

LZ2

PAIRS TRADING WITH MACHINE LEARNING

by

WONG Wen Yan, KO Chung Wa,
BRENDAN Tham Guang Yao,

Advised by

Professor NEVIN L. Zhang

Submitted in partial fulfillment
of the requirements for COMP 4981
in the

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
2018-2019

Date of submission: April 17, 2019

Contents

1	Introduction	5
1.1	Overview	5
1.2	Objectives	6
1.3	Literature Survey	7
1.3.1	Pair Selection	7
1.3.2	Trading Logic	9
2	Methodology	11
2.1	Design - Trading Strategies	11
2.1.1	Recap of Pairs Trading Framework	11
2.1.2	Baseline Strategies - Pair Selection and Spread Definition	11
2.1.3	Baseline Strategies - Trading Logic using Rule Based Approach	12
2.1.4	Cointegration Method - Parameter Estimation	14
2.1.5	Main Strategy - Reinforcement Learning (RL) Trading Agent	16
2.2	Design - System Overview	20
2.2.1	Backtesting System for Trading Logic	20
2.2.2	Web Application	21
2.3	Implementation	23
2.3.1	Tools and Libraries Used - Trading Logic and Backtesting System	23
2.3.2	Tools and Libraries Used - Web Application	23
2.3.3	Software Engineering Practices	24
2.4	Testing	28
2.4.1	Testing of Trading Logic	28
2.5	Evaluation	29
2.5.1	Experiment Setup	29
2.5.2	Performance of Trading Strategies	32
3	Discussion	43
3.1	Comparison of Baseline Strategies vs RL Trading Agent	43
3.1.1	Limitations of Baseline Strategies	43
3.1.2	Ranking and Summary of Trading Strategies	47
3.1.3	Review of RL Trading Agent - Issues in Reinforcement Learning	49

3.1.4	How Scalable Are The Profits; is it Replicable in Multi-Billion Dollar Funds	50
4	Conclusion	54
5	Hardware and Software	55
5.1	Hardware	55
5.2	Software	55
6	Appendix A - Meeting Minutes	58
7	Appendix B	64
7.1	Profit condition for cointegration method (with log price spread)	64
7.1.1	Short the Spread	64
7.1.2	Long the Spread	65
8	Appendix C	67
8.1	RL Trading Agent Parameters	67
8.2	Baseline Methods Parameters	67
9	Appendix D	71
9.1	Distribution of work	71
9.2	GANTT Chart	72

Abstract

Pairs Trading is a simple trading strategy which involves finding a pair of stocks that exhibit similar historical price behavior, and then betting on the subsequent convergence of their prices in the event that they diverge. Since this trading style is very flexible and applicable to all asset classes, the methods to identify pairs trading signals is a widely researched topic. However, the effectiveness existing rule based strategies has reduced because of increased market competition as it is simple and easy to implement.

This project conducted rigorous studies on the existing pairs trading strategies proposed in academic papers. We implemented and backtested three of the most popular pairs trading strategies (Distance method, Cointegration method with Rolling OLS, Cointegration method with Kalman Filter) to better understand their profitability, strengths and weaknesses under different market conditions.

The core achievement of this project is a novel approach to perform pairs trading by adopting the Reinforcement Learning (RL) paradigm. The proposed RL trading agent was backtested over 3 separate periods (2016, 2017 and 2018) to facilitate a comprehensive comparison between the existing strategies and RL trading agent.

Finally, we have implemented a front-end web application for data visualization of the trade actions and performance of various trading pairs trading strategies.

Acknowledgements

Firstly, we are immensely grateful for the advice and guidance given by our supervisor, Professor Nevin Zhang from HKUST's Computer Science and Engineering Department. He has provided us with a clear framework of what we should achieve in our Final Year Project. With regards to the core of the project, which is reinforcement learning, Prof Zhang has advised us on both the theoretical aspects, such as choice of models and refining the problem definition of our project, as well as on the practical aspects, such as choice of dataset, rule of thumbs in testing procedure, and evaluation of the effectiveness of our trading agent.

During the consultation sessions, Professor Nevin has regularly challenged our ideas, prompting us to think more critically about the concept and underlying theories of our proposed model, and provided us with clear and constructive feedback. We have also benefited significantly from Professor Nevin's postgraduate course (COMP 5213), which covers the essential concepts in the reinforcement learning paradigm.

We are also very fortunate to have Assistant Professor Ningyuan Chen from HKUST's Industrial Engineering and Decision Analytics Department as the second informal advisor. Prof Chen's advice specializes in the existing pairs trading strategies (distance and cointegration method), which are implemented in the project for the purposes of being a baseline comparison to the main reinforcement learning trading agent. Prof Chen has also advised us on the design of our backtesting procedure, and has helped us understand the impact of the choice of backtesting parameters and the bias it introduces to the results.

Finally, we would like to thank Mr. Ted Spaeth, the Department of Computer Science and Engineering's communication tutor. Mr. Spaeth has helped us craft a comprehensive by pointing out the blind spots of our report so as to make the content more readable by the general audience without specialized background either in machine learning or trading. Mr. Spaeth has also assisted immensely in the results evaluation and discussion section by suggesting intuitive ways to visualize the data and teaching the audience to interpret the results.

1 Introduction

1.1 Overview

The Rise of Automated Trading Strategies

Every year, investment banks spend billions of dollars to invest in technology with the objective of facilitating efficient flows of funds globally and providing world class services. For example, in 2017, Citigroup and JPMorgan Chase & Co., allocated budgets of USD 8 billion and USD 10.8 billion respectively for maintenance and investment of their technology [1]. Besides investing in infrastructure, such as network security and large scale database management, automated trading algorithms are also an increasingly mission-critical area of development. Guy America, head of credit trading at JPMorgan, recently emphasized the importance of having state-of-the-art trading systems [19]. This is because investment banks want salespeople and traders to focus only on selling and trading, rather than spending time on repetitive and non-value-adding tasks which can be automated. Hence, algorithmic trading is an increasingly important area of development for the financial services industry, because it can significantly boost the efficiency and productivity of traders. Most importantly, having tools that can provide insight to trades to make informed decisions is a huge value added by technology.

Algorithmic trading can be defined as having a machine systematically send orders to the trading exchange to be executed on behalf of human traders. Professional algorithmic traders often employ two major classes of algorithmic trading strategies, namely high frequency trading (HFT) and statistical arbitrage trading. HFT is characterized by high turnover rates and high order-to-trade ratios that leverage high-frequency financial data, usually in the scale of mili- to micro-seconds. For developed markets, HFT is commonly used because of well-developed communications infrastructure for orders to be routed quickly. However, HFT has peaked in terms of its profitability because of reduced volatility in the market. Successful HFT strategies normally rely on positions with significant price swings within the short holding period. Therefore, HFT has become less profitable as the market is moving from active trading strategies to passive trading strategies such as ETF investments.

In contrast, statistical arbitrage involves simultaneously buying and selling assets that normally have the same price trends but whose prices diverge in the short term. This form of trading relies less on the speed of sending orders and more on the mean reverting nature of financial products, i.e., the phenomenon in which asset prices and returns eventually return back to the long-run mean of the historical time series. The rationale is that for similar companies or financial products, they should be priced similarly based on no-arbitrage pricing principle [13]. In other words, products with the same risk should have the same reward, therefore, their prices should fluctuate similarly.

Introduction to Pairs Trading

The most common manifestation of statistical arbitrage is in the form of pairs trading, which can be explained in the following example: Suppose there exist two stocks A and B which belong to the same sector and they are “financially similar” in terms of revenue earned and expense structure. Because both A and B carry similar risk, they should be priced similarly in theory. However, due to fluctuations in supply and demand, the prices of A and B may be subject to idiosyncratic factors which may cause their trading behaviour to temporarily diverge. In essence, a trader can take advantage of this situation by selling the relatively overpriced stock and buying the other, and then reversing the trades when their behaviour converges again.

Effectively, the trader is taking advantage of the mean reverting behaviour of the difference between the prices of stocks A and B. The additional benefit of doing so is that pairs trading virtually eliminates the “market factor”; that is, the risk that is common to all stocks in the market.

Secondly, another reason why pairs trading is commonly used by both machines and humans alike is the fact that when one does a pairs trade, he or she is effectively expressing a view on the relative price of a pair, which is easier to estimate as opposed to just trading one asset with a directional view. This is because the set of factors that affect a directional view is much larger than that of a relative view since a subset of it will be cancelled off. Thirdly, pairs trading is a trading style that is asset class agnostic; the requirement of HFT is to have high frequency financial data. This effectively limits the set of tradeable instruments to those that fulfil these conditions only. In contrast, the pairs trading only requires an instrument with are comparable, either economically or statistically. The set of tradeable instruments are much wider than that of HFT.

Although pairs trading can be used as the foundation for building a portfolio of stocks, there are several practical challenges in implementing this strategy:

- Pair selection: What are the characteristics of a *good* pair of stocks and how can we systematically quantify it?
- Trading logic: What are the optimal entry and exit levels? How can we limit the downside in the event if the trade goes wrong?

In terms of pair selection, the goal is to systematically identify similar companies based on some measure of “similarity” because the notion of similarity is highly subjective. Based on existing research, metrics such as *sum of squared difference* and *cointegration* are used to assess whether two stocks form a “good” pair (elaborated in **Section 1.3.1**). In terms of trading logic, the goal is to implement a trading agent that maximizes expected returns. Ideally, this agent can outperform rule-based trading algorithms.

This paper has three main contributions. Firstly, we have implemented existing pairs selection methods and test their profitability as our baseline comparison. Secondly, we have implemented an trading agent using reinforcement learning that outperforms the existing methods. Ideally, the agent will also outperform the market. Finally, we have implemented a simple webapp for the user to monitor the performance of the trading agent, calibrate the strategy parameters and better understand their performance through visual analysis.

1.2 Objectives

The end goal this project is to develop a system that will make profitable trades based on the pairs trading framework. The main area of focus is implementing a profitable trading agent using reinforcement learning. On the software engineering aspect, we will build the front end of the system as a web application.

From here on, we will define a **trading strategy** to be a combination of both **pair selection** and **trade logic**. We will also refer to existing pairs trading as baseline strategies.

Our project will focuses on the following objectives:

1. (Artificial Intelligence) To create a trading strategy that outperforms existing pairs trading strategies by:

- (a) Implement existing pairs trading strategies (distance method, cointegration method) as baselines for comparison
 - (b) Implement a profitable pairs trading strategy (trade logic only) using reinforcement learning (RL)
 - (c) Refine the RL Trading Agent to ensure the agent can perform generalization; perform pairs trade on other sectors and asset classes
2. (Software Engineering) To develop a user interface that will allow investors to:
- (a) Monitor trading actions, cumulative returns, and relevant trading statistics
 - (b) Understand the strengths and weaknesses of different pair trading strategies under varying market conditions; analyze their behaviour under changing market conditions

To achieve objective 1(a), we used `ibsync` library to retrieve and preprocess data from the Interactive Brokers API. Then, we implemented these methods and backtested them using `backtrader` library. To achieve objective 1(b), we modelled the problem as a Partially Observeable Markov Decision Process (POMDP) based on our definition of the agent and the environment. The trading agent is implemented using `TensorFlow` library.

To achieve objective 2, we implemented a web application using data visualization library (eg. `bokeh`). For the back end, a REST API will be implemented using `Flask` microframework to handle HTTP requests sent from the browser.

The biggest challenge in our project was defining the problem and choosing the right model to solve it. Based on our study, we believe that our problem can be modelled as a POMDP and it can be solved using the Policy Gradient approach (explained in **Section 2.1.5**).

1.3 Literature Survey

In this section, existing pair selection methods and forecasting models for modelling spread will be presented.

1.3.1 Pair Selection

Distance method

If we assume that similar stocks have similar trading behaviour, then measuring distance between stock price series can be used to quantify the similarity of price trends of stocks. Distance method was first introduced by Gatev et al. (1999) [7] for pairs selection. In this approach, each stock is paired with the stock that has the minimum sum of squared difference between the two normalized stock price series.

Given a starting stock S_0 and let $S = \{S_1, S_2, \dots, S_N\}$ be a set of N stocks under consideration. At time t , the difference between S_0 and S_i is defined as $d_{0,i,t} = p_{0,t} - p_{i,t}$, where $p_{i,t}$ and $p_{i,t}$ are the normalized price of S_0 and S_i at time t respectively. Then, using the distance approach, the matching partner of S_0 is S_j if

$$j = \arg \min_{k \in \{1, 2, \dots, N\}} \frac{1}{T} \sum_{t=1}^T d_{0,k,t}^2$$

This approach however has a disadvantage as mentioned by Have Van Der (2017) [8]. Minimizing the sum of squared difference between two normalized prices series will also minimize the variance of the spread. This

is not ideal because for a pair of stocks to be profitable in the pairs trading setting, their spread must have some degree of variance in order to have more trade entry opportunities to make a profit. The derivation from the paper [8] is as follows:

$$\frac{1}{T} \sum_{t=1}^T d_{0,k,t}^2 = \frac{1}{T} \sum_{t=1}^T d_{0,k,t}^2 - \left(\frac{1}{T} \sum_{t=1}^T d_{0,k,t} \right)^2 + \left(\frac{1}{T} \sum_{t=1}^T d_{0,k,t} \right)^2 = Var(d_{0,k}) + \left(\frac{1}{T} \sum_{t=1}^T d_{0,k,t} \right)^2$$

In addition, Krauss (2016) [10] pointed out that the selection of pairs using the distance method is not statistically sound. Theoretically, the spread between two stocks can have low sum of squared difference, but still be non-mean reverting (eg. remain at a constant level over time).

Cointegration method

Engle & Granger (1987) [6] first introduced the concept of cointegration where Vidyamurthy (2011) [17] first applied it in the context of pairs selection. Let X_t and Y_t be the natural logarithm of the price series of 2 assets, X and Y . If the series $\{X_t\}$ and $\{Y_t\}$ are integrated of order 1, and there exists a linear combination

$$Z_t = X_t - \beta Y_t - \alpha$$

such that $\{Z_t\}$ is a stationary time series, then $\{X_t\}$ and $\{Y_t\}$ are said to be cointegrated. As a result, we could expect $\{Z_t\}$ to exhibit mean-reverting behavior.

To test cointegration of two time series $\{X_t\}$ and $\{Y_t\}$, Engle & Granger provided a 2-step procedure:

1. Given that $\{X_t\}$ and $\{Y_t\}$ are both non-stationary, we first estimate the parameters β and α using ordinary least squares (OLS) regression.
2. Then Z_t is tested for stationarity where the Augmented-Dickey-Fuller (ADF) test (1979) [2] is most commonly used. If the ADF test is passed (ie. Z_t is stationary), then X_t and Y_t are cointegrated.

Several studies in the past have reported the effectiveness of the cointegration method. Dunis et. al (2010) applied the cointegration method to high frequency data and found that pairs selected via the cointegration method tend to be profitable in high frequency setting. Huck and Afawubo (2015) [9] tested both cointegration method and distance method on the same time period. The cointegration method generated higher returns of approximately 5% per month, while distance method failed to produce significant returns. This implies that the cointegration method is a better pair selection method than the distance method.

However, some studies have also pointed out some drawbacks of the cointegration method. For instance, Rajab Ssekuma [16] pointed out that, during the first step of the Engle & Granger cointegration method, it is possible to run the OLS regression with either X_t or Y_t as the “subject” of the equation. If the sample size for OLS is large, regressing in either manner should result in the same conclusion (whether X_t and Y_t are cointegrated) due to large sample properties. However, this argument may be invalid when the sample size is small.

1.3.2 Trading Logic

Rule Based Approach

Gatev et al. (1999) [7] used a simple trading rule to execute pairs trading. During the trading period, positions are opened when the spread diverges more than two historical standard deviations from its historical mean:

$$\begin{aligned} \text{Let } S_{t,i,j} &= P_{t,i} - P_{t,j} \\ \text{Open position when } S_{t,i,j} &\geq \mu + 2\sigma \text{ or } S_{t,i,j} \leq \mu - 2\sigma \\ \text{Unwind position when } \mu - \epsilon &\leq S_{t,i,j} \leq \mu + \epsilon \end{aligned}$$

This type of trading strategy can be considered as model-free and ad-hoc as there is no attempt to model or forecast the spread $S_{t,i,j}$. It is implicitly assumed that, whenever the spread diverges, it will converge back to the mean in the near future. The benefit of such strategy is that it is simple and intuitive. However, many academic articles have demonstrated that model based approaches are more profitable. For instance, an empirical study by Have-Van-Der [8] shows that the Stochastic Spread Method and Recurrent Neural Networks approaches are at least equivalent or better than the model-free approach proposed by Gatev et al.

Model-Based Approach

A model based pairs trading strategy uses a forecasting model to characterize how the spread of two assets varies over time. There are many benefits of employing a forecasting model within pairs trading, Andrew Pole (2007) [14] pointed out that the benefit of having a formal forecast function is that it provides a set of values to compare to realizations and thereby to judge the efficacy of model projections. In other words, the series of predictions errors, also known as residuals of the model, allows us to verify our assumption on the behavior of spread and gain better understanding.

Stochastic Spread Method

Elliot et al. (2005) [5] introduced a state space model for modeling the spread. In this state space model, the spread at price level,

$$S_{t,i,j} = P_{t,i} - \beta P_{t,j}, \text{ where } P_{t,i} \text{ is the log price of stock } i$$

is driven by a latent variable following an Ornstein-Uhlenbeck process:

$$d\xi_t = \kappa(\theta - \xi_t)dt + \sigma dB_t$$

where θ is the mean of the state variable, κ is the speed of mean-reversion and B_t is a standard Brownian Motion. The discretized version of this equation is an AR(1) process:

$$\xi_{t+1} = A + B\xi_t + C\epsilon_{t+1}$$

where $A = \tau \frac{\theta}{\kappa}$, $B = 1 - \kappa\tau$, and $C = \sigma\sqrt{\tau}$, τ represents the time interval between two observations and $\epsilon \sim N(0, 1)$. The observation equation, which describes how $S_{t,i,j}$ is related to ξ_t , is given by

$$S_{t,i,j} = \xi_t + H\omega_t$$

where $\omega \sim N(0, 1)$.

According to Elliott et al. (2005) [5], the parameters (A, B, C, H) can be estimated iteratively using the Expectation Maximization (EM) algorithm, the Kalman filter and Kalman smoother. The trading rule provided is based on the estimated parameters:

$$\text{Open position when } S_{t,i,j} \geq \theta + c_{i,j} \left(\frac{\sigma}{\sqrt{2\kappa}} \right) \text{ or } S_{t,i,j} \leq \theta - c_{i,j} \left(\frac{\sigma}{\sqrt{2\kappa}} \right)$$

$$\text{Unwind position when } \theta - \nu \leq S_{t,i,j} \leq \theta + \nu$$

$c_{i,j}$ is a parameter which can be optimized during the backtesting procedure.

The Stochastic Spread Method has a number of drawbacks. Firstly, it has rarely been applied to empirical data in the academic research, which means that there are uncertainties on whether it performs well in the pairs trading setting. Do et al (2006) [3] criticizes the model, stating that it restricts the long run relationship between two assets because the model assumes that any divergence from the mean is expected to be corrected in the future.

2 Methodology

2.1 Design - Trading Strategies

2.1.1 Recap of Pairs Trading Framework

In the general framework of a pairs trading strategy, there are two key components involved - pair selection and trading logic. In pair selection, an algorithm is run to identify pairs of stocks where each pair of stocks have similar trend over time.

For the remainder of the text, we will call trading actions as **long the spread** or **short the spread**. Loosely speaking, the spread definition has the general form $stock[Y] - stock[X]$. Therefore, long the spread means we expect the spread to increase, so the underlying actions will be to long $stock[Y]$ and short $stock[X]$, vice versa for short spread.

2.1.2 Baseline Strategies - Pair Selection and Spread Definition

As pairs trading is a widely researched trading strategy, we will use two popular approaches for establishing baseline comparisons with our main reinforcement learning strategy. The exact definition of “spread” for each method is summarized as follows:

Method	Spread definition
Distance	$Y_t - X_t$
Cointegration	$\log Y_t - \alpha \log X_t - \beta$

Model Free Approach - Distance Method

For this approach, the rationale is that pairs that are similar will have similar day-to-day price changes, therefore the mean squared error (MSE) should be small. The MSE for all $\binom{N}{2}$ pairs of pool of N stocks over a specified period T :

$$MSE = \frac{1}{T} \sum_{t=1}^T \left(\frac{(Y_t - X_t) - \mu_{Y-X}}{\sigma_{Y-X}} \right)^2,$$

where μ_{Y-X} and σ_{Y-X} are the mean and standard deviation of $Y - X$ over time respectively. We select the first $k\%$ (eg. 0.5%) of the pairs with the lowest MSE scores as good pairs to be executed in the trading strategy.

Statistical Approach - Cointegration Method

For this approach, all good pairs of stocks X, Y fulfills the following relationship:

$$\log Y_t = \alpha \log X_t + \beta + Z_t,$$

where $\alpha > 0$ and $Z_t = \log(Y_t) - \alpha \log(X_t) - \beta$ being the spread of this pair of stocks. To test whether a pair X, Y can be well represented using this model, we first estimate the parameters (α, β) using Ordinary Least Squares (OLS) regression. Then, an Augmented Dickey Fuller (ADF) test is used to test for the stationarity

of the residual Z_t . If a pair passes the hypothesis test (ie. p -value is below a specified level), then it is selected to be executed in the trading strategy [17].

2.1.3 Baseline Strategies - Trading Logic using Rule Based Approach

Once good pairs have been selected, a rule based algorithm is used to enter and exit positions based on the spread level. The same trading logic is applied to both distance method and cointegration method to ensure that the baseline methods are comparable.

General Framework of Trading Logic

At each discrete time interval, the algorithm will first check the current state of the portfolio (ie whether it is currently no position, long the spread or short the spread) and observe the spread value **at the end of** the observation period. Subsequently, it will decide an action depending on the pre-defined set of rules. (See Algorithm 1)

To evaluate the performance of the trades, we define the return of the trade as follows: if a trade is executed (ie. we SHORT the spread) at $t = 0$, then the return of the trade at time t is

$$r_t = y(Y_0 - Y_t) + x(X_t - X_0)$$

assuming that X is the stock in long position with x number of shares bought, and Y is the stock in short position with y number of shares sold, vice versa.

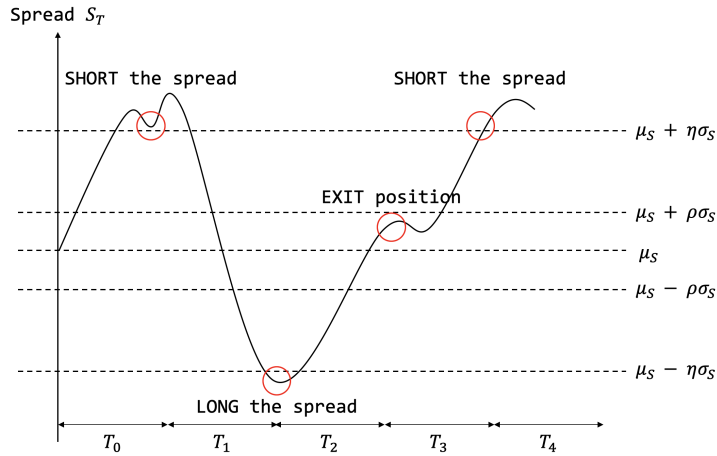


Figure 1: Graphical Example of Rule Based Logic

Algorithm 1 Trade logic at time t

$S_t \leftarrow$ spread at time t
 $\mu_S \leftarrow$ mean of spread, estimated by rolling window
 $\sigma_S \leftarrow$ standard deviation of spread, estimated by rolling window
 $\eta \leftarrow$ a positive number for setting entry threshold level
 $\rho \leftarrow$ a positive number for setting exit threshold level, should be less than η
 $L \leftarrow$ loss limit, a value for determining when we should give up on a losing trade

if current position is “*no position*” **then**
 if $S_t > \mu_S + \eta\sigma_S$ **then**
 SHORT the spread
 else if $S_t < \mu_S - \eta\sigma_S$ **then**
 LONG the spread
 end if

else if current position is “*long spread*” **then**
 if $S_t > \mu_S + \eta\sigma_S$ **then**
 EXIT the spread, and then SHORT the spread
 else if $S_t > \mu_S - \rho\sigma_S$ **then**
 EXIT the spread
 else if return of trade $< -L\%$ **then**
 EXIT the spread
 end if

else if current position is “*short spread*” **then**
 if $S_t < \mu_S - \eta\sigma_S$ **then**
 EXIT the spread, and then LONG the spread
 else if $S_t < \mu_S + \rho\sigma_S$ **then**
 EXIT the spread
 else if return of trade $< -L\%$ **then**
 EXIT the spread
 end if

end if

Enhancements to Trading Logic

To ensure that the results of this project can be replicated in real-world trading scenarios, we have included additional constraints on the trading cost aspect of the trading logic. Firstly, we have included **transaction cost** in accordance to the rates provided by Interactive brokers:

$$\text{Transaction Cost} = \min(\max(1, 0.005q), 0.01pq)$$

, where p is the price of the stock and q is the number of shares we transacted. Secondly, we have also implemented **borrowing cost** involved for the short sale of the stock (since it is required by law that a trader must obtain a stock before selling it). The borrowing cost is calculated as a flat rate quoted by the broker on pro-rata basis depending on the duration the short position is held (see equation below). Lastly, we have implemented a “hard-exit” function in the event of a **margin call**.

$$\text{borrowCost} = \text{borrowRate} \times \text{tradedValue} \times \text{numDaysHeld}/365$$

From the risk aspect, we have also implemented a “cut-loss” function as a parameter for tuning. The purpose of this parameter is to ensure that we limit the downside of the positions in the event if the spread never converges (ie. jump risk).

2.1.4 Cointegration Method - Parameter Estimation

In the cointegration method (see 2.1.2), a linear model is used to represent the relationship between the logarithmic price series of two stocks. Since α and β may change over time, they should be updated with new estimations regularly. We implemented 2 approaches:

- **Rolling OLS regression:** This is a straightforward approach. At every time step t of trading period, we use the past T_W data points (eg. 60 points, which corresponds to prices of both stocks of the past 60 trading days) for performing OLS regression to estimate the new α and β .

The downside of this approach is that a good value for T_W can be difficult to calibrate. An overly small window results in the estimations being noisy, while an overly large window causes the trading logic to respond slowly to trends in α and β . Hence, T_W is optimized by grid search and backtesting.

- **Time-varying Kalman Filter:** If we assume that α and β both follows a random walk stochastic process:

$$\alpha_{t+1} = \alpha_t + \epsilon_{\alpha,t}$$

$$\beta_{t+1} = \beta_t + \epsilon_{\beta,t}$$

Then, the Kalman Filter framework can be applied to estimate α and β dynamically [12]. The observation and state transition equations are:

$$\text{State transition equation: } \begin{bmatrix} \beta_{t+1} \\ \alpha_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_t \\ \alpha_t \end{bmatrix} + \begin{bmatrix} \epsilon_{\beta,t} \\ \epsilon_{\alpha,t} \end{bmatrix}$$

$$\text{Observation equation: } \log Y_t = \begin{bmatrix} 1 & \log X_t \end{bmatrix} \begin{bmatrix} \beta_t \\ \alpha_t \end{bmatrix} + \epsilon_t$$

where $\begin{bmatrix} \epsilon_{\beta,t} \\ \epsilon_{\alpha,t} \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix})$ and $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$

Note that the observation equation is equivalent to the linear model defined previously (see 2.1.2). The parameters σ_1^2 , σ_2^2 and $\sigma_{\epsilonpsilon}^2$ can be estimated using the Expectation-Maximization (EM) algorithm over a set of historical data.

2.1.5 Main Strategy - Reinforcement Learning (RL) Trading Agent

In addition to using traditional rule based trading logic, we also used RL to train a trading agent to initiate or exit trading positions as RL is a useful framework to model decision making problems. In the RL framework, the main components include an **agent**, which acts as the decision maker, and an **environment**, which is the subject the agent interacts with.

In this section, we will introduce Markov Decision Process (MDP), which is a basic model to decision making problems. Next, we will introduce Partially-Observable Markov Decision Process (POMDP), which is an extension to MDP. We will then describe our problem definition and how POMDP can model it. Finally, this section will end by describing the policy gradient method which enables the agent to learn from past experiences.

Markov Decision Process (MDP)

A process starts at time step $t = 0$ where the agent will observe the environment with initial **state** s_t . Then, the agent will choose an **action** among its predefined set of possible actions and respond to the environment. Subsequently, a response is immediately generated by the environment, known as the immediate **reward** r_t . The next state s_{t+1} is also passed to the agent simultaneously. This sequence of events will occur within a time step, and upon completion, the time step is updated and the process repeats itself until the agent receives a terminal state s_T . A process from an initial state s_0 to s_T is defined as an **episode**.

To define the goal of the agent, we first define the **return** R_k in some episode at time step k to be $R_k = \sum_{t=k}^{T-1} \gamma^t r_t$. Since our problem only consist of finite number of steps, we can set γ to be 1. The goal of the agent is to maximize the **expected initial return** $\mathbb{E}[R_0]$ over all possible episodes.

Overall, the interaction between the agent and the environment is formally modelled as a **Markov Decision Process (MDP)** which according to [11] is a tuple $\langle S, A, P_{sa}, R, \gamma \rangle$ where:

- S is a set of states.
- A is a set of actions.
- P_{sa} is the state transition probabilities over S given any $s \in S$ and any $a \in A$
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function. (We will use $R : S \times A \times S \rightarrow \mathbb{R}$ instead. The first dimension is the current state and the last dimension is the next state. This will much simplify the state definition in our problem.)
- $\gamma \in [0, 1]$ is the **discount factor** of rewards.

Note that the states in a MDP are Markov, meaning that given the present state, the future states are independent of the past states [15]. Similarly, the trading agent can be modeled by a **policy** π which defines a probability distribution over the action space given a state s .

Partially-Observable Markov Decision Process (POMDP)

In MDP, an important characteristic is that the environment state (which contains the key information for decision making) is directly observable by the agent. However, the environment state is usually hidden in most real world problems, and this is where Partially-Observable Markov Decision Process (POMDP) is

more suited to model these problems. In POMDP, the agent only observes **observation** o_t at each time step and the observation depends on the hidden environment state.

In POMDP, a policy is defined differently. According to [18], a policy is a probability distribution over the action space given the sufficient statistic of the **observed history** h_t at time step t . The observed history h_t is defined as the string $o_0, a_0, o_1, a_1, \dots, o_{t-1}, a_{t-1}, o_t$. This means that a model architecture that accepts variable length sequence input is needed. A common choice suggested by [18] would be **Recurrent Neural Network (RNN)**, in particular, the **Long Short-Term Memory (LSTM)**. Therefore, we can use an LSTM to model the policy.

Another interpretation could be that at each time step t the RNN summarize the observed history by a output vector. We could treat the output vector as an approximation of the environment hidden state. Then, the RNN becomes a state encoding model. The policy network would takes in the output vector from the state encoding model and output a probability distribution over the action space. We prefer this interpretation as it reduces the problem to MDP in some sense.

Problem Definition

We would like the agent to solve the simplest problem first which is to let the agent trade one spread at a time. Then, the environment is defined as the market of two given stocks and the portfolio state of the agent. The state at each time step t is the level of the spread, namely over-priced, under-priced, or the normal price level, together with the agent portfolio value. With the knowledge of the level of the spread, the agent can immediately choose one action from the action space which contains three possible actions, namely selling the spread, buying the spread, or having no position on the spread.

However, the level of the spread is a subjective and not only based on the current observed estimate spread value but maybe related to the spread value history. What the agent can directly observe at each time step t is its portfolio value, the log price of the two given stocks, and the spread value estimated by linear regression with rolling windows. The vector consists of all the observed values is called observation o_t . Because of one hidden information in the state which is the level of the spread, we need to model the problem using POMDP.

To summarize, the observation, state, and action in each time step are:

Observation space:	\mathbb{R}^{1+2+1} where the first dimension is the agent portfolio value, the second and third dimensions are the price of two given stocks, and the last dimension represents the spread of the normalized log price estimated by rolling window regression.
State space (partly unobserved):	\mathbb{R}^{1+3} where the first dimension is the agent portfolio value and the last 3 dimension together is the one-hot encoding of the classes in {"spread is over-priced", "spread is under-priced", "spread is in the normal price level"}.
Action space:	one-hot encoding of the classes in {"selling the spread", "buying the spread", "having no position on the spread"}.

Due to the partial observable issue, at each time step t we use the LSTM to summarize the observed history and hopfully output a vector that can approximate the environment state. However, since we can directly observe the portfolio value at each time step, we do not have to input portfolio value into the LSTM. Therefore, the input of the LSTM at each time step is $\begin{bmatrix} a_{t-1} & o_t \end{bmatrix}^T$ together with the hidden state from the

last time step. Note that one-hot encoding is used when we input an action into the neural network. However, we do need to concatenate one more dimension to the output of the LSTM to include the information of the agent portfolio value. That dimension is V_t at each time step t which is defined as the portfolio value at time step t divided by the initial portfolio value. Also, we can extend this slightly by adding a **Multilayer Perceptron (MLP)** after the output of the LSTM. This allows us to increase the model complexity in the event of underfitting.

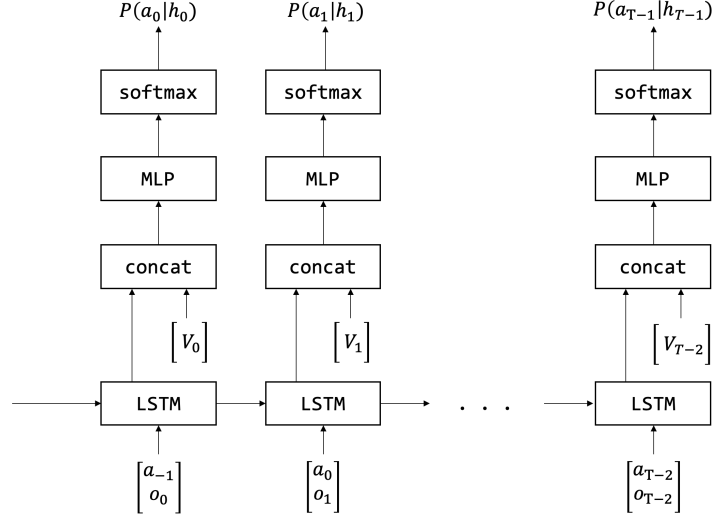


Figure 2: Recurrent Policy Network Model Architecture in our Problem Definition

Note that we added an additional a_{-1} to complete the input vector of the LSTM at time step 0. It is defined to be the “having no position on the spread” action.

After the agent samples an action and take it, the environment will give an immediate reward to the agent. In our problem the immediate reward at each time step t is:

$$r_t = PV_{t+1} - PV_t,$$

where PV_t is the portfolio value at time step t .

Policy Gradient in POMDP

The key method we used to allow the agent to learn from experience is the **Policy Gradient (PG)** method. In PG, we parameterize the policy by a parameter $\theta \in \mathbb{R}^m$ and define the objective function in terms of the parameterized policy. Finally, we perform gradient ascent to update the current θ and search for a better policy. The objective function in our problem is:

$$J(\theta) = \mathbb{E}[R_0] = \mathbb{E} \left[\sum_{t=0}^{T-1} r_t | \pi_\theta \right],$$

where r_t is the immediate reward received by the agent at time step t and T is the number of steps in one episode. Note that T is a fixed number in our problem. According to [18], the gradient of the objective

function with respect to the parameter θ is:

$$\nabla_{\theta} J \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | h_t^n) R_t^n,$$

where N is the number of episodes the agent played before performing a gradient update, h_t^n is the observed history in episode n at time step t , and R_t^n is the return in episode n from time step t onward. With this gradient approximation formula, we can follow the update rule:

$$\theta := \theta + \eta \nabla_{\theta} J,$$

where η is the learning rate.

2.2 Design - System Overview

2.2.1 Backtesting System for Trading Logic

Data Flow Pipeline

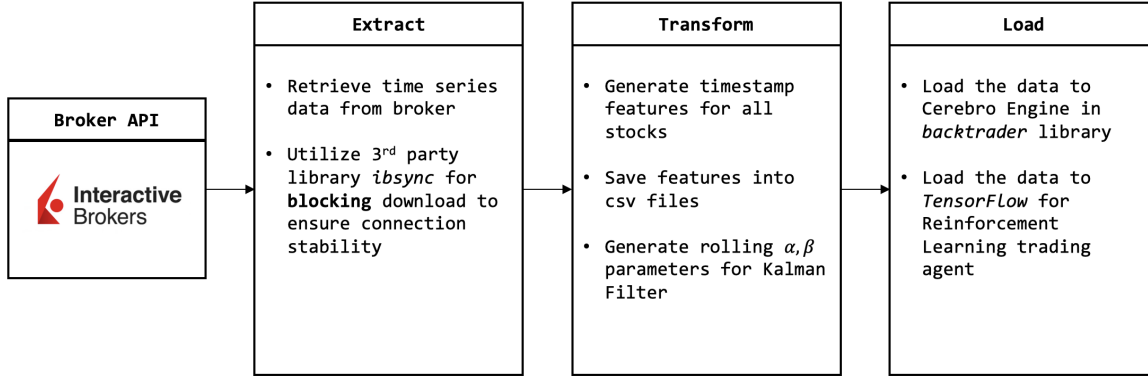


Figure 3: Preprocessing Data for Backtesting. We followed the ETL approach and downloaded, validated and processed around 10 years of daily stock price data available in the New York Stock Exchange (NYSE).

Baseline Strategies - Trading Logic Class Diagram

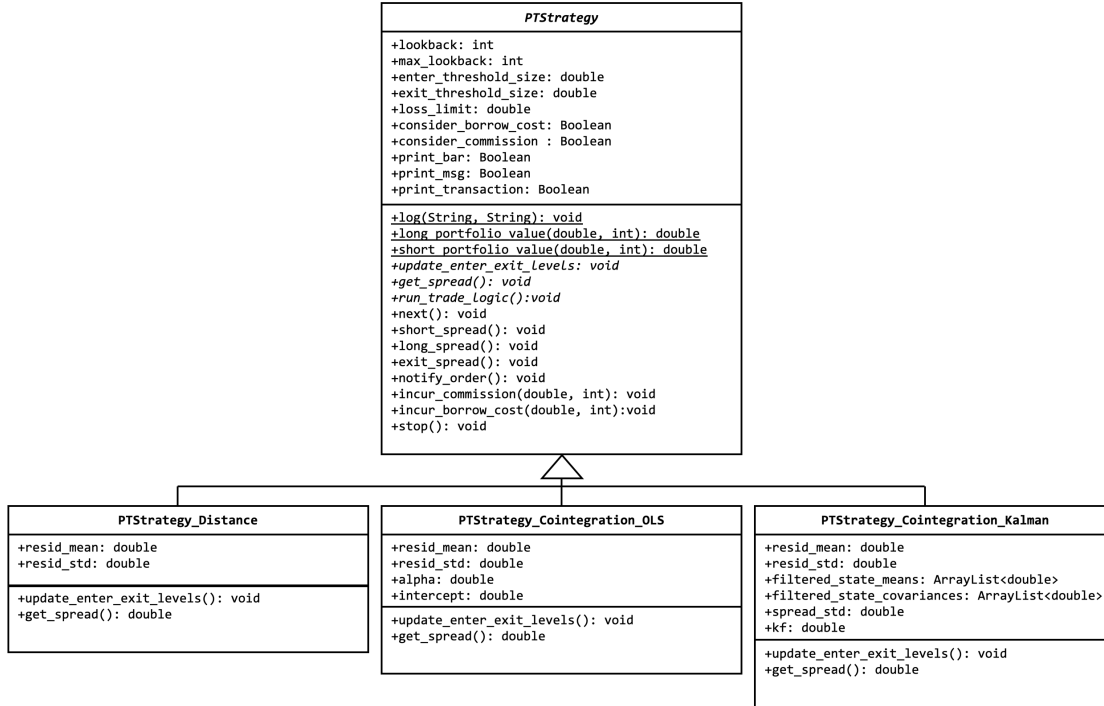


Figure 4: Abstract Base Class Implementation for Baseline Strategies

2.2.2 Web Application

For the second part of this project, a visualization tool was created to with the main goal to enable us to understand the actions of implemented algorithms by visual analysis. Consequently, the targeted audience of this product is intended for professional traders, risk managers, and quantitative researchers with sufficient background knowledge on both trading and reinforcement learning.

System Requirements Specifications and User Workflow

Feature	Description
Set backtest params	-
Backtest a given pair using Distance method	Obtain table of relevant performance statistics
Backtest a given pair using Cointegration (rolling OLS) method	Obtain table of relevant performance statistics
Backtest a given pair using Cointegration (Kalman Filter) method	Obtain table of relevant performance statistics
Backtest a given pair using Reinforcement Learning trading agent method	Obtain table of relevant performance statistics
Show stock prices of a given pair	Eyeball comparison of spread
Show how the spread changes over time	Identify when the spread diverges from the mean to form a trade opportunity
Indicate specific points in time when a trade action is made	Label the action, and the amounts a stock is bought or sold

Table 1: System Requirements Specifications



Figure 5: Design of UI for Visual Analysis of Trading Strategies



Figure 6: Special Features of the UI

2.3 Implementation

2.3.1 Tools and Libraries Used - Trading Logic and Backtesting System

Backtrader

We used this Python Framework for backtesting the distance method and cointegration method in the baseline strategies. We have chosen this library mainly because it can load data directly from our broker. In addition, it supports calculation of “industry standard” performance metrics (eg. Sharpe Ratio, Maximum Drawdown, Overall Return), and the API is well documented with an active community of users. The key features used are:

- Cerebro Engine Class - This is the “cornerstone class” in the **backtrader** library because it gathers all inputs (Data Feeds), actors (Strategies), spectators (Observers), critics (Analyzers) and documenters (Writers) for backtesting.
- Strategy Class: **Backtrader** provided a basic “backbone” **Strategy** class for constructing a Trading Strategy, which consists of several “empty” virtual functions such as **next()** and **stop()**. We implemented a Abstract Base Class named **PTStrategy** (see Figure 4) which inherits from this **Strategy** class.
- Plotting: Backtrader has built-in plotting functions developed using **matplotlib** to help us visualize the performance of a strategy.

TensorFlow

We used this machine learning library for implementing the RL trading agent with the Policy Gradient method. The key feature that was heavily utilized is the “**Eager Execution**” feature because it enables us to dynamically build the computational graph, which is crucial in our model. This is because the input sequence of the LSTM network is generated at run-time; the one-hot action vector at each time step t is unknown until sampling from the probability distribution output from the policy network at the last time step $t - 1$.

Intel AI DevCloud

For training the policy network, we made use of the Intel AI DevCloud which provides higher computational power than our laptops, and thus speeding up the training process.

2.3.2 Tools and Libraries Used - Web Application

Bokeh

We used this Python library is used as the primary tool for visualizing data, actions and insights generated from our trading strategies as it is a interactive vizualization library in python with rich features. The main feature of **Bokeh** that was heavily utilized is the **curdoc.addRoot()**. function, which allows the backend to run the trading strategy in python script and generate a Javascript file for seamless integration with the web application. Another function that was heavily utilized is the **onChange()** function, which tracks the

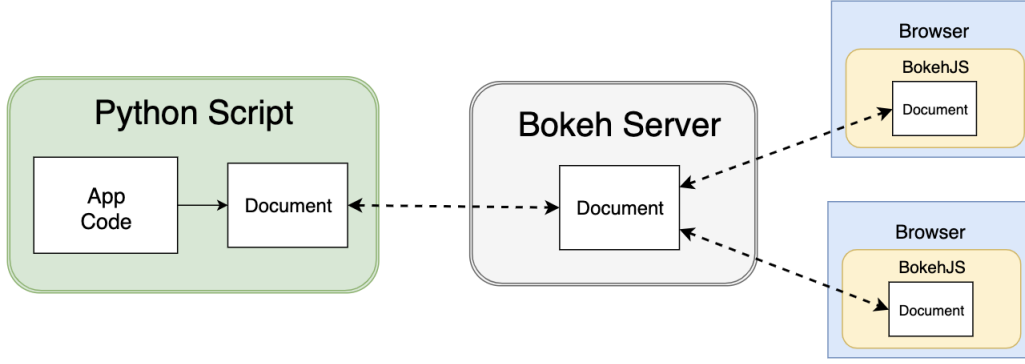


Figure 7: Integration of Bokeh application to UI

changes in parameters from the web application. This allows the backend to register changes from the user's perspective such that the backtester can run with the user's desired input.

Flask and Jinja2

For this project Jinja2, was used to reconcile the charts generated from bokeh `bokeh` and placed into specific location of our HTML backtesting webpage. Jinja2 is a modern templating language for Python used to create HTML and returned to a user via a HTTP request.

2.3.3 Software Engineering Practices

Software Development Lifecycle

For the trading strategies component, we used a mix of **spiral development model** and **waterfall development** as it is most suitable. This section of code can be broken down into trading-specific functions (baseline strategy classes and RL Trading Agent) and peripheral support functions (load data, backtesting, and generate plots). Therefore, spiral development model is used for the former because requirements are unclear and strategy specific (e.g. spread model), while and waterfall development for the latter since the requirements are straightforward. In addition, spiral development model was a natural development model for trading specific functions since this development model excels when releases are required to be frequent, and creation of prototype is applicable and the requirements were unclear. Frequent backtesting was required to test the trading agent and refine it.

For the web application, we mainly used waterfall development method because the system requirements was well defined (see Table 1 above) and the size of this system is small.

Logging

For consistency and reproducibility of results, we ensured that the experiments backtesting, grid search, and statistics were recorded comprehensively. For backtesting of baseline strategies, the following information are logged into a `log` file whenever a new test is performed:

- Git commit number: To ensure that we can trace back to the correct **version** of code that produced a specific result

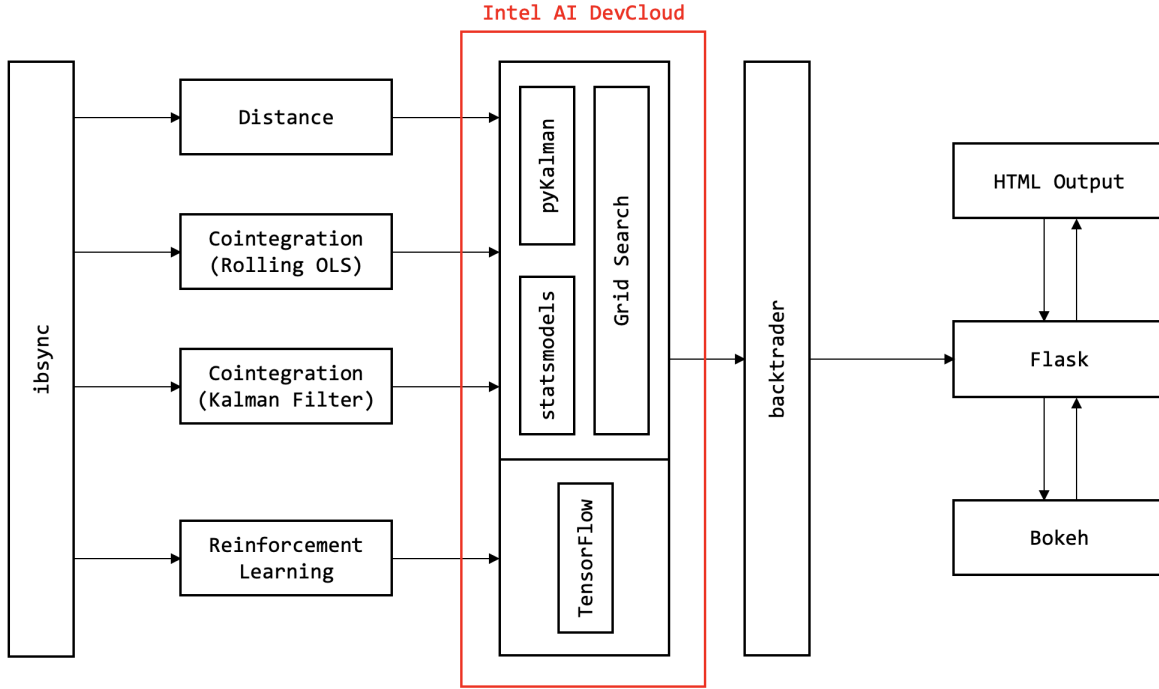


Figure 8: Unified view of system components

- Backtesting parameters: These parameters set the scope of the backtest (timeline, universe of stock). Parameters include `start_date`, `end_date`, `length_of_pair_selection_period`, and `stock_sector`
- Strategy parameters: These parameters were used for grid search and were represented by a list. Parameters include `max_lookback`, `enter_threshold`, `exit_threshold`, and `cut_loss_limit`
- Assumptions: Whether or not transaction costs and borrowing costs were taken into account

The logging procedure has the following steps:

1. Backtesting parameters is fixed
2. A set of strategy parameters is passed to the system
3. The system generates a combination of strategy parameters. Then,
 - The performance of **each pair** under the same set of parameters were recorded in a “child” csv file. The child file is named with a randomly generated UUID to avoid name collision.
 - Once testing is done for all pairs, the **average** performance of the batch is recorded in the “parent” csv file, together with the combination of strategy parameters.
4. Repeat step 3 for all combination of strategy parameters.

During training of RL agent, the model architecture and model weights were recorded and saved in disk whenever the model was found to be the best during training. This can ensure that the best model we found were recorded and can be retrieved in the future. We also recorded the training history of the agent together with the hyperparameters used. This facilitated the manual hyperparameter tuning process.















 01c56347-09a2-41e6-9ee8-d38e8a9f2693.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 2aaf6957-082a-4847-b3a5-60af16a2426a.csv	20/1/2019 4:27 PM	Microsoft Excel Co...	5 KB
 36ab9d7d-3786-451d-9b46-41362cf2c95f.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 53adf4ce-46d3-4230-9d26-18436236e097.csv	20/1/2019 4:29 PM	Microsoft Excel Co...	5 KB
 65c1dc5b-6642-4a0f-89be-91af1596e3e4.csv	20/1/2019 4:27 PM	Microsoft Excel Co...	5 KB
 8076a8fe-c089-47fe-8567-8be91aa7ca27.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 a9763435-1b37-4253-8c6b-c621f029b7f2.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 c8ea5028-8d0c-4bcb-bddb-83d1c28663d4.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 cb72e072-c686-418c-be66-f1cbbbfc6f60.csv	20/1/2019 4:29 PM	Microsoft Excel Co...	5 KB
 config.json	20/1/2019 4:26 PM	JSON File	1 KB
 d6c70f00-8606-4bdf-aafa-5dacad48be8e.csv	20/1/2019 4:27 PM	Microsoft Excel Co...	5 KB
 e81eec9c-8edc-4d44-bfe3-4ff57276819a.csv	20/1/2019 4:28 PM	Microsoft Excel Co...	5 KB
 f5c64b9f-a9d7-470a-913f-bd79c4de066e.csv	20/1/2019 4:27 PM	Microsoft Excel Co...	5 KB
 summary.csv	20/1/2019 4:29 PM	Microsoft Excel Co...	2 KB

Figure 9: Files Generated During Backtesting

```

2019-04-16 01:40:00,290 INFO - __main__ - Running pair 8/133
2019-04-16 01:40:00,549 INFO - ptstrategy - -
2019-04-16 01:40:00,551 INFO - __main__ - Performance of this pair: {'pair': 'EPAM-PANW', 'sharperatio': 30.88468448022882, 'returnstd': 18758.95593073902, 's
tartcash': 1000000, 'endcash': 1478566.5378974264, 'profit': 0.4785665378974264}
2019-04-16 01:40:00,552 INFO - __main__ - Running pair 9/133
2019-04-16 01:40:00,838 INFO - ptstrategy - -
2019-04-16 01:40:00,842 INFO - __main__ - Performance of this pair: {'pair': 'ASGN-FDS', 'sharperatio': 5.563442801098177, 'returnstd': 11627.40635847032, 'st
artcash': 1000000, 'endcash': 1299698.0355424597, 'profit': 0.2996980355424597}
2019-04-16 01:40:00,845 INFO - __main__ - Running pair 10/133
2019-04-16 01:40:01,091 INFO - ptstrategy - -
2019-04-16 01:40:01,095 INFO - __main__ - Performance of this pair: {'pair': 'AL-FDS', 'sharperatio': -16.529565296663304, 'returnstd': 9980.790007193382, 'st
artcash': 1000000, 'endcash': 800010.1210356123, 'profit': -0.1999898789643877}
2019-04-16 01:40:01,097 INFO - __main__ - Running pair 11/133
2019-04-16 01:40:01,473 INFO - ptstrategy - -
2019-04-16 01:40:01,476 INFO - __main__ - Performance of this pair: {'pair': 'FDS-PAYC', 'sharperatio': 7.078459775384389, 'returnstd': 20421.495376070277, 's
tartcash': 1000000, 'endcash': 2000026.403723224, 'profit': 1.000026403723224}
2019-04-16 01:40:01,478 INFO - __main__ - Running pair 12/133
2019-04-16 01:40:01,681 INFO - ptstrategy - -
2019-04-16 01:40:01,684 INFO - __main__ - Performance of this pair: {'pair': 'FDS-ITW', 'sharperatio': -1.2482493435388984, 'returnstd': 8094.140407971345, 's
tartcash': 1000000, 'endcash': 850531.1890217431, 'profit': -0.14946881097825687}
2019-04-16 01:40:01,686 INFO - __main__ - Running pair 13/133
2019-04-16 01:40:01,871 INFO - ptstrategy - -
2019-04-16 01:40:01,873 INFO - __main__ - Performance of this pair: {'pair': 'ELLI-LXFT', 'sharperatio': 0.6347201961143523, 'returnstd': 26974.525090036877,
'startcash': 1000000, 'endcash': 1392360.9415558677, 'profit': 0.39236094155586765}
2019-04-16 01:40:01,876 INFO - __main__ - Running pair 14/133
2019-04-16 01:40:02,234 INFO - ptstrategy - -
2019-04-16 01:40:02,237 INFO - __main__ - Performance of this pair: {'pair': 'NSP-STM', 'sharperatio': -0.6216624796359101, 'returnstd': 18475.875830757537, '
startcash': 1000000, 'endcash': 786656.4381350664, 'profit': -0.21334356186493358}
2019-04-16 01:40:02,239 INFO - __main__ - Running pair 15/133
2019-04-16 01:40:02,425 INFO - ptstrategy - -
2019-04-16 01:40:02,427 INFO - __main__ - Performance of this pair: {'pair': 'ELLI-NSP', 'sharperatio': -0.23808217053879352, 'returnstd': 22572.672112224966,
'startcash': 1000000, 'endcash': 1005066.717807885, 'profit': 0.005066717807885026}
2019-04-16 01:40:02,428 INFO - __main__ - Running pair 16/133
2019-04-16 01:40:02,725 INFO - ptstrategy - -
2019-04-16 01:40:02,727 INFO - __main__ - Performance of this pair: {'pair': 'ELLI-PANW', 'sharperatio': 1.2699599314416008, 'returnstd': 27576.815246514532,
'startcash': 1000000, 'endcash': 1427885.8651378388, 'profit': 0.4278858651378388}

```

Figure 10: Log file generated during Grid Search and Backtesting

```

(traders_nlp) C:\Users\emara\coding_workspace\statistical-arbitrage-private-18-19>git log --all --dec
* 128c147 (HEAD -> develop, origin/develop) Merge branch 'develop' of https://github.com/wyongbd/s
|
| * bcb1ea3 Merge branch 'feature/12-2-2019/backtest-progress-report' into develop
| |
| * 9cc2feb edited coint kalman strat
| |
| * 6443de7 Merge branch 'feature/31-1-2019/grid-search' into develop
| |
| * 47a41af edited distance, coint and cointkalman
| * e4f762a save result
| |
| * 1b1abe5 (read-code) Merge branch 'develop' into read-code
| |
| |
| * 3102968 (public-origin/develop) added scatter plot for showing correlation.
| * ed3c803 another training.
| * ded4d22 changed learning rate. it can be smaller and the batch_size can increase.
| |
| * 7e85bb1 Merge branch 'develop' of https://github.com/wyongbd/statistical-arbitrage-private-18-
| |
| * 861182c Merge branch 'feature/29-1-2019/backtest-kalman-for-real' into develop
| |
| * 235397d kalman grid search not bad
| * f522dbf trained for all pairs
| * e0ab654 save progress
| |
| ...skipping...
* 128c147 (HEAD -> develop, origin/develop) Merge branch 'develop' of https://github.com/wyongbd/s
|
| * bcb1ea3 Merge branch 'feature/12-2-2019/backtest-progress-report' into develop
| |
| * 9cc2feb edited coint kalman strat
| |
| * 6443de7 Merge branch 'feature/31-1-2019/grid-search' into develop
| |
| * 47a41af edited distance, coint and cointkalman
| * e4f762a save result
| |

```

Figure 11: This image is a screenshot of our Git branch history, illustrating how we adhered to the GitFlow model.

Version Control using Git

For this project, we used Git for doing version control and our repository is hosted privately on GitHub. As for the development model, we adopted the GitFlow model created by Vincent Driessen [4].

By following the GitFlow model, there were two main branches in our repository, namely **master** and **develop**. The **master** branch always reflects a production-ready state of our code. The **develop** branch is the branch which reflects a state with the latest developed development changes. Once **develop** has reached a stable state, it may be merged into **master**. Each new individual feature is implemented by first branching out from **develop** and then merged back when it is completed and stable.

2.4 Testing

2.4.1 Testing of Trading Logic

In our FYP, the main purpose of testing is to ensure:

- The data being used and analysed is valid
- Feature generation (calculation of rolling α , β in cointegration model) is mathematically correct

Validation of Downloaded Data

To ensure that the data downloaded from Interactive Brokers is valid, we iterated through each row in the dataset and performed the following checks:

1. Basic sanity check: Is the stock price greater than 500,000 (the most expensive stock in 2019 is Berkshire Hathaway Inc, priced at \$296,900 per share)? Are there negative values?
2. Invalid values: Are there null or negative values?
3. Cross checking (only for the most recent 2 years of historical data): Are the values consistent as those retrieved from other sources (Bloomberg terminal, Yahoo Finance).

To handle problems 1 and 2, we either provided a new value based on other data sources, or performed a simple quadratic interpolation provided by the `scipy` Python package.

Validation of Feature Generation

To ensure that our features were generated correctly, we picked the starting, middle and ending points of our dataset (chronologically) and calculated the rolling α , β manually and compared them with the values computed by our algorithm.

2.5 Evaluation

2.5.1 Experiment Setup

To evaluate the performance of our RL trading agent, we conducted experiments to compare its performance with other baseline strategies. To ensure consistency and comparability of the results, the experiment has the following constraints:

1. **Stock Universe:** All technology stocks available in the New York Stock Exchange (NYSE) that have no missing data point from 2015-01-01 to 2019-01-03 were selected. This amounts to 116 stocks, which provides $\binom{116}{2} = 6670$ possible pairs.
2. **Experiment Timeline:** The time series data used is from 2015-01-01 to 2019-01-03. As each strategy has different data use requirements, the time series was sliced into periods as shown in diagram 12 below.

Label	Comparison Period (200 Trading days)
Period 0	2015-03-19 to 2015-12-31
Period 1	2016-03-18 to 2016-12-30
Period 2	2017-03-20 to 2018-01-02
Period 3	2018-03-20 to 2019-01-03

Table 2: Labels for the 4 main comparison periods used in our experiment setup

3. Number of Pairs Tested:

- **Distance Method:** Test on all 6670 pairs; pair selection not needed.
- **Cointegration Method:** Due to the stationarity requirement of the spread (see 2.1.2), we only backtest pairs whose spread passes the stationarity test during the **pair selection stage**. This is applied to both the Rolling OLS and Kalman Filter variants of cointegration method.
- **RL Trading Agent:** Test on all 6670 pairs; pair selection not needed.

4. Trade Logic:

- **Distance Method:** Uses trading logic as specified in Algorithm 1.
- **Cointegration Method:** Uses trading logic as specified in Algorithm 1. This is applied to both the Rolling OLS and Kalman Filter variants of cointegration method.
- **RL Trading Agent:** Agent performs own trading decisions as specified in the action space.

5. Parameter Optimization:

- **Distance Method:** Grid Search was applied to obtain performance of the best parameters.
- **Cointegration Method:** Grid Search was applied to obtain performance of the best parameters. This is applied to both the Rolling OLS and Kalman Filter variants of cointegration method.
- **RL Trading Agent:** We tried different neural network parameters such as different dimensions of hidden layers in LSTM and MLP.

Details of parameters used can be found in Appendix C.

6. **Day Count Convention:** The time series used is for **trading days** only. Each year is assumed to have 252 trading days.
7. **Training Data in RL:** When we evaluate the RL Trading Agent in period t , we will train the agent using data from period $0, \dots, t - 1$ where t from 1 to 3. (See Figure 12 for illustration)
8. **Experiments in RL:** As we defined the possible training data in RL above, we have 3 possible experiments in RL agent training.

Experiment	Training data	Testing data
1	Period 0	Period 1
2	Period 0, 1	Period 2
3	Period 0, 1, 2	Period 3

Table 3: Experimental Setup in RL

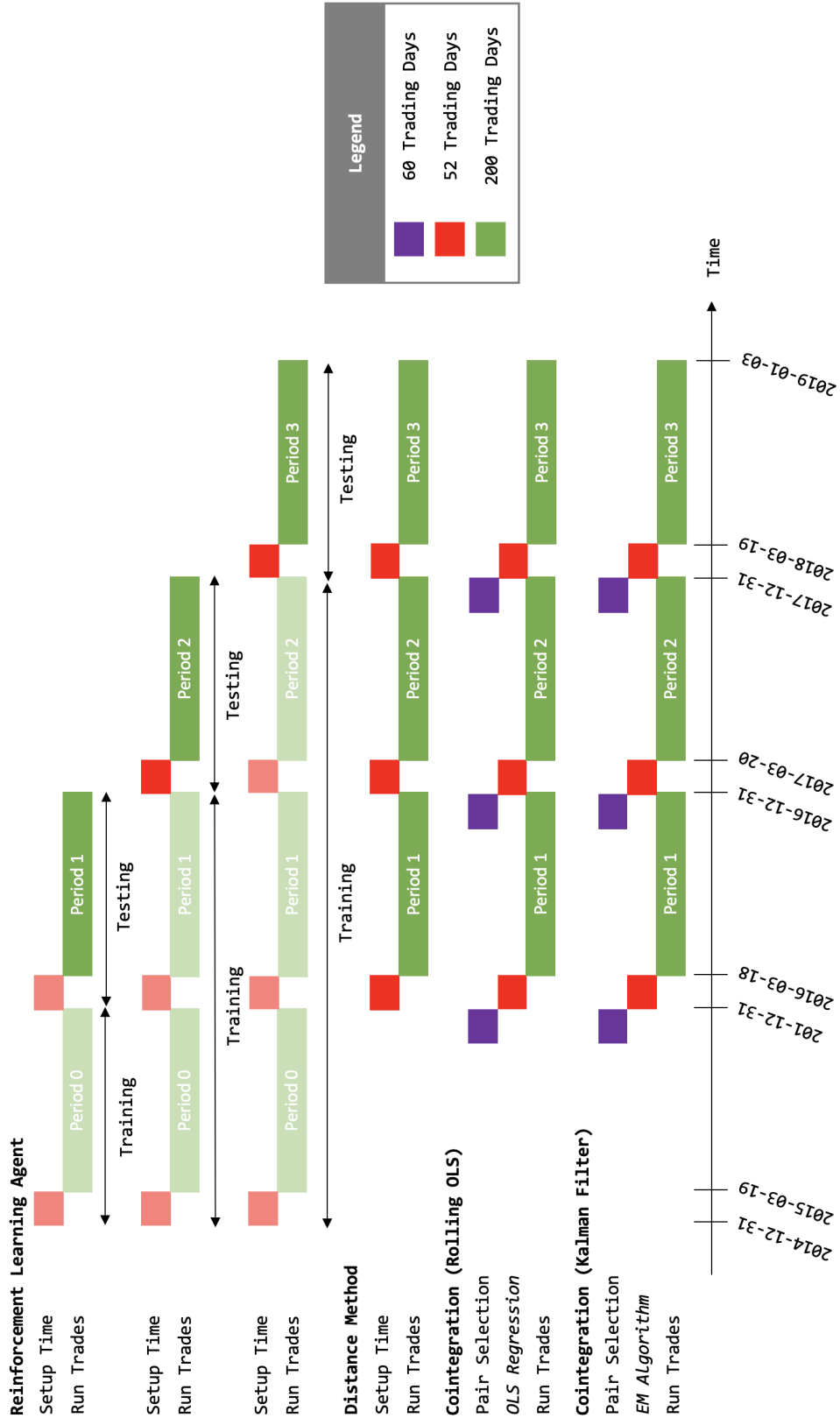


Figure 12: Timeline of Backtesting

2.5.2 Performance of Trading Strategies

Empirical Results of Different Methods

In total we have three testing periods, period 1, 2, and 3. In each period, we ran the backtesting of each method and generated the plots of return distribution of all tested pairs. Based on the distribution plots below, the results suggest that our proposed RL trading agent outperforms the existing baseline logic. In all 3 testing periods, the RL trading agent achieves more than **30% overall return** on average, which is much higher than the grid search optimized baseline strategies (Distance, Cointegration with rolling OLS, Cointegration with Kalman).

The following distributions represent the density of the returns of a given method. In simpler terms, the area under curve represents percentage of tested pairs. Empirical results show that RL Trading Agent is the best performing method because it has the highest proportion of area under the curve with positive returns. Additionally, these densities have in very highly positive returns as they are distributed more towards the right.

The RL agent model architecture and grid search results of all baseline methods can be found in Appendix C.

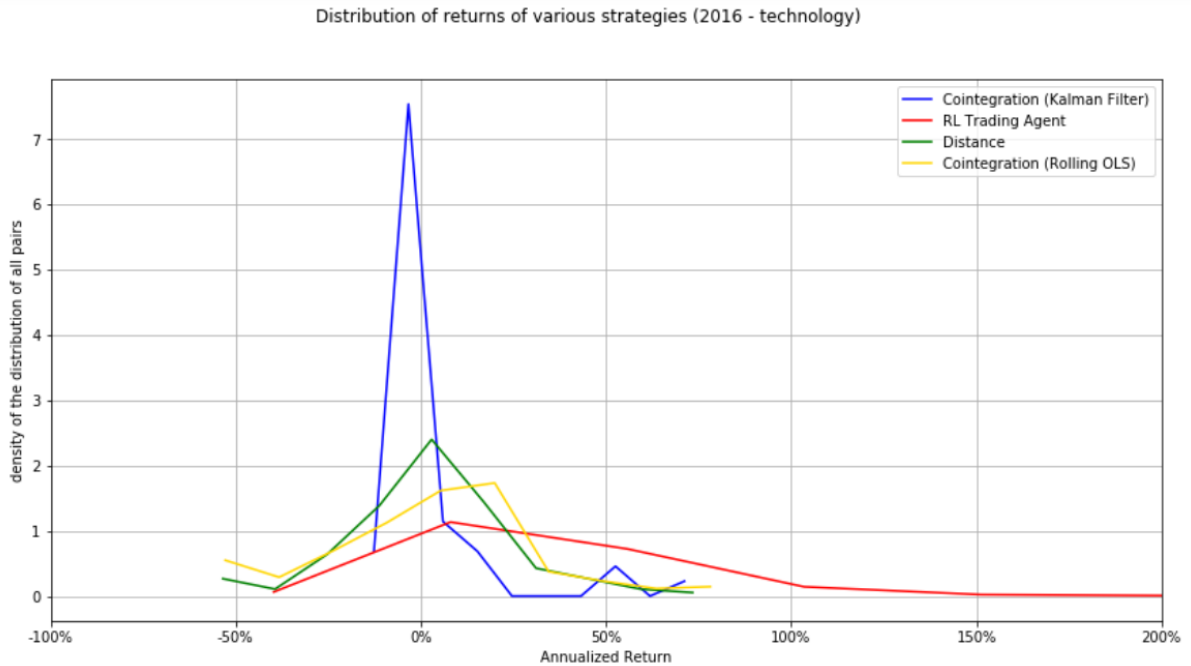


Figure 13: Performance of all strategies from **2016-03-18** to **2016-12-30** on **NYSE technology stocks**. The mean returns of Cointegration (Kalman Filter), Distance, Cointegration (Rolling OLS), Reinforcement Learning Agent are **4.28%**, **2.86%**, **3.21%** and **32.69%** respectively.

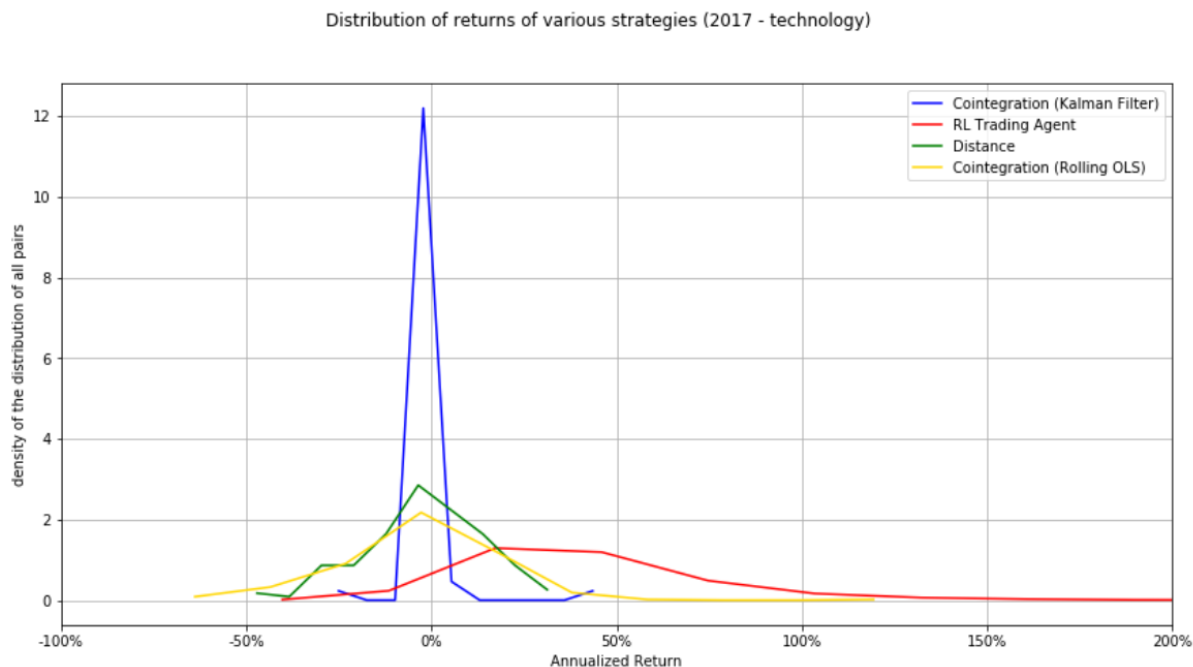


Figure 14: Performance of all strategies from **2017-03-20** to **2018-01-02** on **NYSE technology stocks**. The mean returns of Cointegration (Kalman Filter), Distance, Cointegration (Rolling OLS), Reinforcement Learning Agent are **0.46%**, **-1.79%**, **-3.34%** and **40.83%** respectively.

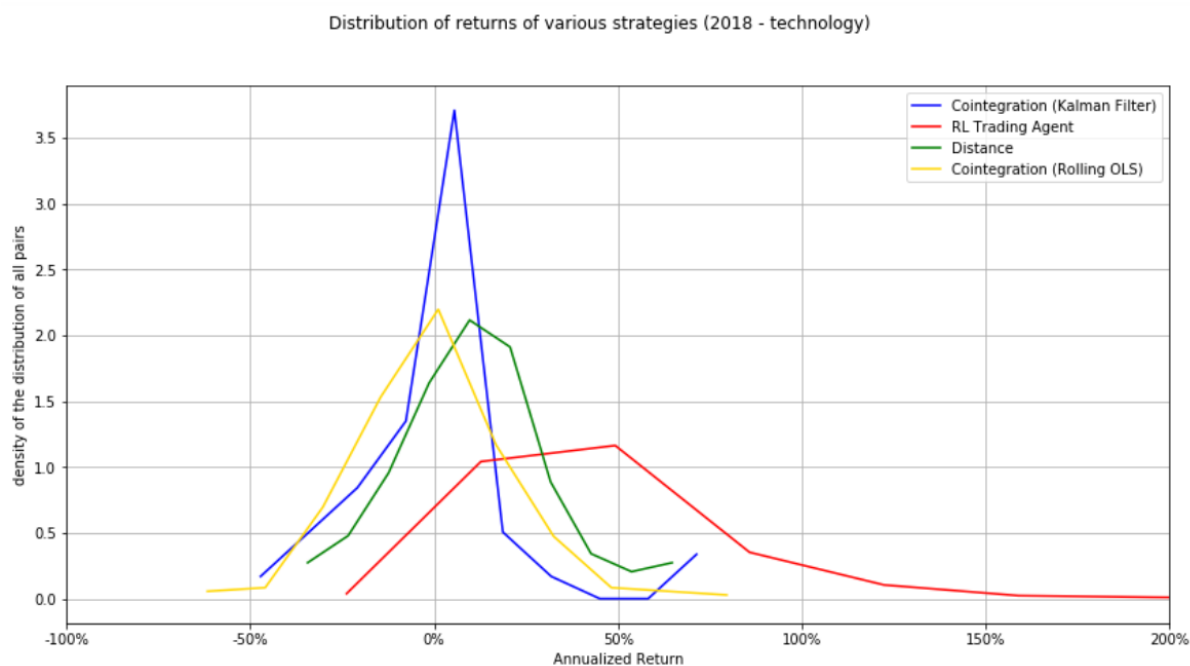


Figure 15: Performance of all strategies from **2018-03-20** to **2019-01-03** on **NYSE technology stocks**. The mean returns of Cointegration (Kalman Filter), Distance, Cointegration (Rolling OLS), Reinforcement Learning Agent are **0.26%**, **10.48%**, **-0.59%** and **43.78%** respectively.

Comparison of RL Trading Agent vs Market Benchmarks

In addition to outperforming baseline methods, the RL trading agent also outperforms the simple buy-and-hold strategy of market indices.

We have included the following market indices in the tables below:

- PSE (NYSE Arca Tech 100 Index) is the technology sector index for NYSE. It is represented by the top 100 market cap technology stocks.
- NYE (NYSE Energy Index) is the energy sector index for NYSE. It is represented by the energy listed stocks.
- SPX (S&P 500 Index) is the market index for US Equities. It is represented by the top 500 market cap stocks listed in NYSE, NASDAQ, and CBOE Exchanges.

Strategy/Portfolio	Period 1	Period 2	Period 3
Distance	+2.86%	-1.79%	+10.48%
Cointegration (Kalman Filter)	+4.28%	+0.46%	+0.26%
Cointegration (Rolling OLS)	+3.21%	-3.34%	-0.59%
Reinforcement Learning Agent	+32.69%	+40.83%	+43.78%
S&P 500 Index	+9.23%	+12.65%	-7.60%
PSE Index	+14.80%	+18.74%	-7.76%

Table 4: **Backtested return (profit)** of Distance, Cointegration (Kalman Filter), Cointegration (Rolling OLS) and Reinforcement Learning on **NYSE Technology stocks** over Period 1, Period 2 and Period 3.

Strategy/Portfolio	Period 1	Period 2	Period 3
Reinforcement Learning Agent	+53.80%	+69.60%	+46.02%
S&P 500 Index	+9.23%	+12.65%	-7.60%
NYE Index	+2.69%	-5.15%	-2.58%

Table 5: **Backtested return (profit)** of Reinforcement Learning Agent on **NYSE Energy stocks** over Period 1, Period 2 and Period 3.

Reinforcement Learning Training Progression

During the training of a RL trading agent, we have saved some models. After training, we retrieved those models and evaluated them on the corresponding period. The plots in this section is intended to show what the agent has learnt during the training.

For the ease of visualization, we only selected 4 models during the training progress. Note that model 0 is the random agent in the beginning of the training. Model 3 is the trained agent in the end of the training. Model 1 and 2 are the trained agent in the middle of training; model 1 was saved earlier than model 2.

In the plots below, we would see that the return distributions of all pairs shifted to the right as training went on. This means that the agent learnt to improve its overall trading performance.

Note that in each experiment, the model with the same index in the training data plot and the testing data plot are the same.

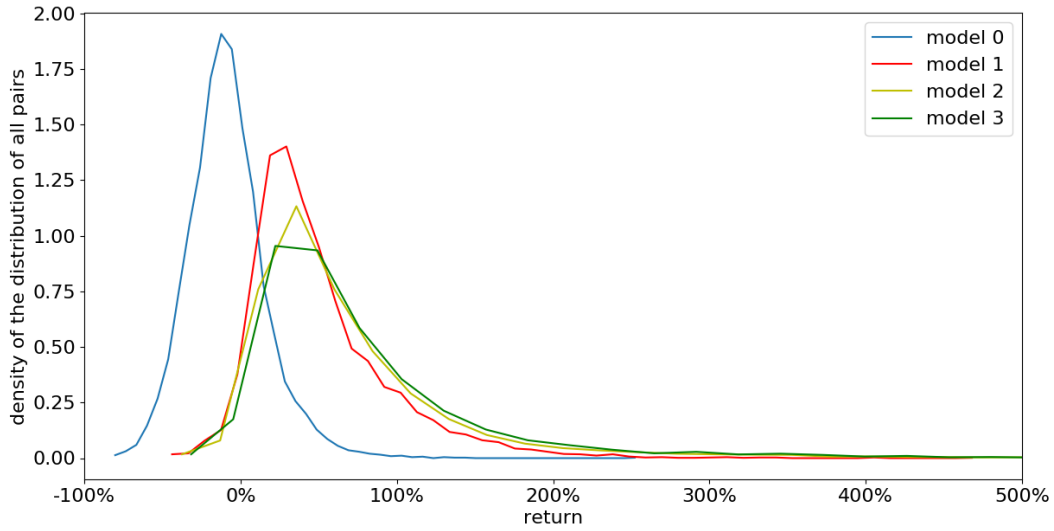


Figure 16: RL agent performance on **training data** (period 0) in the **first experiment** (see Table 3) as training goes on.

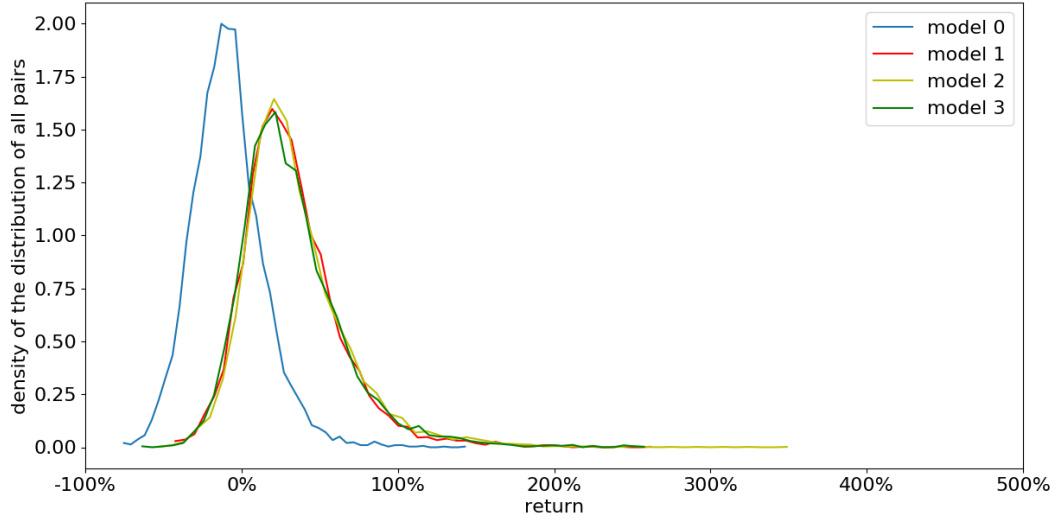


Figure 17: RL agent performance on **testing data** (period 1) in **the first experiment** (see Table 3) as training goes on. The models correspond to those in Figure 16.

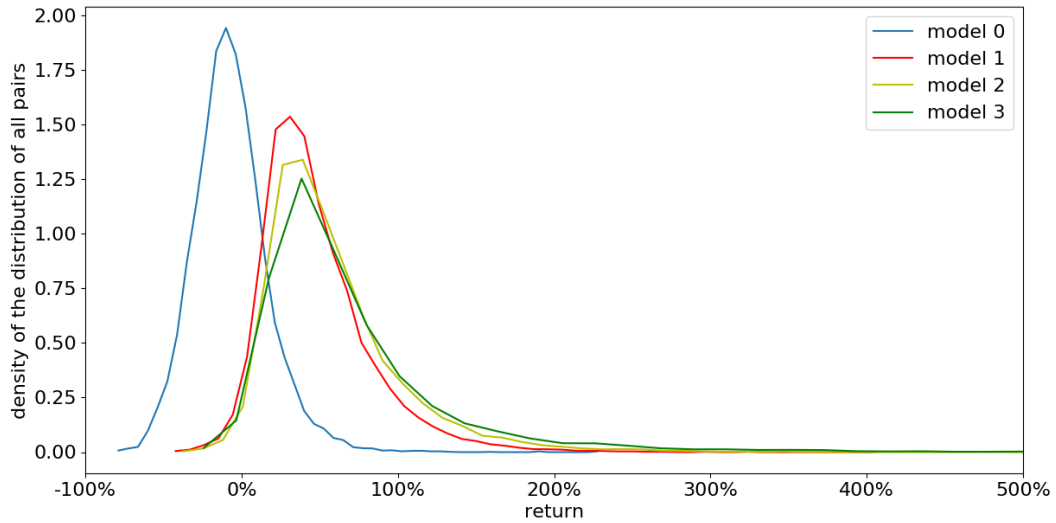


Figure 18: RL agent performance on **training data** (period 0, 1) in **the second experiment** (see Table 3) as training goes on.

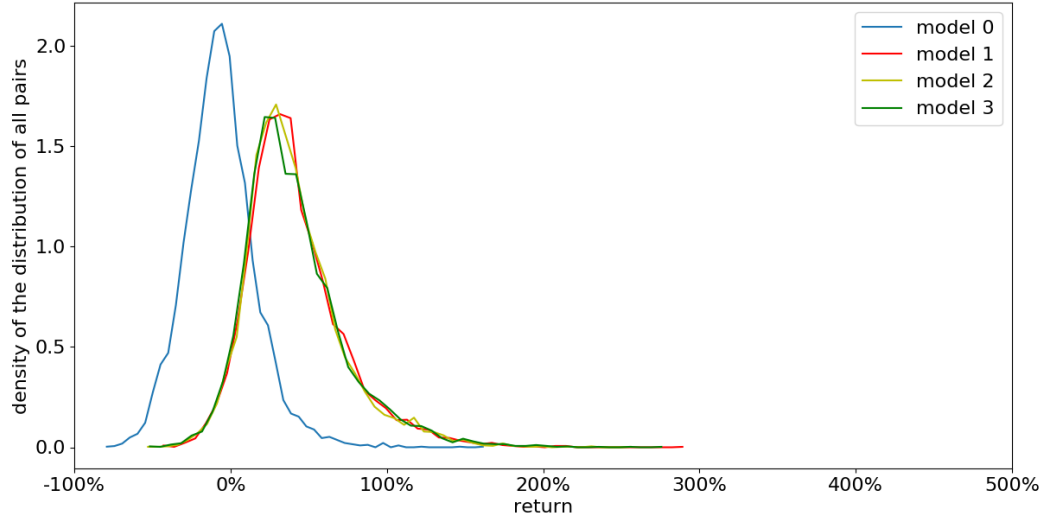


Figure 19: RL agent performance on **testing data** (period 2) in **the second experiment** (see Table 3) as training goes on. The models correspond to those in Figure 18.

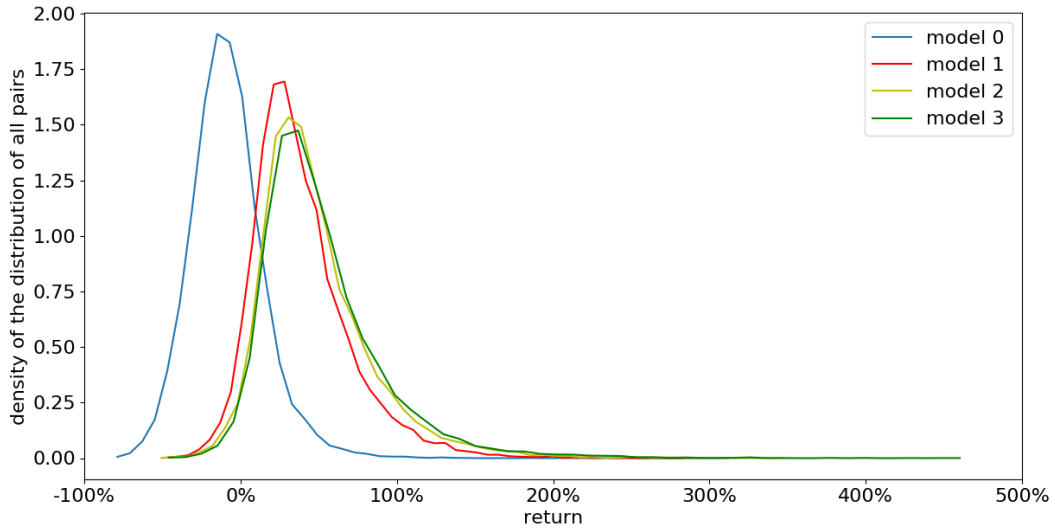


Figure 20: RL agent performance on **training data** (period 0, 1, 2) in **the third experiment** (see Table 3) as training goes on.

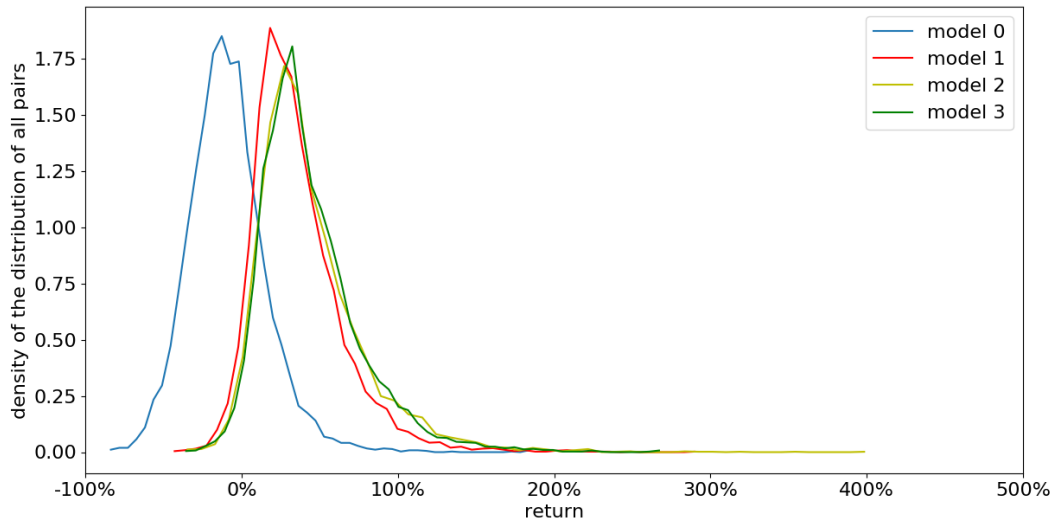


Figure 21: RL agent performance on **testing data** (period 3) in **the third experiment** (see Table 3) as training goes on. The models correspond to those in Figure 20.

Performance of Pre-trained Agent on Energy Sector

After we trained a RL agent using the technology stocks available in NYSE, a similar test was conducted using stocks from the energy sector using the same periods. The total number of stocks from energy sector is 155. The number of all possible pairs is $\binom{155}{2} = 11935$.

The plots below show the performance on energy stocks of the trading agent that was trained with technology stocks in the three experiments.

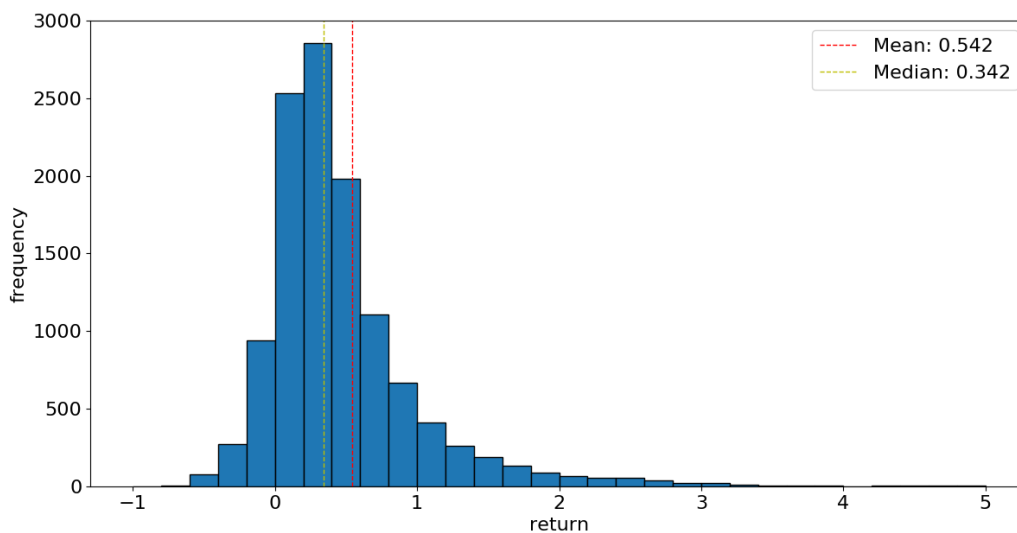


Figure 22: The return distribution of pre-trained RL trading agent, backtested on energy stocks in period 1

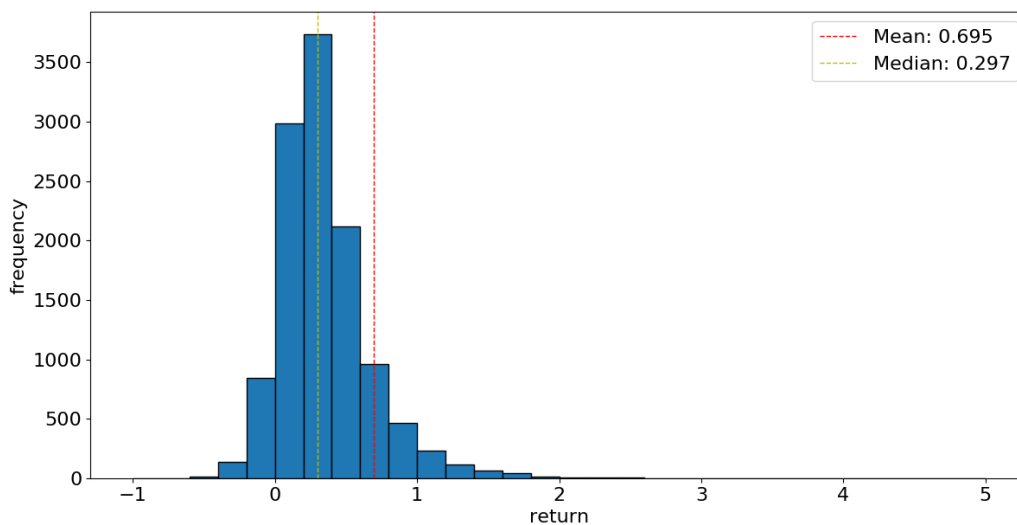


Figure 23: The return distribution of pre-trained RL trading agent, backtested on energy stocks in period 2

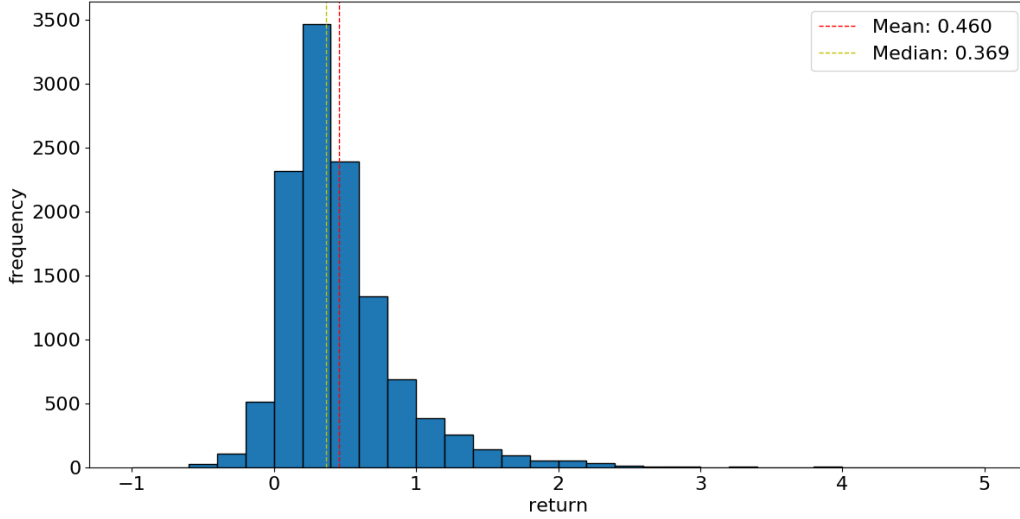


Figure 24: The return distribution of pre-trained RL trading agent, backtested on energy stocks in period 3

Performance of Pre-trained Agent on Other Asset Class

To further validate the performance of the RL Trading Agent, we have chosen a few indices widely used in the fixed income asset class and some global macro instruments to be tested, and they are as follows:

1. **Credit Indices:** Aside from equity, the most similar asset class are the credit assets as they are directly above the equity in a company's capital structure, therefore, the factors influencing its prices are still somewhat company-specific. For this asset class, we have chosen iTraxx CDS indices, which measure a basket of company credit worthiness analogously to how S&P500 index measures the performance of a basket of equities. We included indices from Europe, and North America.
2. **Interest Rates:** A common interest rate trading strategy is to bet that the rates of two different tenors will steepen (analogously, spread widen) or flatten (tighten). This is a natural extension of pairs trading in the interest rate market. We included 2 year, 5 year, 10 year, and 30 year (US Treasury Bonds), and USD Libor.
3. **Currencies and Commodities:** On the global macro setup, the common assets are the USD Currency Index Strength index (DXY), Crude Oil price index.

In total this gave 8 assets. The number of all possible pairs is $\binom{8}{2} = 28$.

The plots below show the performance on other assets of the trading agent that was trained with technology stocks in the three experiments.

Note: The use of RL Trading Agent is to determine to optimal time to bet the spread will widen or tighten. Exact financial instruments used to express these views are not considered in this study.

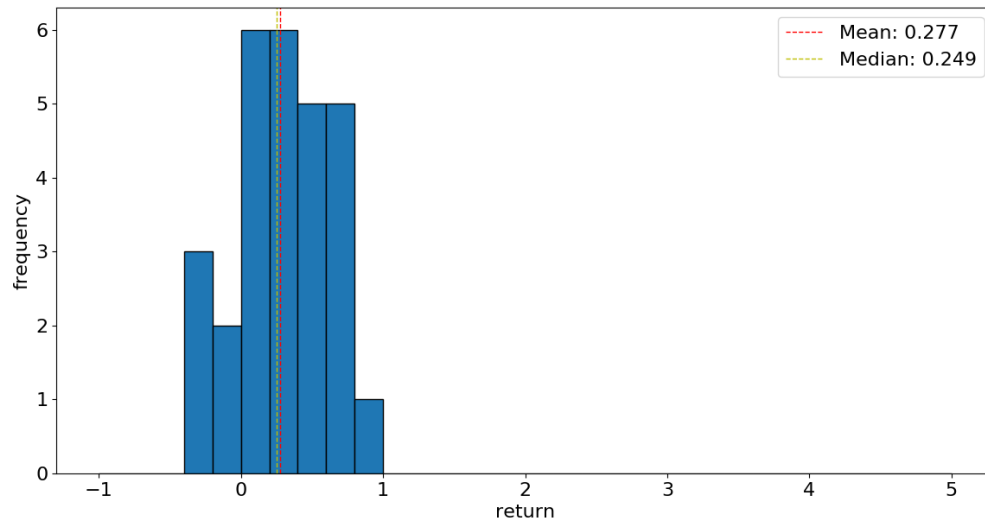


Figure 25: The return distribution of other assets in period 1

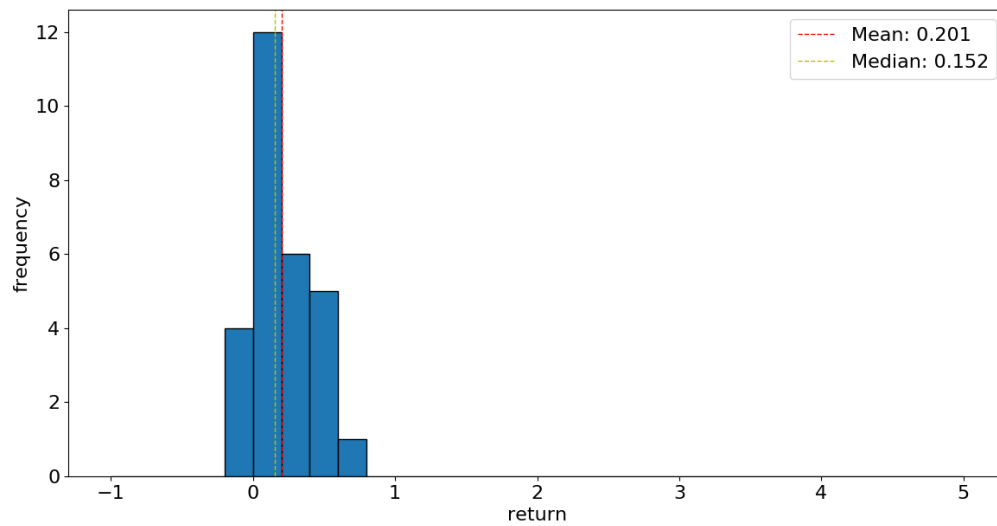


Figure 26: The return distribution of other assets in period 2

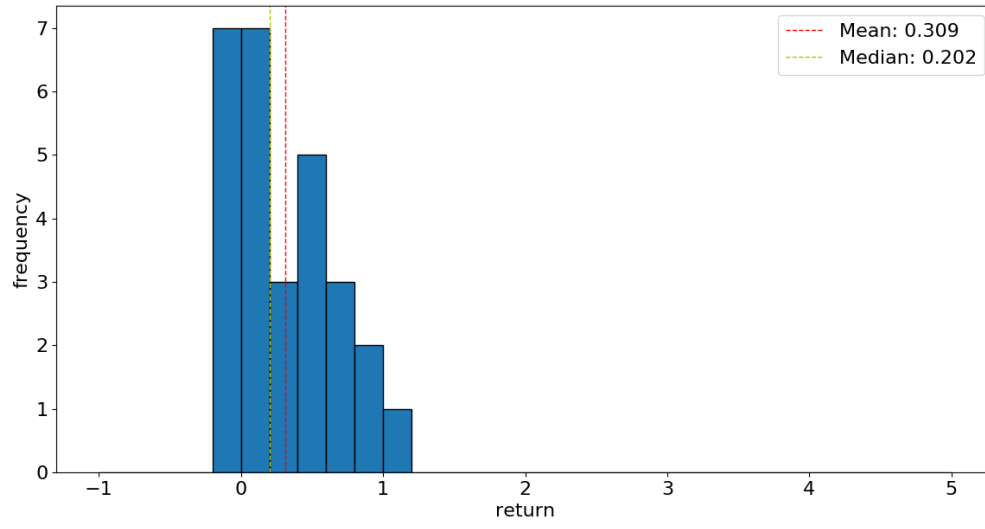


Figure 27: The return distribution of other assets in period 3

3 Discussion

3.1 Comparison of Baseline Strategies vs RL Trading Agent

3.1.1 Limitations of Baseline Strategies

Execution - Baseline Suffers From Premature Exits Due to Explicit Definition of Loss Limit Parameter

The economic meaning of loss limit is straightforward: if one is more risk taking, he or she is willing to take more loss in the hopes that short term fluctuations in the opposite direction “is not real”. Conversely, a risk averse investor with low loss limit will exit positions early, forgoing the possibility of profit while ensuring the “certainty” of limited loss at small amounts. Unlike humans with discretionary capabilities, the issue of premature exits is more significant with rule-based trading logic. This is because systematic strategies will not evaluate parameters on a case-by-case basis like human.

In the context of our backtesting, a premature exit means that after the trading strategy has decided to enter a position (by “longing” or “shorting” the spread), it exits quickly afterwards before the spread returns to the mean because it has suffered a small amount of loss (controlled by the **loss limit** parameter).

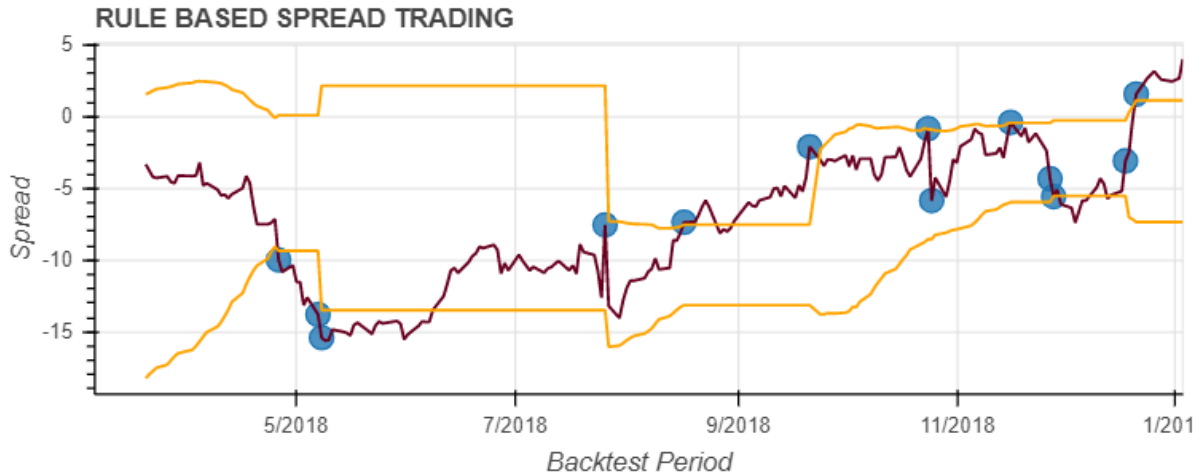


Figure 28: We can observe that in **June 2018**, the strategy decides to enter positioning (long the spread), but very quickly exits afterwards because it has suffered a small loss (due to the spread trending down temporarily).

Our proposed reinforcement learning (RL) trading agent does not suffer from this limitation because during the optimization procedure (training phase), the model has learnt to make trade actions (short spread, long spread, exit spread) optimally based on the current normalized log prices to maximize expected returns. One possible interpretation is that our RL trading agent has learnt to set *adaptive loss limit* which is more robust and catered toward market conditions.

For baseline strategies, we have explored different ways to overcome the issue of reduced profits due to premature exits, and they are documented as follows:

Approach 1: optimize the “best” loss limit parameter (grid search)

The first solution that could solve the issue of premature exits is to increase the size of loss limit such that small fluctuations near the upper or lower boundaries are allowed. If the loss limit is too small, for spreads with high standard deviation, the loss limit may be hit even before the spread can be given an opportunity to converge towards the mean. Conversely, if the loss limit is too high, we may unintentionally hold positions where the spread never converges because there is a new long term mean. Hence, this solution to premature exits problem introduces a trade off between higher average returns and having a fat tailed returns distribution

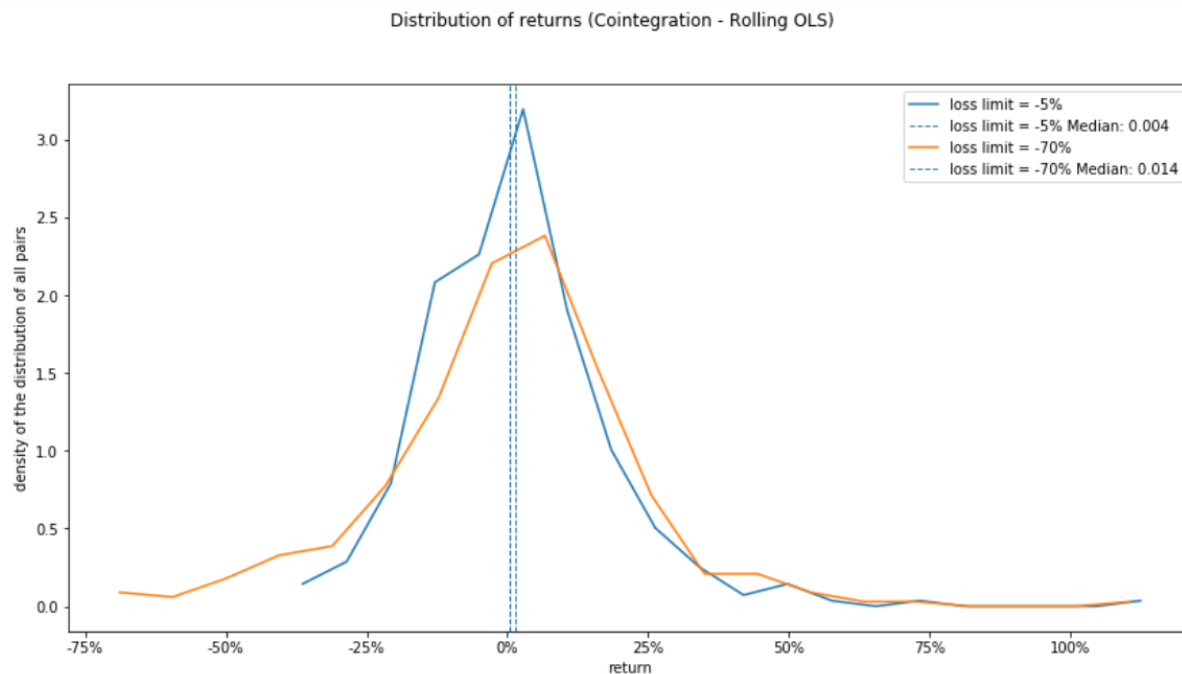


Figure 29: Results of backtesting over 10 months using Cointegration (rolling OLS) using `loss_limit = -5%` and `loss_limit = -70%`. We can observe that the distribution of returns when `loss_limit = -70%` has a slightly better median but more fat-tailed (ie. more extreme values at both ends).

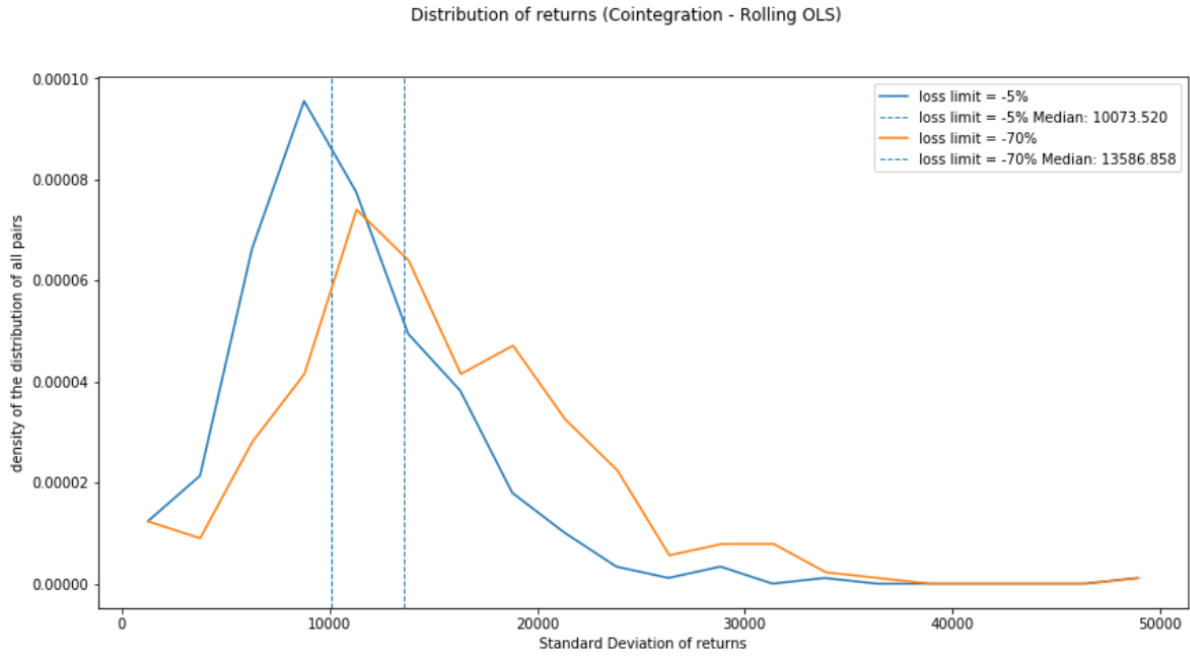


Figure 30: Results of backtesting over 10 months using Cointegration (rolling OLS) using `loss_limit = -5%` and `loss_limit = -70%`. We can observe that the distribution of *standard deviation of returns* when `loss_limit = -70%` is significantly higher.

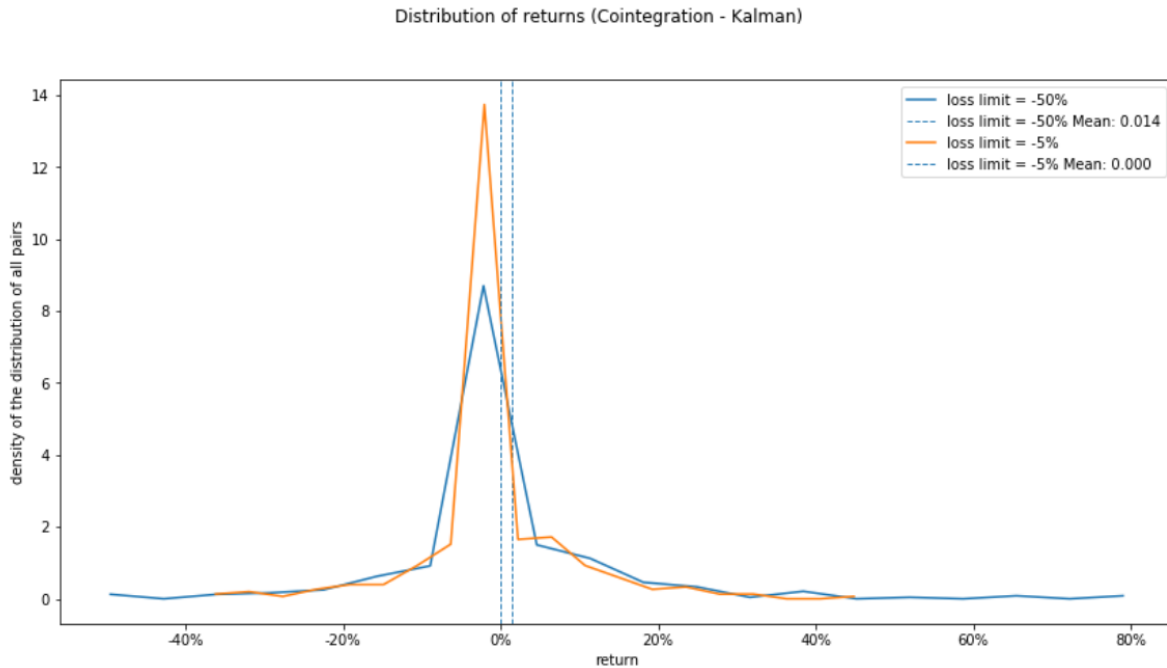


Figure 31: Results of backtesting over 10 months using Cointegration (Kalman) using `loss_limit = -5%` and `loss_limit = -50%`. We can observe that the distribution of returns when `loss_limit = -50%` has a slightly better mean but more fat-tailed (ie. more extreme values at both ends).

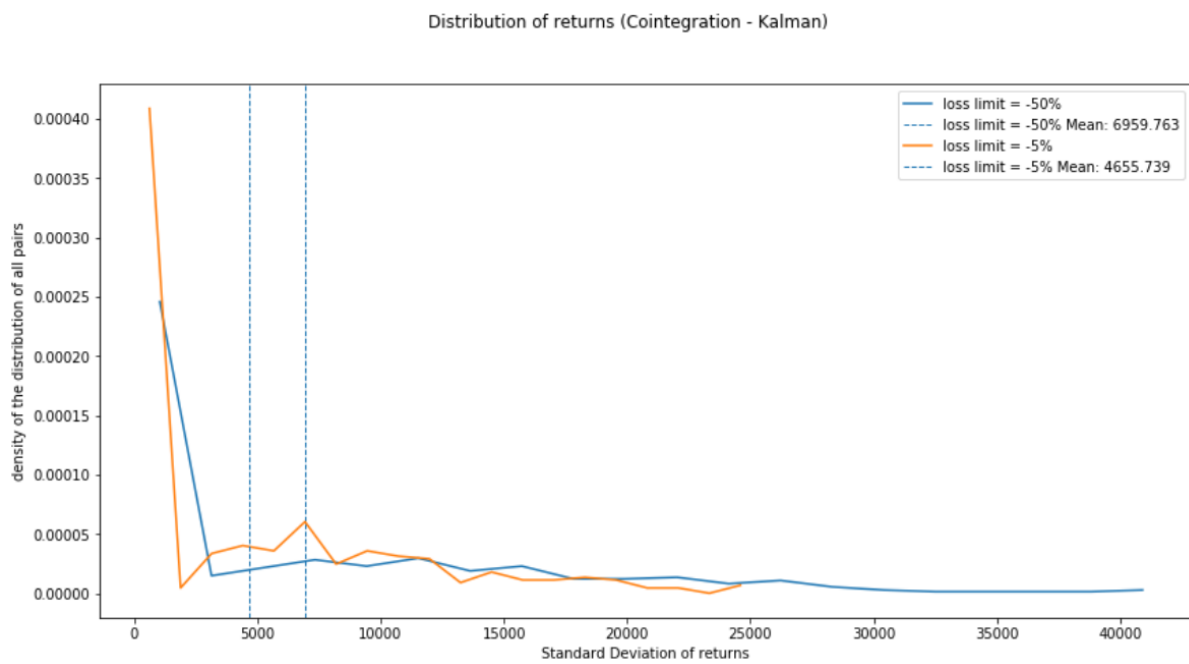


Figure 32: Results of backtesting over 10 months using Cointegration (Kalman) using `loss_limit = -5%` and `loss_limit = -50%`. We can observe that the distribution of *standard deviation of returns* when `loss_limit = -50%` is significantly higher.

Approach 2: Introduction of New Parameters

This approach is not implemented in our project, but an educated guess to tackle the issue of premature exits is to model the premature exit itself with new parameters. One possible solution is to include two additional parameters that model the maximum tolerance for delay before a trade becomes profitable. This can be done by introducing `touch_count` TC , and `touch_horizon` TH . These parameters are activated only when a trade is entered. `touch_count` means the max number of periods out of the `touch_horizon` observation period that a position is allowed to be maintained.

For example, consider a trade entered at $t = 0$ with parameters $TC = 3$ and $TH = 5$. as long as a trade does not violate the max loss parameter for TC number of times within $t = 1$ to $t = TH$, then the trade position is allowed to be maintained. This is potentially be beneficial as it provides a grace period for the mean-reversion property of the pair to be activated while limiting losses to be nearer to loss limit.

Spread Modelling - Inability of Baseline to Detect Changes in Long Term Mean

The key assumption in pairs trading is that the spread will always converge back to its long term mean. However, this is not always the case as there can be transitional phases where the spread moves from one long term mean to another. Therefore, this poses a challenge to baseline strategies (more generally, rule based strategies) as the entry and exit levels is expressed as a multiple of the standard deviation from its mean. At the time when a trade is opened, the mean is recorded and the exit levels are based on this static number.

The novelty of using reinforcement learning to make optimal trading decisions is that the rules for entry and exit are no longer bounded by some optimal level of spread. In other words, the RL trading agent avoids the task of identifying optimal entry and exit levels as a multiple from the mean altogether. However, there is no conclusive proof from our tests that suggests the agent can identify when there is the long term mean is in a transitional phase.

Finally, close inspection of the spread time series in baseline strategies suggests an exception: the Kalman Filter variant of cointegration method models the spread with “**good stability**”. This is because the Kalman Filter has a underlying model of the spread that can filter out noise.

3.1.2 Ranking and Summary of Trading Strategies

1. **RL Trading Agent:** Overall, our proposed RL Trading Agent performs bests under general conditions.
2. **Cointegration with Kalman Filter:** Among all the baseline strategies, Kalman can be considered as the **most risk averse** strategy among all, because the average returns distribution of all pairs has very positive kurtosis, therefore making it a low-risk low-reward strategy.
3. **Distance:** In contrast to Kalman variant of cointegration, the returns is highly unstable. Analysis of our results suggest that the returns is somewhat related to general market volatility. When markets are extremely volatile, such as in period 3, the returns is highest among all baseline methods, around +10%, but under periods of low market volatility, the return is negative.
4. **Cointegration with Rolling OLS:** On average, this strategy is the worst performing strategy. There are two main issues with rolling OLS approach. Firstly, this approach has noisy estimation of essential

parameters (α and β). Secondly, we are unable to update entry and exit levels dynamically when there is an open position because the trade needs to reference back that static entry and exit levels recorded at time of entry.

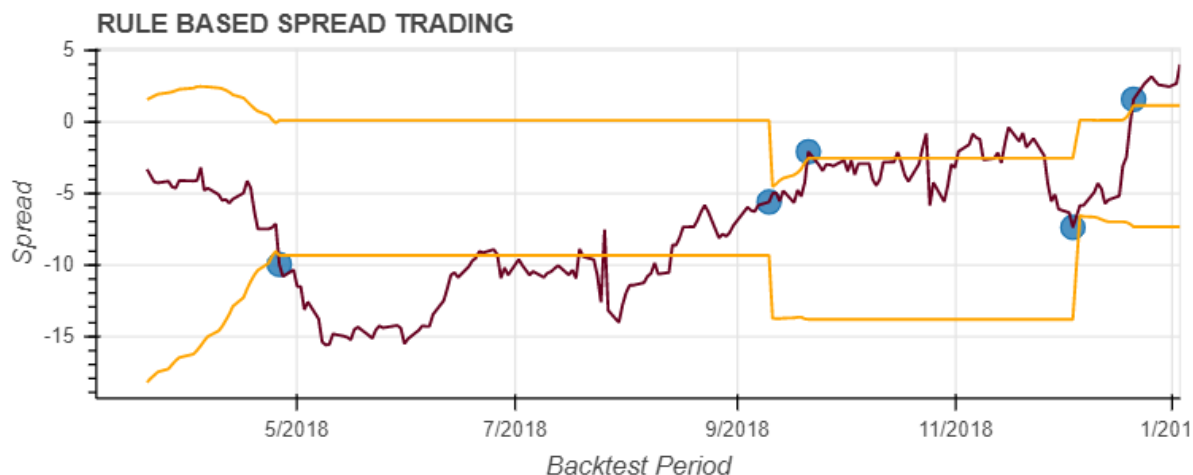


Figure 33: Red line is the spread of the pair of stocks AAN and AER backtested over a period of 10 months using **distance method**. We can see that the spread estimated by this method is highly unstable (does not seem stationary at all).

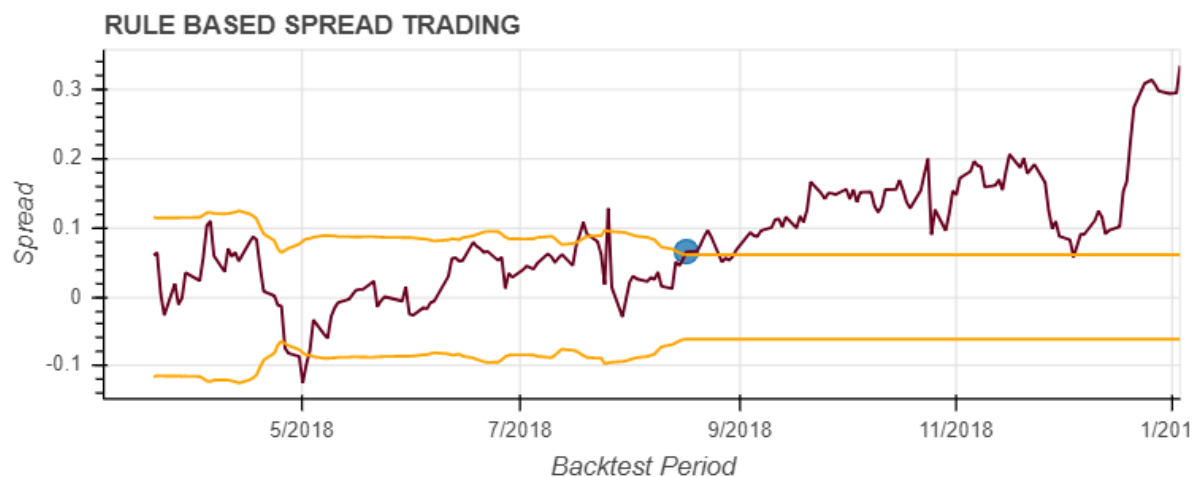


Figure 34: Red line is the spread of the pair of stocks AAN and AER backtested over a period of 10 months using **cointegration - rolling OLS**. We can see that the spread estimated by this method is stable at the beginning but does not respond quickly enough when parameter changes over time.

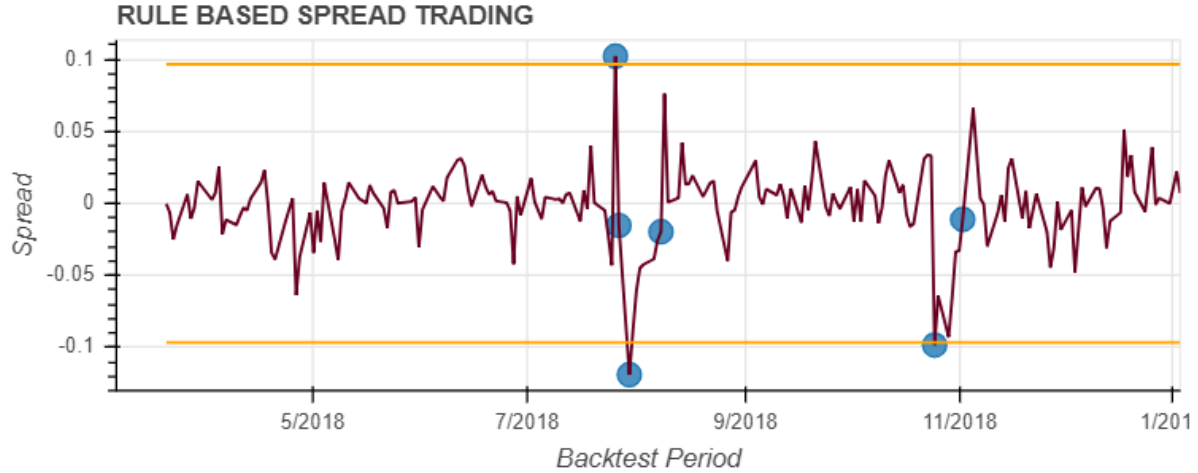


Figure 35: Red line is the spread of the pair of stocks AAN and AER backtested over a period of 10 months using **cointegration - time varying kalman filter**. We can see that the spread estimated by this method is very stable (almost stationary).

3.1.3 Review of RL Trading Agent - Issues in Reinforcement Learning

For any reinforcement learning project, the task to evaluate the effectiveness of one's algorithm poses a challenging task due to the “blackbox” nature of machine learning. To further ensure the robustness of our RL Trading Agent, we have evaluated the algorithm in the following areas aside from its profitability:

Learning

By inspecting the trading action made by the agent given a pair of stock price series, this is to ensure the agent learned to pick the right action at the right time. It seems that the trading agent can decide the best action using only the normalized log prices as inputs. It may not be sensible to act according to the spread, as shown in Figure 36 because since the spread definition provided as input is constantly changing (computed with rolling OLS).

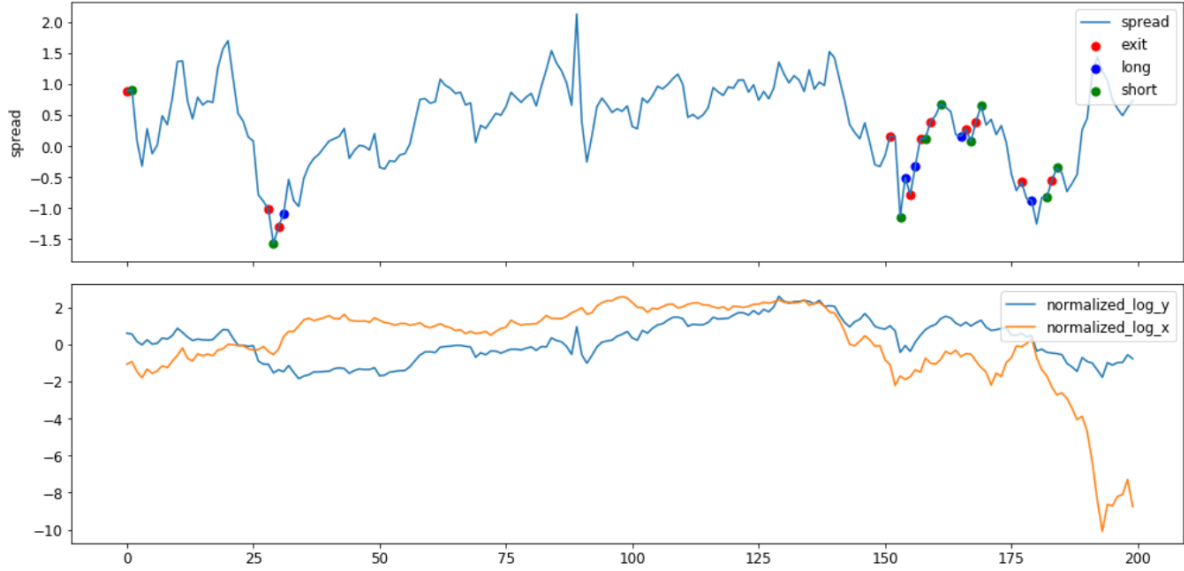


Figure 36: The trading actions made by the RL agent when trading the pair AAN and AER in period 3.

Generalization

Since the goal is for the agent to **learn the skill of pairs trading**, it is worthwhile to see if it can extrapolate what it learned from a specific sector in the stock market to different sectors and different asset classes. Therefore, after we trained a agent using data from the technology stocks from NYSE, we evaluated the performance of the trading agent on the energy stocks from NYSE and some other assets. The result showed that the agent successfully transferred the learned information using stocks from the technology sector to other assets class as indicated by the results.

The first test of the agent's generalization ability is applied on the energy sector stocks. Underlying this test is the economic intuition that the factors affecting technology stocks are different to that of those affecting energy stocks (e.g. breakthrough technology and change in consumer lifestyle vs global energy demand and inflation). Although we initially assumed that the factors affecting tech stocks was more short term since changes are more fast paced, and energy stocks was more long term, it seems that performance in both sectors (median returns) improved with longer training data. It is unclear if there is a third confounding economic factor affecting both sectors which the agent has learned, or if the agent just becomes better by observing more cycles in longer dataset.

The second test data is applied to data that has completely different factors as opposed to the technology sector, such credit asset class, interest rates, currencies, and commodities. Yet, the RL trading agent still successfully made profits while trading these stocks. This observation is in line with our findings in what the agent has learned.

3.1.4 How Scalable Are The Profits; is it Replicable in Multi-Billion Dollar Funds

Review of Assumptions

- **Transaction Costs:** To make the performance more realistic, we included the transaction costs that is paid by us to the **Interactive Brokers** for using their agency services. This **is included** in the

project and is calculated by a formula as shown in section 2.1.3.

- **Transaction Price:** To make trades under real circumstances, market takers need to buy at the market's ask price and sell at the market's bid price. Conversely, market makers will transact at the opposite to that of market takers. This requirement **is omitted** in the scope of this project for two reasons as we assume all transactions occur in the **mid-price**, which is $(Bid + Ask)/2$.

Also, the prices quotes on the markets are usually in prices for the first N quantities of stocks (see 37). Therefore, for a large trade, one may be required to transact at multiple levels of prices to fill the whole size of the trade. For simplicity of modelling, we assume that the entire size of the trade is done on the mid price.

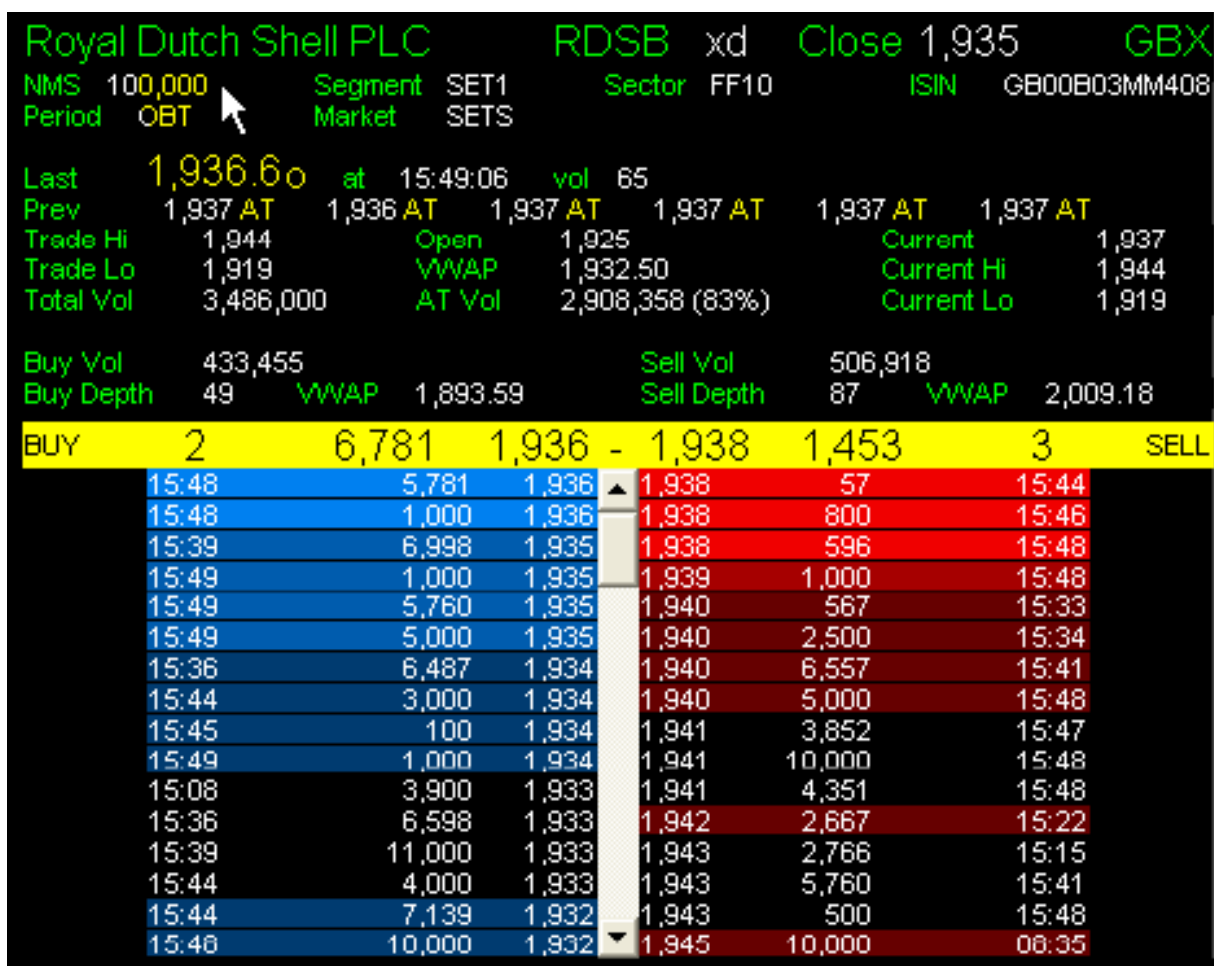


Figure 37: Example L2 market bid-ask data. Each column represents a price which is quotes on the bid or ask side, together with the associated amount for the specified transaction price and direction.

- **Borrowing Costs:** Legally, to sell a stock, a trader is required to first borrow it from a stock lender, and then sell it. This will incur a borrowing cost as quoted by the lender, and this cost is subject to supply and demand of that stock. This requirement **is omitted** because of difficulty retrieving the borrowing cost time series.

Initially, we applied a static cost taken from one day for all the stocks and assume that the stocks have that specified borrowing for the backtesting period, but this will affect the accuracy of the reported

profits in backtesting. This is because the entry and exits (profit condition) also depends on the borrowing cost. If an abnormally high borrow cost is used statically for the entire backtest period, the pair may never be traded. Therefore, we think it is better to omit this requirement and instead, assume that there is an annualized yield to borrowing cost, and subtract this from the raw profits obtained.

- **Borrowing Stability:** Another issue of selling borrowed stocks is that the pairs trade position could be exited because the lender (which has the right to do so) opts to recall the lent stock. This requirement is **omitted** for the project.
- **Latency of Building Position:** For this project, we assume an initial capital of 10,000 US Dollars for a fund. Relative to the daily traded volume of the stock market, this is a small amount. Hence, we assume that there is no delay between time where RL Trading Agent initiates a position and the time required to completely fill the position.

In addition, we assume that there is no price slippage due to our trade. This assumption may not hold if we scale up our initial fund size, for example, to 1 Billion USD, liquidity becomes a key issue in the successful implementation of the long and short positions of the pairs trade, especially if small cap stocks are included.

- **Dividends:** On the long side, if we long a stock that gives dividend during the time of a trade, the receipt of cash dividends (or stock div, or stock splits) are **not** handled by the code. The reverse issue (i.e. paying the dividends to the stock lender), is also omitted from the code.

Similar to borrowing cost, we can overcome this issue by assuming an annual dividend yield as an arbitrary positive amount, but this is hard to estimate, so it is omitted from our analysis. Hence, we assume that the effects of dividends on the long and short side net off each other, therefore the impact to returns is negligible.

- **Portfolio Construction:** For this project, we assume that a portfolio contains exactly one pair with the long and short to be 10,000 USD each. Under real life conditions, a fund manager may want to have a portfolio of multiple pairs at any one time. This introduces two potential issues that is not covered in this project.

Firstly, **netting** of positions may be an issue. Theoretically, if pair (A, B) and (B, C) are both cointegrated, pair (A, C) should also be cointegrated. Suppose the fund manager wants to select the top $k\%$ of pairs but where pair (A, B) and (B, C) are included, but pair (B, C) is not included for whatever reason. In the event that the algo suggest long A short B for (A, B) , and long B short C for (B, C) , the fund manager is indirectly trading pair (A, C) due to effect of netting.

Secondly, **correlation** of spreads could be an issue to the performance of a portfolio of pairs. Suppose a fund manager has chosen N pairs that do not have any netting effect. There is possibility that spread of pair (A, B) and spread of pair (C, D) are negatively correlated, so profits of trading (A, B) could be offset by losses in trading (C, D) .

A counterargument for this phenomenon to be beneficial is that this implies the portfolio is diversified, but this does not make sense from an economic perspective, since trading a pair itself should have already diversified systematic market risk common to both stocks in the pair; it is unclear what other risks are diversified from a basket of spreads.

Separately, a negative correlation of pair (A, B) and (C, D) , could be mean positive correlation if the spread of (C, D) is reversed (e.g, calculate spread as $D - C$ instead of $C - D$). Therefore, it makes sense to seek uncorrelated pairs.

- **Liquidity of Pairs:** Liquidity means the ability for one to enter or exit a position easily. A highly liquid stock (or high liquidity market) means that there is sufficient trading volume or bidding interest such that a trader can completely fill the size of the order. In order for the returns to be scaleable, it is worthwhile to analyze the distribution of profitable pairs as a combination of their market capitalization, since small cap stocks are usually illiquid. Therefore, if the RL Trading agent has identified a large proportion of profitable pairs to be small-small combination, the returns may not be scalable to positions more than 1 million USD.

4 Conclusion

Summary of Key Technical Achievements and Lessons Learned

Firstly, we conducted a comprehensive study of traditional strategies in pair trading and implemented three of the most famous approaches: Distance Method, Cointegration (Kalman), and Cointegration (Rolling OLS) as baselines.

More importantly, we developed a profitable trading logic using Reinforcement Learning, and the RL trading agent outperforms both baseline methods and standard market indices. The remarkable outcome of this project is the generalizing ability of this RL trading agent and the applicability of it to other sectors (e.g. Energy) and pairs from other asset classes, and this is demonstrated in the evaluation section. This is because the agent has learned to differentiate meaningful inputs from noise to maximize its rewards: the RL agent only observes the normalized log prices of a pair of assets.

Issues Not Addressed In Our Project and Future Work

To make pairs a trading strategy complete, we need both pair selection and trading logic. In this FYP, we have developed a very profitable trading logic (trading agent knows to choose right action at a given time), but due to time constraints we have yet to develop a novel pair selection method which can outperform baseline pair selection methods. (cointegration & distance). On a bigger picture, the issue of liquidity, such as modelling transaction costs accurately, must be addressed to ensure the results are replicable in real life. In Summary, the following items are suggested future work to build on this project:

- **Modelling the environment:** Incorporate L2 trading data for accurate bid and ask price. L2 data should also allow the trading agent to decide how much of the long-short size of the pairs trade: if the prices are too far off the mid price, it may be less worthwhile to make a big pairs trade due to price slippage. Lastly, it may be worthwhile to experiment with “touch parameters” as introduced in .
- **Trading and Portfolio Selection:** Pair selection using trading agent performance data to return the top N pairs to be traded. Ideally, the RL Trading agent will also specify the size of the pairs trade and rank them according to the criteria of potential profits and liquidity in the market. Lastly, it may be worthwhile to investigate the applicability of the trading agent on different data frequency (e.g. 5 minutes) such that the constraint of liquidity becomes reduced; one may slice the orders into smaller orders to obtain a better price.

In conclusion, our RL Trading Agent has succeeded on “when to trade” a given pair, so future work should tackle the questions “which pairs to trade and prioritize”, “how to execute large size of pairs”, and “how to allocate a portfolio of pairs”.

5 Hardware and Software

5.1 Hardware

- GPU for training neural networks.
- Intel AI DevCloud
- Laptops with Ubuntu OS for development.

5.2 Software

- Ubuntu OS.
- Python.
- Jupyter Notebook.
- Sublime text.
- Git.

References

- [1] T. Demos, “Citigroup: Big bank, big spender on tech,” *The Wall Street Journal*, Dow Jones & Company, 2018. [Online]. Available: blogs.wsj.com/moneybeat/2018/05/30/citigroup-big-bank-big-spender-on-tech/.
- [2] D. A. Dickey and W. A. Fuller, “Distribution of the estimators for autoregressive time series with a unit root,” *Journal of the American Statistical Association*, vol. 74, no. 366, p. 427, 1979. DOI: 10.2307/2286348.
- [3] B. Do, R. Faff, and K. Hamza, “A new approach to modeling and estimation for pairs trading,” Jan. 2006.
- [4] V. Dreissen, “A successful git branching model,” *nvie.com*, 2010. [Online]. Available: <https://nvie.com/posts/a-successful-git-branching-model/>.
- [5] R. J. Elliott, J. V. D. H. *, and W. P. Malcolm, “Pairs trading,” *Quantitative Finance*, vol. 5, no. 3, pp. 271–276, 2005. DOI: 10.1080/14697680500149370.
- [6] R. F. Engle and C. W. J. Granger, “Co-integration and error correction: Representation, estimation, and testing,” *Econometrica*, vol. 55, no. 2, p. 251, 1987. DOI: 10.2307/1913236.
- [7] E. Gatev, W. Goetzmann, and K. G. Rouwenhorst, “Pairs trading: Performance of a relative value arbitrage rule,” en, 1999. DOI: 10.3386/w7032. [Online]. Available: <https://pdfs.semanticscholar.org/e5f1/b9530a587c5910ceea57ed80270ee22c73b4.pdf>.
- [8] R. v. d. Have, “Pairs trading using machine learning: An empirical study,” Master’s thesis, Jan. 2018. [Online]. Available: <http://hdl.handle.net/2105/41548>.
- [9] N. Huck and K. Afawubo, “Pairs trading and selection methods: Is cointegration superior?” *Applied Economics*, vol. 47, no. 6, pp. 599–613, 2015. DOI: 10.1080/00036846.2014.975417. eprint: <https://doi.org/10.1080/00036846.2014.975417>. [Online]. Available: <https://doi.org/10.1080/00036846.2014.975417>.
- [10] C. Krauss, “Statistical arbitrage pairs trading strategies: Review and outlook,” *Journal of Economic Surveys*, vol. 31, no. 2, pp. 513–545, Sep. 2016. DOI: 10.1111/joes.12153.
- [11] A. Ng, *Cs229 lecture notes: Part xiii reinforcement learning and control*. [Online]. Available: <http://cs229.stanford.edu/notes/cs229-notes12.pdf>.
- [12] D. P. Palomar, *Pairs trading*, 2018. [Online]. Available: http://www.ece.ust.hk/~palomar/MAFS6010R_lectures/week%5C%2012/slides_pairs_trading.pdf.
- [13] S. H. Penman, *Accounting for Value*. Columbia Business School Publishing, 2011.
- [14] A. Pole, *Statistical arbitrage: algorithmic trading insights and techniques*. Wiley, 2007.
- [15] D. Silver, *Lecture 2: Markov decision processes*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MDP.pdf.
- [16] R. Ssekuma, “A study of cointegration models with applications,” en, 2011.
- [17] G. Vidyamurthy, *Pairs Trading: Quantitative Methods and Analysis*. John Wiley & Sons, Inc., 2011.
- [18] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, “Solving deep memory pomdps with recurrent policy gradients,” in *ICANN‘07*, Max-Planck-Gesellschaft, Berlin, Germany: Springer, Sep. 2007, pp. 697–706.

- [19] R. Wigglesworth, “Bond trading: Technology finally disrupts a \$50tn market .,” *Financial Times*, 2018. [Online]. Available: www.ft.com/content/67e48ae4-4fab-11e8-9471-a083af05aea7.

6 Appendix A - Meeting Minutes

Minutes of the 1st meeting

Date: 07/09/2018

Time: 2PM

Place: Room 3504

Present: Prof. Nevin L. Zhang, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: This is the first meeting.
2. Reports on progress:
 - All group members have read some papers regarding pairs trading.
3. Discussion items:
 - Clarified on logistics issues regarding the project such as project deadlines, future meeting schedules.
 - Explained project idea in detail to Professor Nevin.
 - Professor Nevin reminded us to get our datasets as soon as possible.
 - Professor Nevin advised us to approach one of his MPhil student, Nan Zhang to seek further advice on our project because Nan Zhang has had some relevant experience.
4. Goals for the week:
 - Read more papers relating to pairs trading.
 - Approach Nan Zhang for potential help and guidance.

Minutes of the 2nd meeting

Date: 10/09/2018

Time: 3PM

Place: Room 5559F

Present: Prof. Ningyuan Chen, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.
2. Reports on progress:
 - All group members have studied relevant papers assigned by Prof Ningyuan Chen relating to PCA analysis.
 - Completed first draft of literature review.
3. Discussion items

- Clarified the mean-reverting behavior to be the returns of a stock, not the price itself as we initially assumed. Specifically, the mean-reverting term is identified to be the idiosyncratic returns.
- The idiosyncratic term is unobservable; therefore, to “trade the mean-reversion”, we need to buy something and sell another, such that the “common factors” are net off, leaving behind the idiosyncratic term.
- Separately, since the idiosyncratic returns is unobservable, we can classify this part of the project as a expectation maximization part. This is done using the spread of two stocks and is treated as a signal processing step.
- For the stochastic model, we find that a “noise” boundary separately from the targeted entry and exit boundary, otherwise we will make a mistake in entry signals.
- To obtain the variance, it is derived from the steady state equilibrium.

4. Goals for the week

- Read more papers.
- Finalize literature survey section and continue the proposal report draft.

Minutes of the 3rd meeting

Date: 18/09/2018

Time: 3PM

Place: 5559F

Present: Prof. Ningyuan Chen, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.
2. Reports on progress
 - Gordon proposed the idea of BETA clustering
 - Gordon proposed autoencoder model to capture the non-linear factors in the stock factor model of returns
 - Wen Yan proposed ensemble method of spread prediction by combining stochastic spread model, recurrent neural network, and feedforward neural network.
 - Follow up session with Prof Ningyuan conducted to interpret eigenportfolio and how to trade it.
3. Discussion items
 - Concluded that eigenportfolio of stocks is not feasible to trade because of constant rebalancing of many stocks will incur huge transaction cost.
 - It is better for us to model factor returns using ETFs. The additional benefit is the ability to easily sell an ETF by buying an “inverse ETF”, rather than having to borrow the stock and then sell it.
 - From modelling perspective, using small number of “factors” is beneficial for small sample size because of increased robustness of linear regression coefficients.

- Raised key discussion point on how to handle special cases of spread: how to handle stocks with sudden jumps, and stock with upward/downward momentum. This is because stocks with frequent jumps or strong momentum in a certain direction will make the spread non mean reverting.
- Proposed to reuse the spread model for volatility spread as a follow up contribution, after price spread model. Prof Ningyuan suggested that it may not be feasible because volatility is time-varying.
- Proposed to model the stock spread, and trade the mean-reversion using options to express the view. However, the conclusion is that it will not work because we are only trading delta of the option; an option will also be affected by other factors such as vega and gamma.

4. Goals for the week

- Complete final version of progress report.
- Read more papers on backtesting time series data
- The team will assume no arbitrage pricing theory: if a pair has similar risk factors, and the returns are cointegrated, then their price time series should be cointegrated.

Minutes of the 4th meeting

Date: 29/11/2018

Time: 4PM

Place: Room 3504

Present: Prof. Nevin Zhang, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.

2. Reports on progress

- Completed application for broker account with Interactive Brokers.
- Completed setup of API connection with broker to retrieve data (historical and live stream) as well as send dummy orders on paper account.
- Completed understanding backtrader API for backtesting simple distance method. Performed sample backtest on GOOG-GOOG pair and demonstrated to Prof Nevin.

3. Discussion items

- We noticed that the pair which was initially identified as a good pair could violate the mean-reverting property in the future. It is possible that if a pair is identified to be mean-reverting from t to $t + 60$, the same pair may not be mean-reverting from $t + 60$ to $t + 120$.
- Prof Nevin suggested us to test pairs within sectors first, instead of testing pairs randomly.
- Although we have retrieved 5-minute level data, we have concluded that it not feasible to start development process of the code using this data because the training process is too slow, hence the trial and error process may be slow.

4. Goals for the week

- Complete implementation to systematically backtest all pairs and generate reports to analyze results.
- Complete implementation of cointegration method as baseline strategy.
- Explore machine learning methods for pairs trading as the main strategy of the project.

Minutes of the 5th meeting

Date: 11/01/2019

Time: 2PM

Place: Room 3504

Present: Prof. Nevin Zhang, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.
2. Reports on progress
 - Completed implementation of baseline strategies (Distance method and cointegration method) and performed experiments on NYSE listed technology stocks. On average, the results returned 5-10% annually over the 2015-2019 trading period.
 - Completed first implementation of proposed machine learning trading algorithm using policy gradient method. This was further discussed with Prof Nevin.
3. Discussion items
 - We discussed the overall model architecture of the policy gradient with Prof Nevin.
 - At each time step t the observation of the agent to be the current price of the pair of stocks and the spread of the pair.
 - The observation at each time step t and the output from the time step $t - 1$ will then be fed into an Recurrent Neural Network (RNN) and output a single state vector. We decided to use an RNN because we would like to encode the entire history of observation into one state vector.
 - The output at each time step t from the RNN together with the current trade position will then be fed into a neural network and a softmax layer which will output a probability distribution over the action space.
 - Originally, we defined a new reward function (by separating the returns long stock leg vs the short stock leg). Prof Nevin suggested that it is better off to treat the returns as a whole portfolio for simplicity of modelling in the RL method.
 - In terms of presentation of the model architecture, Prof Nevin recommended that we separate the RNN (used for encoding observations into a state vector), and the policy network.
 - If we want to use trading algorithm (reinforcement learning) to generate training data for pair selection classifier, we must first prove the correctness and good performance of the reinforcement learning based trading algorithm.
4. Goals for the week

- Change the reward function for the trading agent.
- Continue to run tests (different parameters i.e. number of hidden layers) to ensure the algorithm is proven to make profit.
- Include trading costs (transaction, borrowing cost) into PnL calculation.

Minutes of the 6th meeting

Date: 16/01/2019

Time: 3PM

Place: Room 5559F

Present: Prof. Ningyuan Chen, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.
2. Reports on progress
 - Same as in 5th project meeting.
3. Discussion items
 - We explained our derivation for the profit condition to verify its correctness. Prof Ningyuan suggested that it is probably better to avoid (assuming alpha equal to 1); its better to just fit the parameters.
 - We described in detail the procedure used in tuning the parameters for backtesting baseline strategies. Specifically, for the fitting of alpha and beta in OLS regression, we concluded that it is better to freeze those parameters when a trade is already in place, otherwise the trading logic will be problematic.
 - For optimization of parameters, Prof Ningyuan recommended us to use simple grid-search method.
4. Goals for the week
 - Same as in 5th project meeting.

Minutes of the 7th meeting

Date: 14/02/2019

Time: 3PM

Place: Room 2541

Present: Prof. Nevin Zhang, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.
2. Reports on progress

- We showed some figures and charts for Professor Nevin to understand the performance of our proposed RL trading agent.

3. Discussion items

- Professor Nevin advised us to explain explicitly to user how to read the result charts
- Emphasize on telling the read why result is good. In the chart, tell them how many % of returns is significant & profitable
- Discussion on the issue where performance of a pair in training time and testing time have little to no correlation
 - Maybe because predictive power of RL model decays over time
- More discussion on the reinforcement learning formulation:
 - Is portfolio part of state?
 - Is there exploration of the environment by the agent?
 - Where to encode the portfolio information? Reward function or state? Whats the difference?

4. Goals for the week

- Discuss what the RL agent has learned in your experiment? Pros/cons?
- Prepare more visualizations to show the trade actions performed by the RL trading over time.

Minutes of the 8th meeting

Date: 13/04/2019

Time: 3PM

Place: Room 2541

Present: Prof. Nevin Zhang, KO Chung Wa, Wen Yan, Brendan

Recorder: Brendan

1. Approval of minutes: The minutes of last meeting were approved without amendment.

2. Reports on progress

- We showed the frontend UI to Professor Nevin.

3. Discussion items

- Professor Nevin advised us put the histogram on the same x, y axis scale in the final report so that it is easy to understand.
- It might be useful to add a background story before explaining the spread so that people with less finance background can understand.
- Some portion of our work is novel, so it might be possible to publish a paper about it if we want.

4. Goals for the week

- Finish the FYP final report.

7 Appendix B

7.1 Profit condition for cointegration method (with log price spread)

Consider two stock X and Y , and their observed true price at time t being X_t and Y_t . We assume the two observed true price follows the relation:

$$\log Y_t = \alpha \log X_t + \beta + Z_t,$$

where $\alpha > 0$ and $Z_t = \log(Y_t) - \alpha \log(X_t) - \beta$ being the spread of this pair of stocks. According to the past statistics of the spread, we may view the spread being over-priced or under-priced, and then we can short (sell) the spread or long (buy) the spread. Note that $\alpha > 0$ means that the two log prices have similar trend if there is one. When we short or long the spread, we take opposite position on the two stocks and thus the risk associated with the stock trends cancel each other.

7.1.1 Short the Spread

Following figure 1, suppose at $t = 0$ we think that the spread Z_0 is over-priced. We can short the spread by selling y shares of Y at Y_0 and buying x shares of X at X_0 . Assume y is some constant and $x = \frac{yY_0}{X_0}$.

Later at $t = c$ we unwind the position by buying back y shares of Y at Y_c and selling x shares of X at X_c .

The net gain for investing in Y :

$$y(Y_0 - Y_c) - fee$$

The net gain for investing in X :

$$\begin{aligned} & x(X_c - X_0) - fee \\ &= \frac{yY_0}{X_0}(X_c - X_0) - fee \\ &= yY_0 \left(\frac{X_c}{X_0} - 1 \right) - fee \\ &= yY_0 \left(\frac{\exp(\alpha^{-1}(\log Y_c - \beta - Z_c))}{\exp(\alpha^{-1}(\log Y_0 - \beta - Z_0))} - 1 \right) - fee \\ &= yY_0 \left(\exp \left(\alpha^{-1} \left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c \right) \right) - 1 \right) - fee \end{aligned}$$

The overall net gain for investing in both X and Y :

$$\begin{aligned} & y(Y_0 - Y_c) - fee + x(X_c - X_0) - fee \\ &= y(Y_0 - Y_c) + yY_0 \left(\exp \left(\alpha^{-1} \left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c \right) \right) - 1 \right) - totalFee \\ &= -yY_c + yY_0 \exp \left(\alpha^{-1} \left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c \right) \right) - totalFee \end{aligned}$$

In order to make a profit, the net gain should be greater than zero:

$$\begin{aligned}
-yY_c + yY_0 \exp\left(\alpha^{-1}\left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) - totalFee &> 0 \\
\exp\left(\alpha^{-1}\left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) &> \frac{totalFee/y + Y_c}{Y_0} \\
\log \frac{Y_c}{Y_0} + Z_0 - Z_c &> \alpha \log\left(\frac{totalFee/y + Y_c}{Y_0}\right) \\
Z_0 - Z_c &> \log\left(\frac{(totalFee/y + Y_c)^\alpha}{Y_c} Y_0^{1-\alpha}\right)
\end{aligned}$$

Note that if $\alpha = 1$, we have

$$Z_0 - Z_c > \log\left(\frac{totalFee/y}{Y_c} + 1\right).$$

7.1.2 Long the Spread

Suppose at $t = 0$ we think that the spread Z_0 is under-priced. We can long the spread by buying y shares of Y at Y_0 and selling x shares of X at X_0 . Assume y is some constant and $x = \frac{yY_0}{X_0}$.

Later at $t = c$ we unwind the position by selling back y shares of Y at Y_c and buying x shares of X at X_c .

The net gain for investing in Y :

$$y(Y_c - Y_0) - fee$$

The net gain for investing in X :

$$\begin{aligned}
&x(X_0 - X_c) - fee \\
&= \frac{yY_0}{X_0}(X_0 - X_c) - fee \\
&= yY_0\left(1 - \frac{X_c}{X_0}\right) - fee \\
&= yY_0\left(1 - \frac{\exp(\alpha^{-1}(\log Y_c - \beta - Z_c))}{\exp(\alpha^{-1}(\log Y_0 - \beta - Z_0))}\right) - fee \\
&= yY_0\left(1 - \exp\left(\alpha^{-1}\left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right)\right) - fee
\end{aligned}$$

The overall net gain for investing in both X and Y :

$$\begin{aligned}
&y(Y_c - Y_0) - fee + x(X_0 - X_c) - fee \\
&= y(Y_c - Y_0) + yY_0\left(1 - \exp\left(\alpha^{-1}\left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right)\right) - totalFee \\
&= yY_c - yY_0 \exp\left(\alpha^{-1}\left(\log \frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) - totalFee
\end{aligned}$$

In order to make a profit, the net gain should be greater than zero:

$$\begin{aligned}
yY_c - yY_0 \exp\left(\alpha^{-1}\left(\log\frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) - totalFee &> 0 \\
yY_0 \exp\left(\alpha^{-1}\left(\log\frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) &< yY_c - totalFee \\
\exp\left(\alpha^{-1}\left(\log\frac{Y_c}{Y_0} + Z_0 - Z_c\right)\right) &< \frac{Y_c - totalFee/y}{Y_0} \\
\log\frac{Y_c}{Y_0} + Z_0 - Z_c &< \alpha \log\left(\frac{Y_c - totalFee/y}{Y_0}\right) \\
Z_0 - Z_c &< \log\left(\frac{(Y_c - totalFee/y)^\alpha}{Y_c} Y_0^{1-\alpha}\right) \\
Z_c - Z_0 &> -\log\left(\frac{(Y_c - totalFee/y)^\alpha}{Y_c} Y_0^{1-\alpha}\right) \\
Z_c - Z_0 &> \log\left(\frac{Y_c}{(Y_c - totalFee/y)^\alpha} Y_0^{\alpha-1}\right)
\end{aligned}$$

8 Appendix C

8.1 RL Trading Agent Parameters

The batch dimension here is ignored for simplicity.

Layer Name	Dimension
Input	3
LSTM	150
concat V_t	151

Table 6: State Encoding Network

Layer Name	Dimension
Input	151
Dense-50	50
LeakyReLU	50
Dense-50	50
LeakyReLU	50
Dense-3	3
Softmax	3

Table 7: Policy Network

8.2 Baseline Methods Parameters

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.607871	0.350765	0.028614	0.030806	0.232211
2.0	0.5	-0.5	-0.526656	-0.147333	0.028767	0.017432	0.246957
2.0	0.5	-0.5	-0.294544	-0.155318	0.017320	0.017311	0.256126
2.0	0.5	-0.5	-0.459566	-1.057460	0.011530	-0.010915	0.235033

Table 8: Grid search of Distance from 2016-03-18 to 2016-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.524014	-0.675543	-0.017909	-0.010959	0.159916
2.0	0.5	-0.5	-0.453507	-0.685904	-0.024520	-0.028000	0.161680
2.0	0.5	-0.5	-0.397209	-0.635323	-0.017118	-0.015714	0.166005
2.0	0.5	-0.5	-0.483952	-0.640478	-0.024263	-0.008118	0.173431

Table 9: Grid search of Distance from 2017-03-20 to 2017-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.027708	0.412960	0.097522	0.096888	0.210586
2.0	0.5	-0.5	0.032743	0.419664	0.104805	0.105455	0.209587
2.0	0.5	-0.5	0.027216	0.451046	0.103432	0.085590	0.209949
2.0	0.5	-0.5	0.017593	0.427719	0.095741	0.090333	0.204060

Table 10: Grid search of Distance from 2018-03-19 to 2018-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.055720	0.710392	0.032070	0.067910	0.279280
2.0	0.5	-0.5	-2.460292	0.338448	0.015761	0.020786	0.274067
2.0	0.5	-0.5	-0.476017	0.473876	0.011611	0.034297	0.273283

Table 11: Grid search of Cointegration (Rolling OLS) from 2016-03-18 to 2016-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.593822	-0.600647	-0.043428	-0.021960	0.209438
2.0	0.5	-0.5	-0.364467	-0.576762	-0.033428	-0.007704	0.214335
2.0	0.5	-0.5	-0.371848	-0.599222	-0.038036	-0.015467	0.217431

Table 12: Grid search of Cointegration (Rolling OLS) from 2017-03-20 to 2017-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.5	-0.521315	-0.527849	-0.005908	-0.005320	0.211914
2.0	0.5	-0.5	-0.402114	-0.531833	-0.006107	0.001783	0.220995
2.0	0.5	-0.5	-0.359215	-0.542585	-0.009350	-0.001281	0.210937

Table 13: Grid search of Cointegration (Rolling OLS) from 2018-03-19 to 2018-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.10	-0.375979	0.577020	0.042809	0.000000	0.167329
2.0	0.5	-0.05	-0.774229	-1.046671	0.005013	0.000000	0.117561
1.0	0.5	-0.10	-0.277535	-1.051741	0.035267	0.000000	0.289717
1.0	0.5	-0.05	-0.671175	-1.106761	0.016121	-0.042197	0.288231

Table 14: Grid search of Cointegration (Kalman Filter) from 2016-03-18 to 2016-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.10	-1.061149	-0.682831	0.003595	0.0	0.076037
2.0	0.5	-0.05	-0.149293	0.111252	0.004590	0.0	0.075036
1.0	0.5	-0.10	-0.737996	-0.815322	-0.025094	0.0	0.078794
1.0	0.5	-0.05	-0.667460	-0.856972	-0.017755	0.0	0.097153

Table 15: Grid search of Cointegration (Kalman Filter) from 2017-03-20 to 2017-12-31

enter_threshold_size	exit_threshold_size	loss_limit	avg_sharpe_ratio	median_sharpe_ratio	avg_overall_return	median_overall_return	overall_return_std
2.0	0.5	-0.10	-0.158710	-0.064349	-0.018564	0.000000	0.111510
2.0	0.5	-0.05	-0.501058	-0.816271	-0.020858	0.000000	0.103610
1.0	0.5	-0.10	-1.351692	-0.374700	0.002646	0.000000	0.221902
1.0	0.5	-0.05	-1.669186	-0.809022	-0.035053	-0.004048	0.157478

Table 16: Grid search of Cointegration (Kalman Filter) from 2018-03-19 to 2018-12-31

9 Appendix D

9.1 Distribution of work

Task	Gordon	Wen Yan	Brendan
Perform Literature Survey	A	A	L
Write Proposal Report	A	A	L
Design Spread Models	A	L	A
Optimize Trade Execution Parameters	A	L	A
Write Monthly Reports	A	A	L
Design Pair Selection Methods	L	A	A
Connect to Broker API	L	A	A
Implement Backtesting	L	A	A
Implement Trade Execution Parameters Optimization	A	L	A
Implement Pair Selection Methods	A	L	A
Implement Spread Models	A	L	A
Backtesting Profit of strategy	L	A	A
Test Pair Selection Methods	A	A	L
Test Spread Models	A	A	L
Test Trade Execution Parameters Optimization	A	L	A
(Optional) Extend Pairs spread model to basket model	L	A	A
Design Data Visualization	A	A	L
Design User Interface	A	A	L
Implement Data Visualization	A	A	L
Implement Design User Interface	L	A	A
Write Progress Report	A	A	L
Write Final Report	A	A	L
Design Project Poster	A	A	L
Prepare Presentation Deck	A	A	L

Figure 38: Distribution of Work

9.2 GANTT Chart

Task	Sep-18	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19
Perform Literature Survey								
Write Proposal Report								
Design Spread Models								
Design Optimization of Trade Execution Parameters								
Write Monthly Reports								
Design Pair Selection Methods								
Implement connection to Broker API								
Implement Backtesting								
Implement Trade Execution Parameters Optimization								
Implement Pair Selection Methods								
Implement Spread Models								
Design Backtesting Profit of strategy								
Test Pair Selection Methods								
Test Spread Models								
Test Trade Execution Parameters Optimization								
Implement Reinforcement Learning Trading Agent								
Design Data Visualization								
Design User Interface								
Implement Data Visualization								
Implement Design User Interface								
Write Progress Report								
Write Final Report								
Design Project Poster								
Prepare Presentation Deck								

Figure 39: Gantt Chart