1 Deep Reinforcement Learning and PCA Approach in Pairs Trading

Howard Jin, Zhanyi Lin, Congcong Ma & Minghui Zhu

In this project, we use advanced data science techniques to enhance and optimize the pairs trading strategy. We investigate the use of Reinforcement Learning (RL) as an agent for decision-making and determining optimal enter-exit thresholds, as well as exploring the application of Principal Component Analysis (PCA) for constructing an optimal portfolio. The pairs trading strategy showed strong performance during in-sample and out-of-sample backtesting periods, with high Sharpe ratios and relatively low drawdowns. The strategy also performed well during a blind out-of-sample period, indicating its robustness and stability. The outcomes of this study could have practical implications for traders and investors aiming to improve their returns while minimizing risks.

1 Introduction

In this project, we aim to enhance and optimize the pairs trading strategy using advanced data science techniques. Specifically, we plan to investigate the applicability of Reinforcement Learning (RL) as an agent for determining optimal enter-exit thresholds and simulating decision-making. Furthermore, we will explore the use of Principal Component Analysis (PCA) for constructing an optimal portfolio. The results of this study can potentially have practical implications for traders and investors seeking to enhance their returns while minimizing risks.

1.1 Strategy Overview

1.1.1 Pairs Trading

Pairs trading is a widely used strategy in finance for exploiting the relative mispricing of two highly correlated assets. The approach involves identifying a pair of securities that have a long-term relationship but whose prices have temporarily diverged. The trader then buys the undervalued security while simultaneously selling the overvalued one, aiming to profit from the convergence of their prices. The goal of pairs trading is to profit from the price difference between the two assets, rather than trying to predict the direction of the market. This strategy has gained popularity in recent years because it allows traders to hedge their positions against market volatility and reduce their exposure to overall market risk. The logic of pairs trading strategy is illustrated in Figure 1.

Evidence indicates that pairs trading is a profitable approach in a wide range of markets, including equities, commodities, and currencies. In the stock market, for example, a pairs trader would find two strongly correlated equities, such as Coca-Cola (KO) and PepsiCo (PEP), and then buy the underpriced one and sell the overvalued one. Similarly, a trader can select two commodities that are highly correlated, such as gold and silver, and then buy the undervalued commodity while shorting the overvalued one. Pairs traders may find two strongly correlated currency pairs, such as EUR/USD and GBP/USD, and then buy the undervalued currency while selling the overvalued one in the currency market. This strategy make profits when two assets converge.

Despite its popularity, pairs trading can be challenging to implement, as traders need to identify the optimal entry and exit thresholds, construct an optimal portfolio, and deal with transaction costs and liquidity issues. To address these challenges, we propose applying state-of-the-art techniques, such as Reinforcement

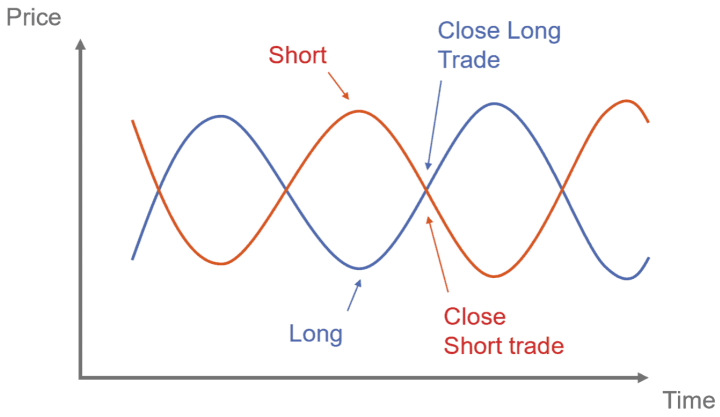

Figure 1: Illustration of Pairs trading
(Source: https://algotrading101.com/learn/pairs-trading-guide/)

Learning (RL) and Principal Component Analysis (PCA), to improve and optimize the pairs trading strategy.

1.1.2 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning which trains an agent to interact with an environment with a goal of maximizing the reward. RL algorithms are trial-and-error based learning, where the agent takes actions in the environment to maximize its expected future reward. The agent receives feedback in the form of rewards or penalties based on evaluating its actions, helping it learn and improve its decision-making over time.

RL has been successfully applied in various real-world applications, such as robotics, gaming, and autonomous vehicles. For example, RL can train robots to perform complex tasks including grasping objects, playing games like Go and Chess, and driving autonomous vehicles.

In the context of pairs trading, RL can be applied to determine the optimal entry and exit thresholds for trading the pairs. The RL agent learns from historical price data and adjusts the trading thresholds to maximize its expected future reward, which is the profit obtained from trading the pairs. By using RL, traders can automate the decision-making process and improve the performance of the pairs trading strategy.

Furthermore, RL can be used to simulate the decision-making process of pairs traders. The algorithms can test and refine their strategies without incurring real-world trading costs (but considering the trade costs in learning process). This simulation can be used to evaluate the performance of different trading strategies under various market conditions, which can inform traders' decision-making and improve their overall trading performance.

Overall, our first try is to implement an RL agent to optimize the pairs trading strategy, and evaluate its performance. By applying RL, we aim to automate the decision-making process, improve the accuracy of the trading thresholds, and enhance the profitability of the pairs trading strategy.

1.1.3 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while retaining as much information as possible. It works by transforming the original variables into a new set of variables called principal components. These new variables are linear combinations of the original variables and are ordered in terms of the amount of variance they explain in the data. The first principal component explains the most variance, the second explains the second most variance, and so on.

PCA has various applications in finance, including in pairs trading strategies. In pairs trading, the goal is to find two securities that are co-integrated, which means they move together over time. One way to find co-integrated securities is to use PCA to identify a common factor that influences the prices of the securities. This common factor can be interpreted as the spread between the two securities.

Once the spread has been identified, pairs traders can create a trading strategy by going long on the underpriced security and shorting the overpriced security. This allows them to profit from the convergence of the spread back to its long-term average.

Overall, PCA is a powerful tool for identifying patterns in financial data and can be used to develop trading strategies that capitalize on these patterns. By reducing the dimensionality of the data and identifying common factors, PCA can help traders find profitable opportunities in complex financial markets.

1.2 Market Background

The equity market is the target domain for our trading strategy in this study. We concentrate on trading the Apple (AAPL) and Microsoft (MSFT) pair for our RL research as we are investigating enter or exit thresholds. For our PCA research, we employ the statistical techniques developed by Avellaneda, Marco Jeong-Hyun Lee to create a market-neutral eigenportfolio.

1.3 Main Results

The backtesting results for the pairs trading strategy were generally positive, with each period generating positive returns and demonstrating a favorable Sharpe ratio. The in-sample period from January 1, 2017 to January 1, 2022 resulted in a return of 71.492%, while the first out-of-sample period from January 1, 2022 to November 1, 2022 generated a return of 12.829%. The second out-of-sample period from January 1, 2016 to January 1, 2017 resulted in a compounding annual return of 5.550%. The blind out-of-sample period from December 23, 2022 to March 24, 2023 generated a return of 11.505%. This strategy successfully passed the stress test with an annual return of 4.846% and a sharpe ratio of 1.386. Overall, the strategy exhibited a relatively low drawdown with a high Sharpe ratio and PSR, indicating that it performed well during these periods. The results suggest that the pairs trading strategy was able to withstand market fluctuations and maintain its value.

2 Theoretical Background

2.1 RL Literature

In the learning process of reinforcement learning, an agent interacts with the environment, where the agent's actions can affect the environment, and the environment evaluates the agent's action and gives rewards. It involves several key concepts of RL: agent, environment, state, action, and reward, which are depicted in Figure 2.

2.1.1 Concepts

Agent

An agent is an entity that interacts with the environment to achieve a specific goal. In the learning algorithm, the agent makes decisions and takes actions based on its observations and the rewards it receives from the environment. In pairs trading, the agent is the RL algorithm that learns to identify the optimal entry and exit points for the trading strategy based on historical market data.

Environment

The environment is the external system or world in which the agent operates. It provides the agent with observations and rewards based on its actions. In pairs trading, the environment is the financial market, including the prices and trends of the two assets being traded.

State

A state is a representation of the current situation or observation of the environment. It provides the agent with information about the current market conditions. In pairs trading, the state includes the current portfolio and position, current prices and trends of the two assets being traded.

Action

An action is the decision made by the agent to perform a specific operation based on the current state of the environment and policy learned. In pairs trading, the action is the decision to enter or exit a trade or hold, based on the current state of the financial market.

Reward

A reward is the feedback given to the agent by the environment after it takes an action. It is used to reinforce or discourage certain actions in the future. In pairs trading, the reward is the profit or loss generated by the trade, which is used to reinforce or discourage certain trading decisions in the future.

2.1.2 Deep Q-Network Learning (DQN)

Deep Q-Network (DQN) is an extension of the Q-learning algorithm that employs deep neural networks to approximate the Q-function. In DQN, the Q-function is a deep neural network, which takes in the state of the environment and outputs the Q-values for each possible action and stores in a Q-table. To be specific, it maps a state-action pair to a scalar value, representing the expected long-term reward of taking that action in that state and following a particular policy thereafter.

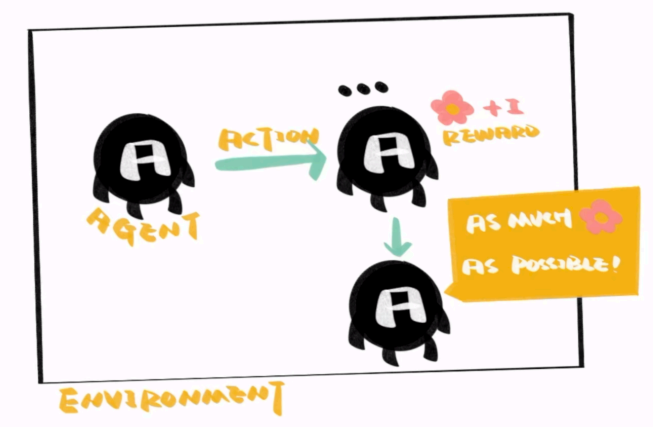

Figure 2: Illustration of Reinforcement learning
(Source: @KnowingAI)

The Q-function is updated iteratively during the training process using a technique called temporal-difference learning. The goal is to find the optimal Q-function that maximizes the expected cumulative reward for the agent. This is achieved by repeatedly estimating the Q-values for state-action pairs, and updating the Q-function parameters based on the difference between the predicted and actual Q-values. Algorithm 1 from Thoma (2015) describes how it works.

DQN has been shown to be effective in solving complex decision-making problems in various domains, including robotics and gaming. In finance, DQN has been applied to pairs trading to improve the decision-making process of the trading agent. By training a DQN on historical data, the agent can learn to make better decisions about which pairs to trade and when to enter and exit positions. The neural network is trained to optimize a certain objective function, such as maximizing profits or minimizing risk, by adjusting the weights of its connections through backpropagation. The resulting policy can then be used to guide the trading decisions of the agent.

2.1.3 Advantages of DRL in Pairs Trading

Previous literature has discussed the advantages of DRL in pairs trading.

Model-free DRL optimizes overall profit directly

First, DRL-based methods optimize overall profit directly by learning a policy that maps market data to trading actions that maximize profits (Han et al. 2023).

Algorithm 1 Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:
States $\mathcal{X} = \{1, \dots, n_x\}$
Actions $\mathcal{A} = \{1, \dots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLearning($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Start in state $s \in \mathcal{X}$
    while s is not terminal **do**
      Calculate π (e.g. $\pi(x) \leftarrow_a Q(x, a)$)
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow T(s, a)$ ▷ Receive the new state
      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
      $s \leftarrow s'$
    end while
  end whilereturn Q
end procedure

This is in contrast to traditional approaches that involve predicting future prices or returns as an intermediate step, which can be challenging to modeling due to the complex and unpredictable nature of financial markets. The complex and non-linear relationship between two assets may not be apparent for human reader. However, DRL-based methods are model-free, meaning that they do not require priori assumptions about the underlying structure of the market or the nature of the trading strategy, making it can potentially capture exploit more nuanced and subtle patterns in the market data (Sato 2019). This makes DRL-based methods more flexible and adaptable to different market conditions and trading scenarios.

By directly optimizing for profits, DRL-based methods can potentially lead to more effective and profitable trading strategies. This is because the agent learns to take into account not only the current state of the market, but also the potential consequences of its actions on future profits. Furthermore, DRL-based methods can learn to balance risk and reward, taking into account factors such

as transaction costs and market volatility.

Dynamically interaction

In dynamic market conditions, traditional trading strategies may struggle to adapt to rapidly changing market conditions. However, DRL-based methods can adapt to changing market conditions by continuously learning from new market data (Brim 2020). This allows traders to identify and exploit profitable opportunities in real-time, even in highly volatile market conditions. Additionally, DRL-based methods can learn to adjust trading strategies based on changing market conditions, such as changes in asset correlations or shifts in market sentiment. This adaptability is especially important in dynamic markets, where the ability to quickly adapt to changing market conditions can make the difference between a profitable and unprofitable trading strategy.

2.2 PCA Literature

According to Avellaneda & Lee (2010), in the context of statistical arbitrage, the typical characteristics include:

- (i) utilizing systematic or rule-based trading signals instead of fundamental analysis.
- (ii) maintaining a market-neutral trading portfolio, with a beta of zero relative to the broader market.
- (iii) relying on statistical methods to generate above-average returns.

In other words, the strategy involves making numerous trades that are expected to yield positive returns while leveraging the benefits of diversification across various stocks. The objective is to create a low-volatility investment approach that is not correlated with the broader market.

When employing pairs trading as a strategy, a fundamental assumption is based on mean reversion, which means that the spread time series between two assets should exhibit *stationarity*, indicating that any significant deviations from their long-term relationship will ultimately be corrected. Essentially, pairs trading relies on the assumption that the spread between two assets will eventually return to its historical mean, allowing traders to profit from short-term discrepancies in the relationship between the two assets.

To study market-neutrality, the Principal Component Analysis is utilized to extract risk factors. Let $\{R_i\}_{i=1}^{N}$ denote the returns of N different stocks in the trading

universe. Then, we can decompose stock returns into a systematic component, which consists of m systematic risk factors, and an (uncorrelated) idiosyncratic component $\tilde{R}_i$, that is,

$$R_i = \sum_{j=1}^{m} \beta_{ij} F_j + \tilde{R}_i$$

where the collection of F_j's can be thought of as the returns of "benchmark" portfolios representing systematic factors.

Besides, the trading portfolio is considered market-neutral if the dollar amounts $\{Q_i\}_{i=1}^{N}$ invested in each of the stocks are such that

$$\bar{\beta}_j = \sum_{i=1}^{m} \beta_{ij} Q_i = 0, j = 1, 2, ..., m.$$

Here, the coefficients β_j, which correspond to the projections of the portfolio returns on the different factors, demonstrates that the market-neutral portfolio has vanishing portfolio betas. Also, it can be proved that

$$\sum_{i=1}^{N} Q_i R_i = \sum_{i=1}^{N} Q_i \tilde{R}_i$$

which implies that the market-neutral portfolio is only affected by idiosyncratic returns. Then, the question comes to how to extract systematic risk factors from data.

2.2.1 Apply PCA on the empirical correlation matrix

For a cross-section of N stocks, we consider a lookback period of M days, and construct the stock return matrix on a date t_0 going back $M+1$ days using historical price data, where each entry can be represented as

$$R_{ik} = \frac{S_{i(t_0-(k-1)\Delta t)} - S_{i(t_0-k\Delta t)}}{S_{i(t_0-k\Delta t)}}; k = 1, ..., M; i = 1, ..., N$$

After doing the standardization procedure on the stock return matrix, we can obtain the standardized returns, which are calculated as:

$$Y_{ik} = \frac{R_{ik} - \bar{R}_i}{\bar{\sigma}_i}$$

as well as the empirical correlation matrix:

$$\rho_{ij} = \frac{1}{M-1} \sum_{k=1}^{M} Y_{ik} Y_{jk}$$

Eventually, we consider the eigenvectors and eigenvalues of the empirical correlation matrix and rank the eigenvalues in decreasing order:

$$N \geq \lambda_1 \geq \lambda_2 ... \geq \lambda_N \geq 0$$

The corresponding eigenvectors are denoted by

$$v^{(j)} = (v_1^{(j)}, ..., v_N^{(j)}), j = 1, ..., N$$

2.2.2 Eigenportfolio Creation

Let index **j** refer to the j^{th} eigenportfolio, with the respective amounts invested in each of stocks in the universe being

$$Q_i^{(j)} = \frac{v_i^{(j)}}{\bar{\sigma}_i}$$

Thus, the eigenportfolio returns are:

$$F_j = \sum_{i=1}^{N} Q_i^{(j)} R_{ik} = \sum_{i=1}^{N} \frac{v_i^{(j)}}{\bar{\sigma}_i} R_{ik}; j = 1, ..., m$$

Essentially, each eigenportfolio return is the weighted average of the each empirical stock return, with the weights being equal to $Q_i^{(j)}$. In this case, the collection of m eigenportfolios constitutes a market neutral portfolio.

2.2.3 Modelling the Idiosyncratic Component

After decomposing each stock return through the PCA approach by obtaining a natural set of risk factors and formulating a market neutral portfolio, we proceed to model the idiosyncratic component, which is $\tilde{R}_i = R_i - \sum_{j=1}^{m} \beta_{ij} F_j$

Here, the paper first assumes that the residual satisfies:

$$d\tilde{X}_i(t) = \alpha_i dt + dX_i(t)$$

Specifically, α_i represents the drift of the idiosyncratic component, which is equivalent to say that $\alpha_i dt$ is the excess rate of return of the stock in relation to the overall market. On the other hand, $dX_i(t)$ represents the increment of a stationary stochastic process which models price fluctuations corresponding to idiosyncratic fluctuations in the stock price which are not reflected by the overall market. Overall, the model assumes a drift which measures systematic deviations from the market and a price fluctuation that is mean-reverting to the overall industry level. Based on these considerations, the Ornstein-Uhlenbeck process is introduced:

$$dX_i(t) = \kappa_i(m_i - X_i(t))dt + \sigma_i dW_i(t), \kappa_i > 0$$

where $\alpha_i, \kappa_i, m_i, \sigma_i$ are specific to each stock. In addition, they are assumed to vary slowly in relation to the increments $dW_i(t)$. As a result, the authors assume that the parameters are constant over the time frame, and the strategy only considers the stocks with high speed of mean reversion (κ).

2.2.4 Trading Signal Generation

Define the S-score as:

$$s_i = \frac{X_i(t) - m_i}{\sigma_{eq,i}}, where \sigma_{eq,i} = \frac{\sigma_i}{\sqrt{2\kappa_i}}$$

The S-score measures how far away a given asset eigenportfolio is from the theoretical equilibrium value associated with the model.

Finally, the trading signal based on mean reversion is:

$$\text{buy to open if } s_i < -\bar{s}_{bo}$$
$$\text{sell to open if } s_i > +\bar{s}_{so}$$
$$\text{close short position if } s_i < +\bar{s}_{bc}$$
$$\text{close long position if } s_i > -\bar{s}_{sc}$$

Based on the simulating strategies from 2000 to 2004 in the case of ETF factors, the cutoffs $\bar{s}_{bo}, \bar{s}_{so}, \bar{s}_{bc}, \bar{s}_{sc}$ are estimated empirically to be:

$$\bar{s}_{bo} = \bar{s}_{so} = 1.25$$

$$\bar{s}_{bc} = 0.75 \text{ and } \bar{s}_{sc} = 0.5$$

3 Trading Strategy

3.1 Implementation of RL

3.1.1 Environment of Reinforcement Learning

Reinforcement learning algorithms consist of two major components, the environment and the agent. This subsection will describe the environment of our algorithm and the training data set.

The environment could be further divided into states, actions and rewards. States are the information where the agent used to make decisions. Actions are the the set of decisions the agent could make at each states. Rewards are the payoff of each action at each state. The purpose of reinforcement learning is basically training an optimal policy mapping from states to actions so as to maximize the total expected payoffs.

In pairs trading, one common entry and exit criterion is the zscore of regression errors. Specifically, Given a pair (P, Q), one could run the following regression equation:

$$\log P_t = \alpha + \beta \log Q_t + \epsilon_t$$

where ϵ_t captures the spread the P and Q. Traditional pairs trading enters the trade when ϵ_t becomes abnormal, such as having a zscore greater 2. Our reinforcement learning takes similar inputs as our states. Specifically, after we regressing $\log P_t$ on $\log Q_t$, we obtain a series of ϵ_t. We then transform ϵ_t into a series of zscores. And then take the past 5 days history of zscores as our states.

Our reinforcement learning takes only three actions: hold ($a = 0$), buy ($a = 1$) and sell ($a = 2$). Given a pair (P, Q), *buy* indicates long P and short Q. *Sell* indicates short Q and long P. The long short weights are set to be $\frac{1}{1+\beta}$ for P and $\frac{\beta}{1+\beta}$ for Q so that the PnL for this pairs trading strategy is theoretically $\frac{1}{\beta+1}\Delta\epsilon_t$.[1]

The reward we use the next-day return of the action taken by the agent minus trading cost. For instance, at time $T = t$, our state is the zscores of residuals from z_{t-4} to z_t. Then the agent will take an action based on trained policy. The reward of this action is given by the stock return at time $t + 1$ ($\log P_{t+1} - \log P_t$) minus a fixed trading cost (1%).

[1] In backtesting, this might not be the case because the estimation of β changes over time.

3.1.2 Training Data

For the RL part, we choose Apple and Microsoft as our pairs. Trading periods from 2017-1-1 to 2022-1-1 are chosen to train our agents. As mentioned before, we run a log linear regression and then feed in the series of standardized residuals to our model. The agent will take the past 5 days of standardized residuals as its states, make a trading decision and then get the rewards. The algorithm will try to train a policy that maximizes the total discounted rewards.

3.1.3 Implementation and Implication

The algorithm we choose for training the model is Deep Q Network (DQN). The training is implemented with the python package `stable_baseline3`. We then test the validity of our model in different backtesting period. For each backtesting period $T = t$, we first use 90 days as lookback to run a log linear regression as above. Then our trained model will take the past 5 days of standardized residuals as inputs and outputs a trading decision. Finally, we execute this decision directed by the model.

One important implementation detail is that whenever we open the position at time t, we would close the position at time $t+1$. This implementation is driven by two reasons. First, it helps reduce the market risk exposure. Our strategy always exposes to the market risk at most 1 day, which helps in avoiding large fluctuations and divergence of our pairs. Second, this implementation is more aligned with our reinforcement learning model, where the reward of an action is given by tomorrow's stock return.

This implementation actually differentiate us from the trading pairs trading strategy. where people enter the trade and wait until the spread converges. One interpretation of our model is that our model will take a trading decision whenever it is sufficiently confident about the next day spread. In this sense, this trading strategy could be viewed as a combination of pairs trading and momentum-like strategy, where we try to track the trends of the spread instead of waiting the spread to converge.

3.2 Implementation of PCA Eigenportfolio Framework for a Market/Sector-Neutral Trading Strategy

As part of our comprehensive market/sector-neutral trading strategy, this section will delve into the implementation of the Principal Component Analysis

(PCA) eigenportfolio framework. PCA is a statistical technique that reduces the dimensionality of a dataset while preserving most of its variance. In the context of a trading strategy, PCA allows us to identify the common market factor that drives stock returns, enabling us to construct a market/sector-neutral portfolio.

The following steps outline the implementation of the PCA eigenportfolio framework:

1. **Data Collection and Preprocessing:**
    a) Collect historical stock price data for your universe of assets. Ensure that the dataset covers a sufficiently long period to capture the common market factor and other relevant variations.
    b) Preprocess the data by converting stock prices to returns, calculating the logarithmic differences between consecutive prices. This step ensures that the data becomes more stationary and homoscedastic.
    c) Normalize or standardize the stock returns to obtain comparable scales and eliminate any disproportionate influence of individual stocks on the analysis.
2. **Applying PCA to the Preprocessed Data:**
    a) Compute the covariance matrix of the preprocessed stock returns. The covariance matrix quantifies the degree to which the returns of different stocks move together.
    b) Apply PCA to the covariance matrix to obtain the eigenvectors and eigenvalues. The eigenvectors represent the principal components (PCs), while the eigenvalues determine the variance explained by each PC.
    c) Sort the eigenvectors and eigenvalues in descending order based on the magnitude of the eigenvalues. This ordering ensures that the first principal component (PC1) explains the maximum amount of variance in the dataset.
3. **Constructing the Market/Sector-Neutral Portfolio:**
    a) Retain the first principal component (PC1) as the market/sector-neutral portfolio. PC1 represents the common market factor driving the returns of the stocks in the dataset. By investing in a portfolio that

mimics PC1, we can isolate the market component and focus on the idiosyncratic risk and return of individual stocks.

b) Compute the portfolio weights for the stocks by normalizing the eigenvector corresponding to PC1. These weights determine the proportion of each stock in the market/sector-neutral portfolio.

c) Continuously monitor and update the PCA eigenportfolio as new data becomes available. This step ensures that the portfolio remains adaptive and responsive to changing market conditions.

Incorporating the PCA eigenportfolio framework into the market/sector-neutral trading strategy allows us to isolate the common market factor and focus on stock-specific return drivers. In combination with cointegration methods and fundamental data-based selection, as described in sections 3.1.2 and 3.1.3, the PCA eigenportfolio framework contributes to a more sophisticated and robust trading strategy with the potential for improved long-term profitability and risk-adjusted performance.

3.3 Capital Allocation & Risk Management

In all our in-sample and out-of-sample backtesting, we set an equal weight for both strategies. The proportion of this two strategies will be dynamically adjusted to be maintained at 25% of the total portfolio values respectively.

These numbers are chosen for the following reasons. First, even though there exist better capital allocation algorithms or ideas such as Markowitz Mean-Variance Model or Kelly's principle, the variance and Sharpe ratio between these two strategies vary significantly among our backtesting period. There is no a consistent number for us to compute the optimal weights for every period. Secondly, we set aside 50% of cash to reduce our leverage ratio. The default leverage ratio of Quantconnect is 2. Therefore setting aside 50% of our portfolio value helps force the total leverage ratio to be 1. Finally, since our strategies will long and short a large amount of stocks simultaneously, holding cash aside helps in satisfying the margin requirements.

Limiting the total weights of our strategies is also a consideration for risk management. By avoiding **insufficient buying power** and **margin call** warnings, we ensure that the our liquidation is all directed by the algorithm instead of the broker.

The other risk management technique we use is to monitor the level of VIX , which is a measure of the expected volatility of the S&P 500 index. The VIX is calculated using options prices on the S&P 500 index. Specifically, it represents the market's expectation of 30-day volatility, derived from the prices of SP 500 index options. When the VIX is high, it suggests that investors expect the stock market to be more volatile in the coming weeks, and when it is low, it suggests that investors expect less volatility. Our strategy is set to liquidate the position whenever VIX is abnormally large and our portfolio is suffering loss. In other words, we impose a stop-loss criterion whenever the VIX is greater than 40.

The subset of stocks offers several advantages over using the entire index, single-sector stocks, or other indices like the NASDAQ Composite. Key benefits identified in this study include targeted diversification, the customization of the subset to match investor risk tolerance or investment objectives, and cost efficiency.

- **Targeted Diversification**: The selected subset of the S&P 500 provides balanced exposure to specific sectors or industries, allowing for a more focused diversification strategy. This approach enables investors to spread their investments across a range of carefully chosen companies, reducing the impact of underperformance or downturns in any single sector or company. This targeted diversification is superior to investing in single-sector stocks or the entire index, as it allows for a more tailored risk management approach.
- **Customization to Match Investor Risk Tolerance or Investment Objectives**: The subset stock pool can be adjusted to align with an investor's risk tolerance or investment objectives. For example, a more conservative investor may prefer a subset consisting of blue-chip or dividend-paying stocks, while a more aggressive investor may seek high-growth or small-cap companies. This customization enables investors to balance risk and potential returns according to their preferences, potentially leading to a more stable return profile than the entire S&P 500 or other indices.
- **Cost Efficiency**: Investing in a subset of the S&P 500 index can be more cost-efficient than focusing on single-sector stocks or other indices. Numerous index funds and exchange-traded funds (ETFs) are designed to track the performance of specific subsets of the S&P 500, providing investors with easy access to a diversified portfolio at a lower cost. Passive investment products typically have lower management fees and transaction costs compared to actively managed funds or investing in individual stocks. Moreover, using a subset of the S&P 500 as a stock pool for pair

trading stock selection allows investors to take advantage of the cost efficiency associated with passive investing and avoid the higher costs and potential underperformance of active stock picking.

In conclusion, managing risk through the use of a subset stock pool derived from the S&P 500 index, as identified by the DBSCAN clustering algorithm, offers several advantages in pair trading stock selection. These advantages include targeted diversification, the customization of the subset to match investor risk tolerance or investment objectives, and cost efficiency. This data-driven approach can lead to more effective capital allocation and risk management in pair trading strategies.

4 Backtesting Results

4.1 Performance Metrics

When evaluating the success of a trading strategy, it is critical to employ proper performance metrics to determine the strategy's effectiveness. In pairs trading, as with any other trading strategy, there are several key metrics that are commonly used to evaluate the performance of the strategy.

4.1.1 Return and PnL

Return is a fundamental performance metric that measures the profitability of the strategy. This is typically expressed as a percentage of the initial investment or the total capital invested. For pairs trading, the return can be measured using the profit and loss (PnL) generated by the strategy over a given period.

However, the return metric alone does not provide a comprehensive picture of the performance of a trading strategy. It is important to consider other factors such as risk-adjusted returns and drawdown when evaluating the effectiveness of a pairs trading strategy. This is where the Sharpe ratio and drawdown come in.

4.1.2 Sharpe Ratio

The Sharpe ratio measures the risk-adjusted return of a trading strategy by taking into account the volatility of the returns. This metric provides a measure of the excess return generated by the strategy per unit of risk taken. A higher Sharpe

ratio indicates a better risk-adjusted return, and is generally considered to be a desirable characteristic of a trading strategy. The formula is stated as:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p} \tag{1.1}$$

where R_p is the average rate of return of the portfolio, R_f is the risk-free rate of return, and σ_p is the standard deviation of the portfolio returns.

4.1.3 Drawdown

Drawdown, on the other hand, measures the peak-to-trough decline in the equity curve of the trading strategy. This metric helps traders to understand the maximum loss they may face while executing the strategy, and can help in managing risk and setting appropriate stop-loss levels. A lower drawdown indicates a less risky trading strategy.

$$Maximum\ Drawdown = \frac{Peak\ Value - Trough\ Value}{Peak\ Value} \tag{1.2}$$

Drawdown can lead to margin issues in a pairs trading strategy when losses reduce the trader's account balance below the required margin level, resulting in a margin call from the broker. A margin call demands additional funds to meet the minimum margin requirement, and failure to comply may result in the broker closing some or all of the trader's positions to minimize the risk of further losses. Proper risk management and monitoring of margin levels can help traders mitigate the risk of margin calls and optimize their pairs trading strategy's performance.

4.2 Backtesting Results

4.2.1 In-Sample Data Backtest

Figure 3: In-Sample Period Backtesting: 2017-01-01 to 2022-01-01

From January 1, 2017 to January 1, 2022, the backtesting for the in-sample period (demonstrated in Figure 3) resulted in a return of 71.492%. During this period, the strategy's Sharpe ratio was 0.778, while the drawdown was 26.100% and the PSR was 26.807%.

4.2.2 Out-of-Sample Data Backtest

Figure 4: Out-of-Sample Period Backtesting: 2022-01-01 to 2022-11-01

The first out-of-sample backtesting period from January 1, 2022 to November 1, 2022 resulted in a return of 12.829%, which corresponds to a compounding annual return of 15.542%. During this period, the strategy's drawdown was 2.300% and the Sharpe ratio was 1.59. The PSR for the period was 71.376%. Details of the trend is showed in Figure 4.

Overall, the strategy performed well during this period, generating positive returns and exhibiting a relatively low drawdown with a high Sharpe ratio and PSR.

Figure 5: Out-of-Sample Period Backtesting: 2016-01-01 to 2017-01-01

Figure 5 illustrates the second out-of-sample backtesting period from January 1, 2016 to January 1, 2017, resulting in a compounding annual return of 5.550%. During this period, the strategy's PSR was 33.021% and the Sharpe ratio was 0.644. The drawdown for the period was 7.700%.

4.2.3 Blind Out-of-Sample Data Backtest

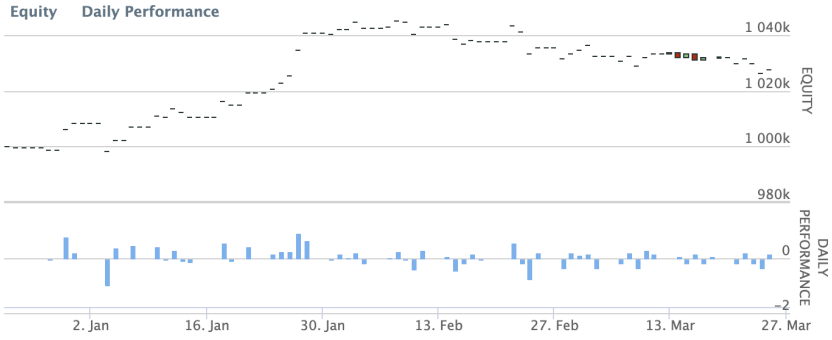

Figure 6: Blind Out-of-Sample Period Backtesting: 2022-12-23 to 2023-03-24

During the blind out-of-sample backtesting period from December 23, 2022 to March 24, 2023, the strategy demonstrated a Sharpe ratio of 1.877, indicating a

favorable risk-adjusted return. The Compounding Annual Return over the period was 11.505%, which represents a significant increase in value. the strategy experienced a drawdown of 1.800%, This indicates that the trading strategy implemented did not experience a significant decline in value from its peak to its trough. This is a positive indicator of the stability and robustness of the strategy, as it suggests that the strategy was able to withstand market fluctuations and maintain its value. Overall, these results suggest that the pairs trading strategy performed well during this period.

4.2.4 Stress Test Backtest

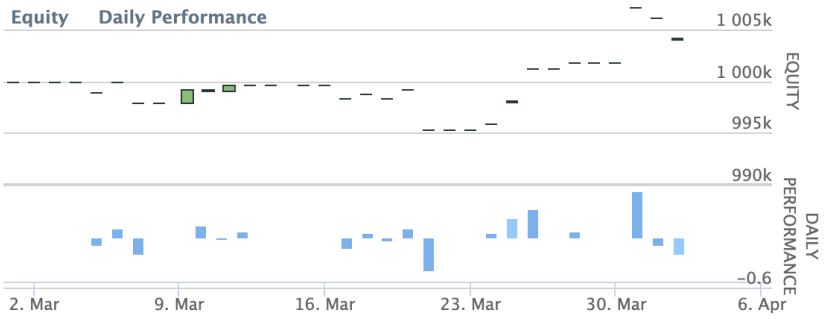

Figure 7: Stress Test: 2020-03-01 to 2020-04-02

A stress test was conducted on the trading strategy during the period from March 1, 2020 to April 2, 2020, which coincided with the COVID-19 pandemic. The results showed a PSR of 55.150%, Sharpe Ratio of 1.386, Compounding Annual Return of 4.846%, and a very small drawdown of 0.500%. The stress test demonstrates the robustness of the trading strategy during a volatile period in the market. The results suggest that the strategy was able to perform well even during a challenging market environment. A graph depicting the stress test results is shown in Figure 7.

4.2.5 Living Paper Trading

We don't have any trade in the recent living paper trading period.

4.2.6 Summary

Table 1: Performance Metrics of the Trading Strategy

Period	Timeline	Sharpe Ratio	Annual Return	Drawdown
In-Sample	01/01/2017 - 01/01/2022	0.778	17.87%	26.10%
Out-of-Sample 1	01/01/2022 - 11/01/2022	1.59	15.54%	2.30%
Out-of-Sample 2	01/01/2016 - 01/01/2017	0.664	5.55%	7.70%
Blind Out-of-Sample	12/23/2022 - 03/24/2023	1.877	11.51%	1.80%
Stress Test	03/01/2020 - 04/02/2020	1.386	4.846%	0.50%

5 Limitation and Future Work

5.1 Overfitting

One potential problem of application of RL in pairs trading is overfitting (Kim & Kim 2019). Overfitting occurs when the agent learns to fit the training data too closely, and as a result, it fails to generalize well to new, unseen data. This can lead to poor performance in real-world trading scenarios, where the market conditions may differ from those seen during training.

One reason of overfitting is that pairs trading data can be complex and high-dimensional, with many potential features and patterns that may not be relevant to the actual trading performance. When the agent is trained on such data, it may learn to exploit spurious correlations or patterns that are specific to the training data, rather than learning a more generalizable trading strategy.

For the issue of overfitting, we plan to research on Wang et al. (2021)'s work, which focuses on utilizing the deuling DQN algorithm and suggested a reward shaping technique that incorporates a baseline policy with a set trading boundary to guide the agent in learning a robust policy and prevent overfitting while having the potential to generate greater returns and Sharpe ratios than traditional approaches.

5.2 Possible Improvements of Reinforcement Learning

First, better approaches could be used to train the reinforcement learning models. In our project, we design the reward to be the next day's return so that our agent could get effective feedback from the action. This design helps in training an effective model but deviates from the target of real trading. Ultimately, we hope

to build a model that maximizes our terminal payoffs instead of a series of daily returns.

The second direction of improvements is to discard the reinforcement learning itself. Knowing that the agent performs a momentum-like strategy under the hood, we could directly the model the spread series. For instance, using time series models or neural networks to track the trend of the spread and the enter the trade whenever the model has enough predictive confidence.

Acknowledgements

We express our gratitude to Dr. David Ye and TA Pranay Jain for their continuous and timely feedback during the project, as well as for the support they offered us.

Bibliography

Avellaneda, Marco & Jeong-Hyun Lee. 2010. Statistical arbitrage in the us equities market. *Quantitative Finance* 10(7). 761–782.

Brim, Andrew. 2020. Deep reinforcement learning pairs trading with a double deep q-network. In *2020 10th annual computing and communication workshop and conference (ccwc)*, 0222–0227.

Han, Weiguang, Boyi Zhang, Qianqian Xie, Min Peng, Yanzhao Lai & Jimin Huang. 2023. *Select and trade: towards unified pair trading with hierarchical reinforcement learning*.

Kim, Taewook & Ha Young Kim. 2019. Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity* 2019. 1–20.

Sato, Yoshiharu. 2019. Model-free reinforcement learning for financial portfolios: a brief survey. *arXiv preprint arXiv:1904.04973*.

Thoma, Martin. 2015. *Q-learning pseudocode in LaTeX*. https://github.com/MartinThoma/LaTeX-examples/blob/master/source-code/Pseudocode/q-learning/q-learning.tex. Accessed: April 26, 2023.

Wang, Cheng, Patrik Sandås & Peter Beling. 2021. Improving pairs trading strategies via reinforcement learning. In *2021 international conference on applied artificial intelligence (icapai)*, 1–7. DOI: 10.1109/ICAPAI49758.2021.9462067.