entire backtesting period, and the traded volume of small caps is significantly less than the traded volume of large caps.

As a result of the former, our investment universe shrank even more to just 1420 tickers to ensure a reliable backtesting platform across the last 4 years of trading. Sadly, the MayStreet simulator does not provide a method for verifying whether a ticker is traded over a given time period, nor does it provide a means of ticker-specific exception handling when a single ticker causes the failure of an entire universe's backtest. Therefore, in order to successfully create the testable investment universe, we were forced to utilize nearly a week of kernel time in which we tested each ticker across 8 dates which spanned our backtesting period.

$$30 \text{ seconds/simulation } * 2000 \text{ tickers } * 8 \text{ dates } = 5.5 \text{ days}$$

The latter issue, namely the lack of traded volume, was handled during backtesting and live during simulation. The backtest didn't eliminate the ticker itself as a contender, and instead opted for discounting any results if the previous days' volume didn't reach a hyper-parameter-set minimum quantity.

Modeling and Results

To develop trading signals from mean reversion, our algorithm tracks short-term and long-term moving averages and submits an order to the exchange whenever the two moving averages cross over. Anticipating the short-term average to return to the long-term average whenever there is any deviation, we buy a ticker in hopes of leveraging a rising price trend when its short-term average drops below its long-term average, and sell it in hopes of escaping a falling price trend when its short-term average surpasses its long-term average instead.

To assess the viability of our model, we track the profit-and-loss (PnL), inventory of the traded ticker and cash balance at every single tick to get a full sense of the progress of algorithm over time. While basic, the winning algorithm should return the maximal end-of-day PnL, without incurring too much risk.

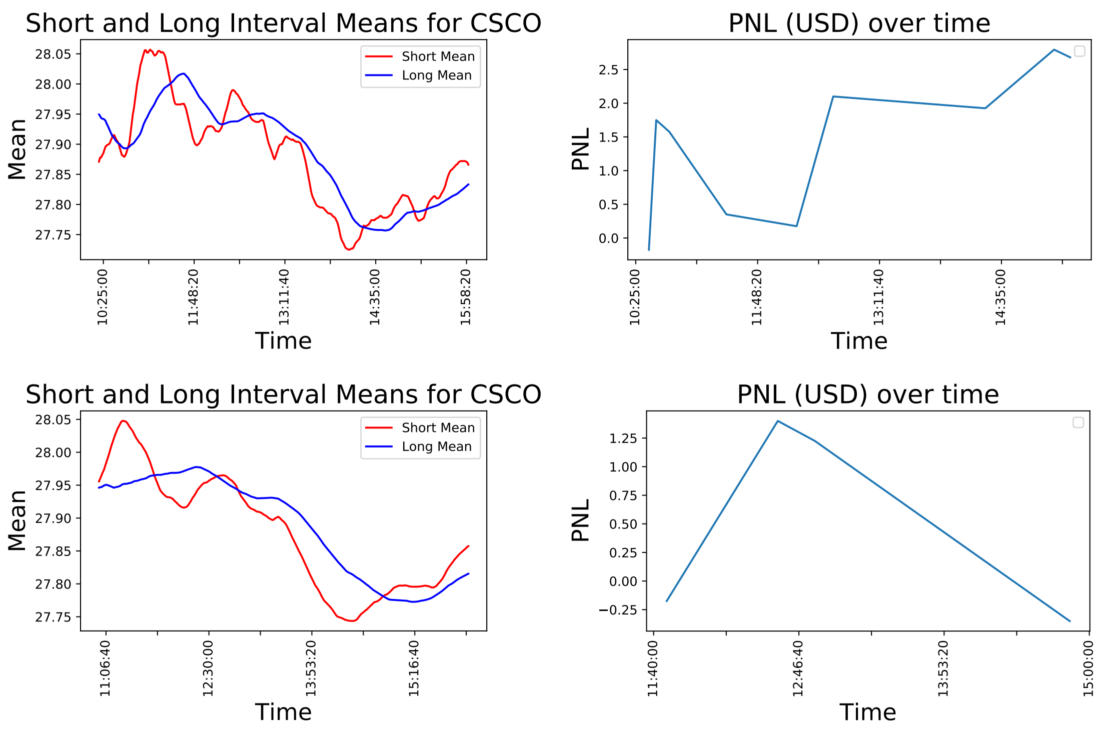

Figure 3: (Modelling) When trading the Cisco stock (ticker: CSCO), choosing a better pair short-term and long-term window sizes (**Top: 10 and 50**) results in a positive, better PnL ($2.68, with $0.297 per trade) over the trading day of January 21, 2015, than a sub-optimal pair (**Bottom: 30 and 90**) with a negative, worse PnL ($2.68, with $0.297 per trade).

However, by visualizing the moving directions of the short-term and long-term means after the cross-overs in Figure 3, we see the mean-reversion philosophy only works if the stock prices show a favorable price trend between the cross-overs. The locations of these cross-overs therefore become key to whether we profit from entering a position on the ticker between two opposite, consecutive cross-overs. These cross-over locations in turn depend on the following hyper-parameters:

1. Window size for the short-term moving average
2. Window size for the long-term moving average

With this realization, it gives us a better focus on what hyper-parameters to optimize over to maximize alpha. While the following hyper-parameters are inputs to our algorithm:

Trading decision	Moving averages between cross-overs	Favorable price trend
Buy	Short < Long	Rising
Sell	Short > Long	Falling

Table 1: Source of alpha

1. Time interval between consecutive reviews of current orders and signal generation (timer callback window for the MayStreet Simulator) (1 minute)

2. Order size (maximal number of shares constrained by the maximal nominal amount of each order)

3. Order price for each limit order (best bid/ask price in the order book)

4. Price to track within the order book at every time instant (mid-price, i.e. average of the best bid and ask prices),

we identify them as hyper-parameters secondary to having significant influence on the alpha performance. Mindful of the size of the hyper-parameter spaces we explore, we make an educated guess on these hyperparameters to save computational power on hyper-parameter tuning on the most alpha-generating portions.

While our mean-reversion strategy works online, we also backtest our strategy on a custom length of days at random times during the last 5 years to ensure generalizability of our mean-reversion algorithm.

Alpha modelling

Our source alpha lies in the predictions on the price trend in Table 1 between each pair of opposite, consecutive cross-overs of the short-term and long-term moving averages.

This prediction is based on the historical observation and intuitive assumption that any deviation from the long-term market behavior must vanish over time, and hence that any short-term abnormal prices must revert to the long-term mean. According to this prediction, we make directional bets on each ticker. Our algorithm enters when the short-term moving average crosses the lower side of the long-term moving average, anticipating a mean reversion, and exits vice versa.

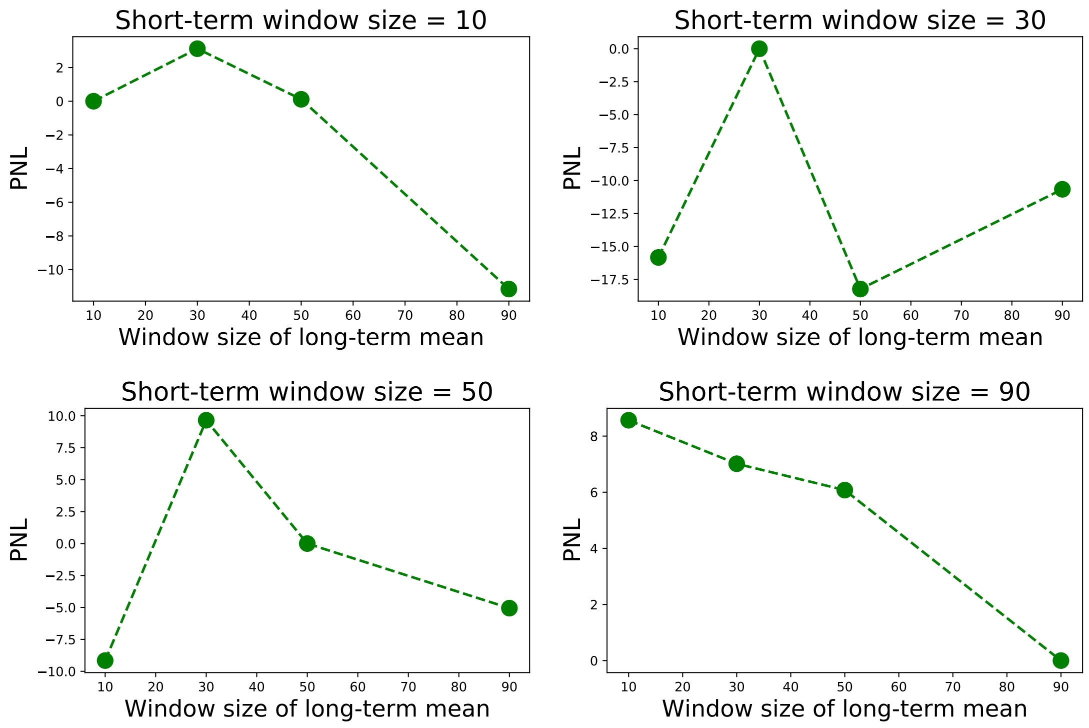

Figure 4: Hyper-parameter tuning of two hyperparameters - short-term and long-term window sizes - to maximize the end-of-day PnL (USD). Notice that the mean-version performance varies largely as the window sizes change slightly, to the extent that the PnL may even be higher when the short-term window size surpasses the long-term one.

By tuning the two hyper-parameters, short-term and long-term window sizes, as in Figure 4, we see the PnL of our mean-reversion algorithm largely depends on the window sizes, and can flip its sign just by a slight change in the window sizes.

Portfolio Construction

Our mean-reversion algorithm, compared to pairs trading, has the advantage of trading every single ticker in a self-contained fashion, making directional bets on each ticker over time instead of relative bets among tickers. This makes trading a portfolio using mean reversion as easy as trading each ticker individually and combining their PnLs.

Within each ticker, since we have to clear our position at the end of each pair of consecutive, opposite cross-overs, in our mean-reversion strategy, the order size we specify whenever we buy or sell the ticker at a cross-over determines our inventory dynamics of the ticker. We set the maximum allowable nominal amount for each order, and cap the order size according to this product amount as if the order were executed at the mid-price.

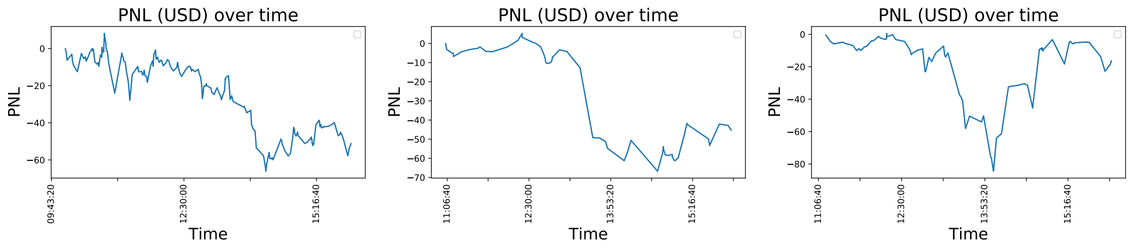

Figure 5: When trading on a portfolio consisting of {`AAPL, CSCO, MSFT, SPY, TSLA, LMT, BA, MRCY, AAN, AAOI, AAON, AAT, AAWW, ABG, ABCB, ABM`}, our mean-reversion strategy improves its end-of-day PnL performance as the short-term and long-term window sizes change from `(30, 10)` (-\$51.23, with -\$0.213 per trade), to `(90, 30)` (-\$45.30, with -\$0.768 per trade), and to `(90, 10)` (-\$16.59, with -\$0.180 per trade).

We are also aware of another formulation of mean reversion which offers a portfolio-wide strategy. In this alternative strategy, the algorithm buys the lower 20% performing stocks in the portfolio and sells the upper 20% performing stocks, anticipating the under-performing assets will do better in the future while the over-performing ones will ultimately lose their edge over time. This idea leverages the hypothetical existence of a portfolio-wide mean each asset should revert to in case they deviate from the market, and features a more interesting variety of portfolio weights we can assign to each asset.

We present our end-of-day PnL performance of our mean-reversion strategy on a custom portfolio contained inside the Russell 2000 universe in Figure 5.

Risk Management Philosophy

To measure the risk of our algorithm, we track the maximum drawdown and sample variance of the PnL across time. However, these measures of risk are dwarfed in significance by the huge dependence of the end-of-day PnL on the short-term and long-term window sizes, as presented in Figure 4.

This phenomenon can be interpreted in the following way. As we decrease the window size of any moving average, the resultant moving average becomes less smooth and has more fluctuations, which increases the chance of crossings between the current and other moving averages. New crossings may emerge and existing crossings may disappear in a discontinuous manner as we increase the window size of either moving average bit by bit. Since we make a new trade and hence a partial contribution to the grand PnL for every pair of consecutive, opposite cross-overs, the discrete emergence and disappearance of cross-overs also cause sudden jumps in the PnL. There is no guarantee that these jumps are always positive or negative. This explains the huge reliance of our algorithm's performance on the window sizes, as predicted above.

To solve this issue, one may smoothen out the sudden emergence and disappearance of cross-overs by setting a new, non-negative hyperparameter called the **margin**. When the margin is zero, this new algorithm reduces back to our original mean-reversion paradigm. Suppose the margin is positive. Then, we buy the ticker only when the short-term moving average is at least such a margin below the long-term moving average, and sell the ticker only when the short-term moving average is at least such a margin above the long-term moving average. Since we are altering the trading criteria from discrete time instants at the cross-over points to a range of time instants, this change will allow for softer entry and exit points throughout the entire trading day, and enables more variety over the order sizes and net positions we take in each ticker. In this way, we may be able to shift part of the PnL's heavy dependence on the window sizes to a new hyperparameter, the non-negative margin, which is a single scalar easier to optimize over, even for each fixed sub-optimal pair of short-term and long-term window sizes.

Execution Discussion

As mentioned in our MayStreet simulator section, the simulator was unfortunately broken for the penultimate week of the quarter. While we had implemented and backtested crude, signal-less strategies prior to this date, the lack of data or access to our strategies provided our team with a chance to reflect on the best ways to optimize our portfolio as well as find consistent sources of alpha. For mean reversion, we developed an interest in utilizing two strategies: grid search and discrete-action-space reinforcement learning.

A grid search involved exploring the low-dimensionality set of high-dimensional hyperparameters and was relatively simple to implement. However, grid search lacks the ability to train to an optimum and is inherently slower than other methods of tuning hyperparameters. A typical execution of the simulation can take anywhere from 30-60 seconds per day of backtesting depending on the parameters given the event space tracked. Couple the time to execute and the large dimensionality of the hyperparameters themselves (theoretically infinite), the time to execute across all 1420 tickers with 3 random trials of length 2 days across 19 different short/long mean lengths and 10 time intervals is conservatively 560 days. Therefore, we ran a subset of 14 tickers (representing just 1%) which took a little under 6 days. At the end of 6 days and within those 14 tickers, there were *no* consistently profitable assets for mean reversion across all 3 random trials. Furthermore, while we recorded the 5 most profitable hyperparameter values, they did not appear to be convex or approaching a global optimum.

A second potential solution is to use discrete-action-space reinforcement learning as a hyperparameter tuning technique. In theory, a reinforcement learning implementation such as DQN or DDQN could find the optimum hyperparameters due to the innate ability of RL implementations to perform in high-dimensionality action spaces. In this case, the state would be the underlying 1st-level metrics of the asset, for example previous day sharpe ratio, volume, etc., and the action would be the set of hyperparameters. However, the involved assumption is that the underlying data is convex and can be reduced to an optimum. Under this assumption, our grid search should have turned out a relatively convex set of optimums, but, as discussed above, this was not the case. Therefore, this cannot be taken as a valid means of tuning hyperparameters for mean reversion on this investment universe.

Retrospective Discussion

All in all, the aim of our implementation of a mean reversal strategy was to determine the viability of mean reversal in an intra-day/ high frequency trading application. Throughout the roughly 6 weeks we had working access to the simulator, we developed a clear understanding of the inherent difficulties of high frequency trading, and the extreme variability in returns based on many hyperparameter settings (discussed above). Within that context and while more study is needed, we believe that consistent profitability using mean reversion in high frequency trading is attempting to solve for an optimum within a non-convex function. In other words, finding a set of hyperparameters in a specific investment universe which are consistently profitable is more likely to be the result of careful choices for these hyperparameters than of statistics-driven optimizations.

2. PAIRS TRADING

The goal of this strategy is to identify pairs of stocks that show similar price patterns and to benefit from small divergences from this pattern. The strategy itself can be divided in two separate parts: **pair selection** and **pairs trading**. In the pair selection phase, we regress a basket of stocks on the S&P 500 returns and derive the intrinsic stock returns (uncorrelated from the market). We then perform a PCA on the residuals to identify similar stocks. During the trading phase, we track the spread of selected pairs and compute running statistics to identify positive trading signals. We typically expect to hold little inventory of stocks, and to have the strategy running every day during market hours.

Investment Universe Selection

One of the most important parts of the pairs trading strategy is to find a pair of assets that will be relevant. We spent a lot of time picking the right pairs and used several methods for this purpose. Among the strategies that we use for the pairs selection, we will discuss a *statistical-based* methodology and more experimental *machine-learning* based approach. The latter is only to confirm or disprove the conclusions from the statistical-based methods that were used to select consistent pairs of stocks.
MayStreet provided us high-frequency data so we used these data to select the pairs.

It is primordial to look at stocks that have similar properties and features on a daily basis. However, in order to make consistent choices, it is also very important to aggregate the results obtained on a daily basis to a larger scale.

Pairs Selection Methodology

The technical details and results of the selection are described in the **Modeling** section.

1. Choose a broad portfolio of stocks from different industries.
2. Pick a random trading day and retrieve the data from this day.
3. Statistical Analysis to get an idea of potential pairs.
4. For each potential pairs, we compute relevant metrics and perform several statistical tests with given thresholds. To illustrate, we compute the correlation or the Hurst exponent, and then we perform an Engle– Granger two-step method for testing the correlation.
5. Get rid of the pairs that fail the tests or that are above our thresholds.
6. Restart step 2 with the remaining pairs.

After applying this methodology to our portfolio, we applied *unsupervised* learning methods to confirm or disprove our results. Precisely, we used clustering methods such as *K-means* and *Affinity Propagation*. The advantage of using these two methods is the ability to use two different distance matrices. *K-Means* model is built upon *Euclidean distance* but it makes more sense to use the *correlation matrix* as a distance since we work with time series: this is why we also used the *Affinity Propagation* algorithm.

Finally, we included in our strategy some pairs from the same industry such as Visa and MasterCard or JPMorgan and Morgan Stanley but we also made less intuitive associations such as Apple with Mastercard or Apple with Goldman Sachs. The way a pair was traded is described in the Portfolio Construction section.

Modeling

First, let us illustrate what statistical analysis methods were performed in order to understand the data we had. Principal Component Analysis (PCA) is a good statistical model to observe correlation patterns within a portfolio. In order to make an analysis of the returns series R_t, we used SPDR S&P 500 returns M_t to compute the linear regression :

$$R_t = \beta M_t + \alpha + \epsilon_t$$

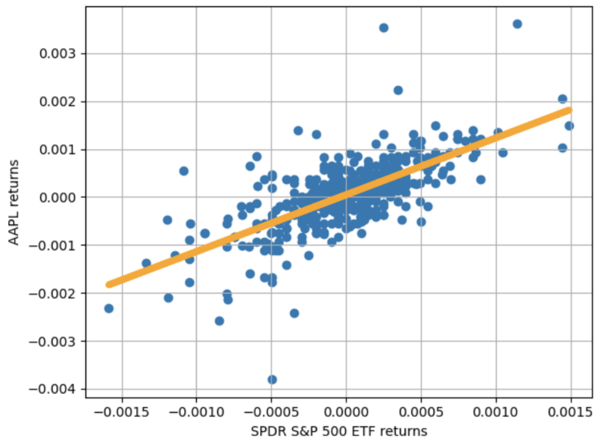

Figure 6: AAPL returns against Market

The idea is to get the intrinsic properties of the stocks returns without the market influence. This is why we applied the PCA framework to the residuals returns in our portfolio $(\epsilon_t)_t$.

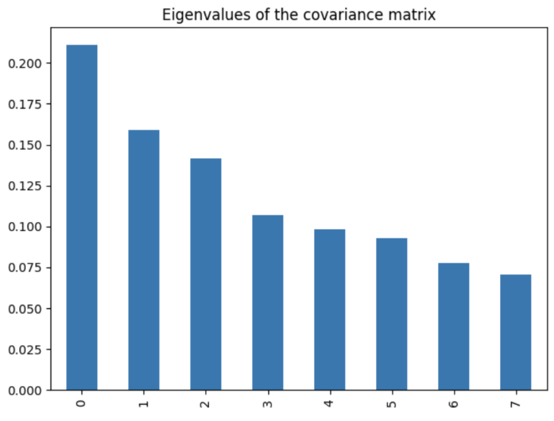

Figure 7

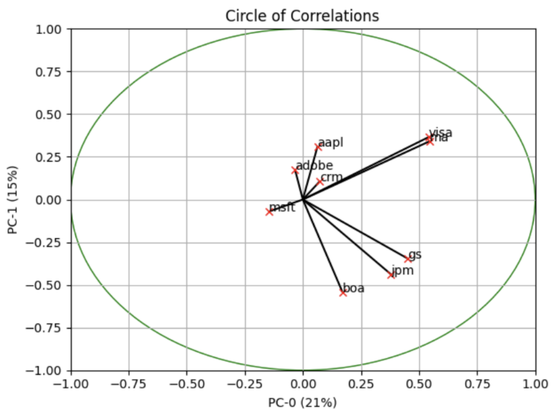

Figure 8

From the results above, we can notice that there exists several correlation patterns within our portfolio. These patterns are mostly clustered by industry. It gives us an idea of potential pairs that are close enough to build an efficient strategy. Furthermore, the *Affinity Propagation* algorithm can provide information from a different perspective depending on the distance matrix used.

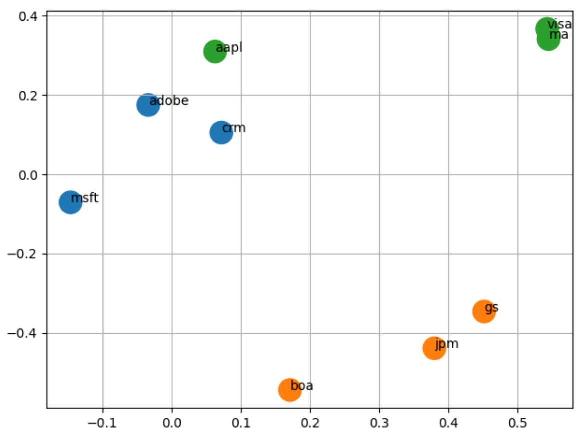

Figure 9: *Affinity Propagation clustering* with PCA dimensionality reduction

In this case, we see that AAPL and MA are clustered together but the clusters are not based on the correlation relationships between the stocks. By considering the fact that Apple introduced its Apple Card recently, by working with Goldman Sachs and Mastercard, we could take advantage of this collaborative association to create a less intuitive pair.

Second, let us describe some of the metrics and tests used to make a precise pairs selection. In order to know if a stock has mean reversal properties, we used the Hurst exponent as described in the Mean Reversion sections. Furthermore, once two stocks are pre-selected as a potential pair, we compute the Pearson correlation coefficient which measures how the securities move together. For two stocks A and B, this metric is given by:

$$\rho_{A,B} = \frac{cov(X_A, X_B)}{\sigma_{X_A}\sigma_{X_B}}$$

Then the second step is to test for co-integration between the stocks. For this purpose, we used the Granger – Engel test to check if two series x_t, y_t are cointegrated, *ie*, when the distance between them does not change drastically over time. The test assumes that $x_t, y_t \sim \mathcal{I}(1)$ so we used a ADF test to show that the first difference of x_t and y_t are stationary. Then it tests for $u_t \sim \mathcal{I}(0)$ in the model : $y_t = \alpha + \beta x_t + u_t$. Two co-integrated stocks were a very good candidate to bulild a pair and usually it was our last criteria to do the pair selection.

Portfolio Construction

Once a pair is identified (e.g. Visa vs. Mastercard), we create a synthetic asset S on which we will trade for the day:

$$S = n_1 S_1 - n_2 S_2$$

With (S_1, S_2) the pair of stocks selected, and $(n_1, n_2) \in \mathbb{N}^{*2}$ such that

$$n_i = round\Big(\frac{\text{Trade nominal}}{\text{Open mid price of } S_i}\Big)$$

Trade nominal is a model parameter that will be discussed in the next sections.

Through this definition, S becomes a vehicle of all the relevant information to monitor by our strategy. By defining (n_1, n_2) in this manner, we ensure that:

- $Price(S) \approx 0$.
- $Cash\ balance \approx 0$, e.g. we have little to no cash flow when we enter a trade.
- S is uncorrelated to market movements.
- Cash borrow costs become negligible compared to equity borrow costs.

Alpha Modeling

Our first approach was to model our asset S by a $\mathcal{N}(\mu_S, \sigma_S^2)$ law, and to derive our trading signals on the position of S with regards to (μ_S, σ_S^2). Our first challenge was to have an accurate measure of these two statistics throughout the trading day. We opted for window based statistics, and computed the running sample mean and variance based on two parameters:

- **Time-interval**: Time interval at which we request market data, compute statistics and send trade orders.
  Typical Value: 1 second
- **Window-size**: Number of *time intervals* on which define the sample for statistics computation.
  Typical Value: 300

We then define:

$$Z = \frac{S - \mu_S}{\sigma_S}$$

and look at the position of the time-series based on a third parameter:

- **Threshold:** Multiple of σ_S serving as trigger for trading signals.
  Typical Value: 2

The alpha signals were then directly derived from this *threshold* value:

- If $S > threshold \Longrightarrow$ Sell S
- If $S < threshold \Longrightarrow$ Buy S

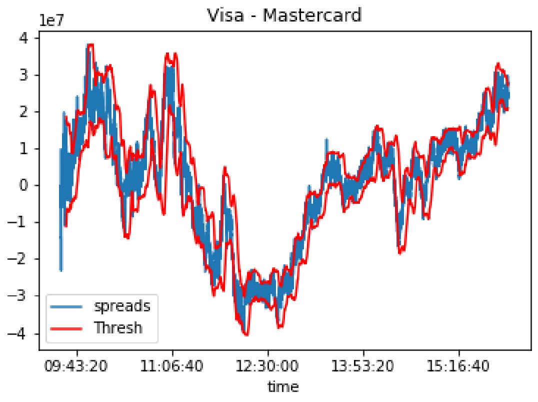

Figure 10

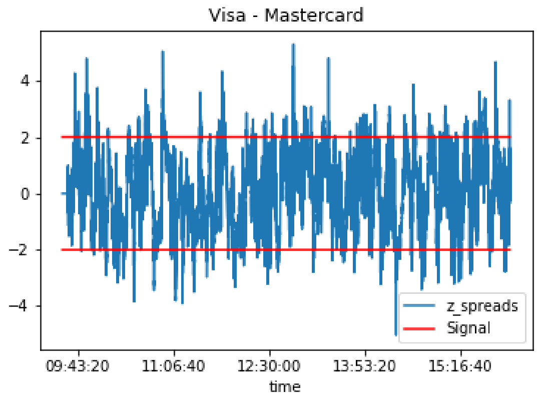

Figure 11

Risk Management Philosophy

We identified multiple sources of risk in out strategy:

Execution Risk

As MayStreet provides a realistic HFT order management system, we were confronted by the execution risks of automated strategies. To prevent the realization of spurious trades, we limited ourselves to market limit orders hitting the first level of the order book. When our trades were not fully executed by the next *time interval* due to slippage or partial fill, we would cancel all outgoing orders and send new trades if the signal was still present.

Regulatory Risk

To ensure that our activity was compliant with market regulations, we monitored our fill ratio :

$$\text{fill ratio} = \frac{\text{number of orders filled}}{\text{number of orders sent}}$$

which widely passed exchanges requirements. As we were only removing liquidity from the order book, we were safeguarded from *spoofing* and *blinking* behaviours. Finally, the nature of our strategy prevented us from filing *wash trades* or *matching orders*.
Concerning equity borrows and *naked shortselling*, we limited ourselves to trading stocks with high ADV (average daily volume) so that we could expect prime brokers to always have inventory at reasonable cost.

Inventory Risk

By modelling $S \sim \mathcal{N}(\mu_S, \sigma_S^2)$ we expected to have :

$$\mathbb{P}(S > \mu_S + threshold \times \sigma_S) = \mathbb{P}(S < \mu_S - threshold \times \sigma_S)$$

and not to build inventory on average. Unfortunately MayStreet allowed only single day market replay, and we could not track our inventory on longer time periods. When we looked at daily inventories, we did see some inventory build-up and we rarely finished the trading day flat.

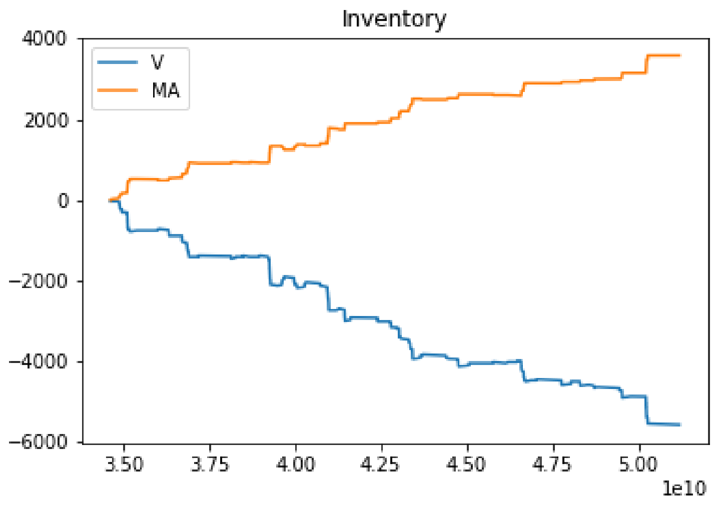

Figure 12: *Inventory build-up*

In order to mitigate the latter, we added two parameters:

- **Maximum Delta:** Maximum dollar value of stocks allowed in inventory.
  Typical Value: $\$1,000,000$

- **Unwind Threshold:** Multiple of σ_S serving as trigger for unwinding positions.
  Typical Value: 0.5

While the first was very efficient in limiting our maximum position, the latter drastically lowered our strategy's performance (dollar per trade, daily PnL) without significantly reducing the inventory build-up. Furthermore, certain market configurations showed that our asset S could follow well defined trends, and that the performance of our strategy was then drived by the trend of our inventory.
In order to identify these trends and adapt our strategy, we computed a second running average on a longer window size. We expected to spotlight trend signals from this statistic, and adapt our inventory accordingly. Due to lack of time, we weren't able to finish this module of our trading algorithm.

Market Risk

The constructed asset S is in theory hedged of market risk. However, our computation of (n_1, n_2) does not hedge us perfectly, but ensures that we have a maximum exposition of $2\times$ *Trade nominal* to market movements. This is our fifth parameter:

- **Trade nominal:** Dollar quantity to trade for each stock at every signal. *Typical Value:* $\$5,000$

The fact that we cancel all unfilled trades at each time interval can also lead to a market position (e.g. one leg of the trade is filled but not the other). Because of these reasons, we closely monitor the dollar value of our inventory at all times, and correct our exposition by sending only one leg of the trade at the next signal. A good proxy for estimating our market risk is to look at our daily cash balance, which stays between $\pm 2\times$ *Trade nominal* as expected.

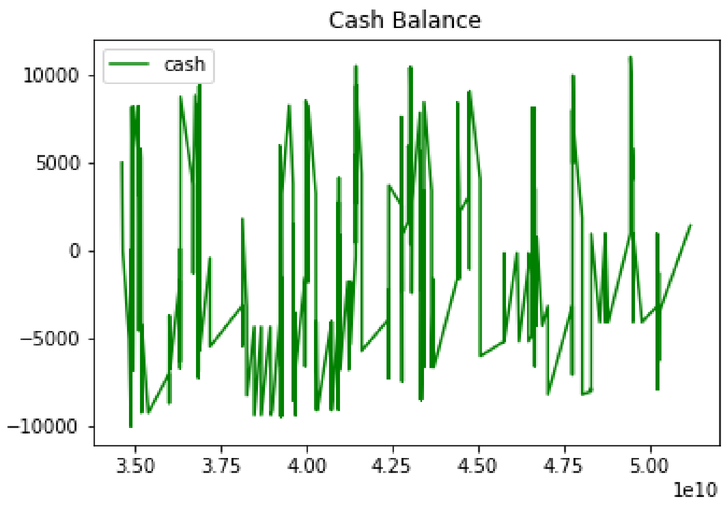

Figure 13: *Cash Balance*

Modeling risk

From a statistical point of view, we took care of not adding future information in our model that could bias our activity. One of our main concern was with the MayStreet simulator, and the fact that we didn't have external agents' responses to our activity. To cope with this absence, we set our *Max Delta* and *Trade Nominal* very low compared to the ADV, so that our activity would go unnoticed in a typical trading day.

Retrospective Discussion

When backtesting our strategy on MayStreet market replay, we realized that the performance of our strategy was highly sensitive to all our parameters, even the seemingly less important ones. Below are plots of the behaviour of our algorithm on a given trading day. All parameters being equal, on the left *Trade nominal* = \$5,000, on the right *Trade nominal* = \$10,000.

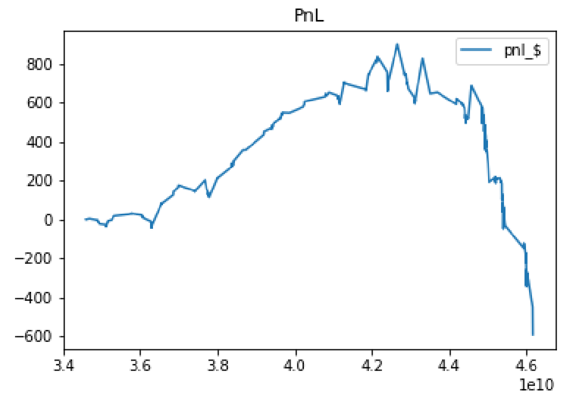

Figure 14

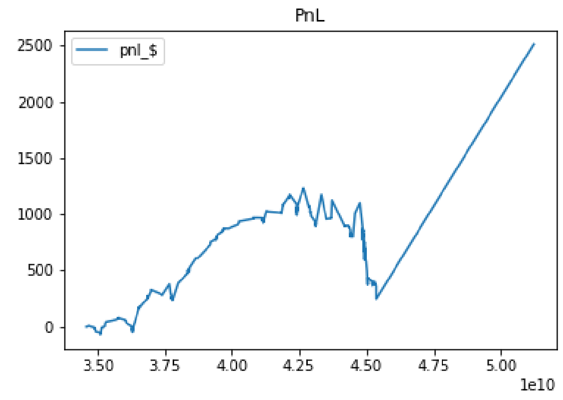

Figure 15

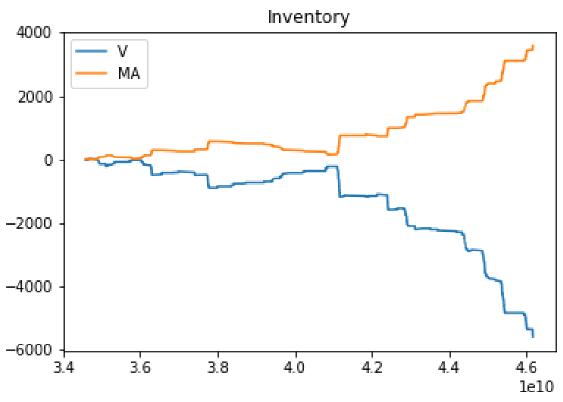

Figure 16

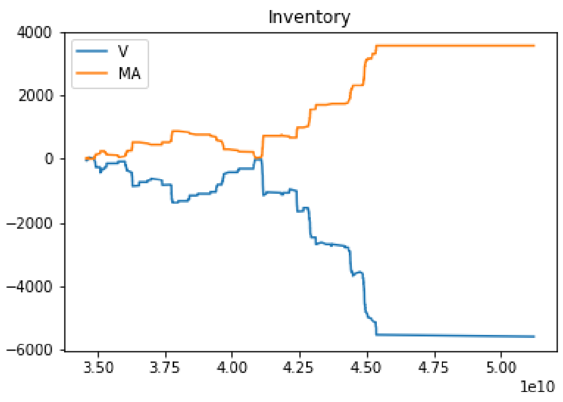

Figure 17

Given these acute variations, we thought of optimizing our model on our five parameters. However, due to lack of time and incompatible MayStreet framework ($\approx$ 50 seconds to simulate one trading day), we were only able to try a discrete subset of parameters, which showed random results depending on the day.
To conclude, we were not able to isolate a consistent alpha signal, but we created a robust Pair Trading framework that allowed us to assess the quality of our strategy.