ISYE 6767 Final Report

12/15/2023

Mitchell Kramer mkramer37

Project Report: Statistical Arbitrage Strategy Implementation

Executive Summary: This report documents the implementation and evaluation of a statistical arbitrage strategy based on the methodology proposed by Avellaneda and Lee (2010). The strategy focuses on a universe of 40 cryptocurrencies, selected based on market capitalization, using hourly price data over a period from February 19, 2021, to September 26, 2022. The main objective is to explore the performance of a stat-arb strategy, specifically a pairs trading strategy within the cryptocurrency market, and assess its practicality and effectiveness based on the Sharpe ratio and maximum drawdown metrics.

1. Project Goal: The project aims to implement and analyze a statistical arbitrage strategy over a dataset of cryptocurrency prices. Pairs trading, a subset of stat-arb strategies, was applied across 40 major cryptocurrencies, with the strategy's performance assessed via back-testing over historical price data.

2. Procedure:

As the goal of the project was to replicate the approach developed by Avellaneda and Lee (2010), the following steps were laid out to be implemented by the code:

- **Factor Return Computation:** The empirical correlation matrix Σt was computed using normalized return series within a 240-hour window. Principal Component Analysis (PCA) was performed on Σt to identify the top two principal component vectors, which were then used to construct the risk factors.
- **Residual Return and s-score Estimation:** Linear regression was applied to the hourly returns of each token to estimate regression coefficients and residual series. These were used to compute the s-score for each token using the specified formulae, allowing for the evaluation of mean reversion characteristics.
- **Trading Signal Generation:** Using estimated parameters, trading signals were generated for each cryptocurrency. The methodology followed the decision rules provided by Avellaneda and Lee (2010) to determine buy or sell positions.
- **Strategy Performance Evaluation:** A back-testing framework was set up to trade tokens according to the generated signals, and the portfolio returns were recorded. Performance metrics such as the Sharpe ratio and maximum drawdown were computed to evaluate the strategy's effectiveness.

3. Project Tasks:

- **Eigenportfolio Weight Computation:** The weights corresponding to the largest two eigenvalues were computed and saved in CSV files. Cumulative return curves for the two eigenportfolios and two individual cryptocurrencies (BTC and ETH) were plotted for visual comparison.

- **Eigenportfolio Weight Visualization:** The weights of the two eigenportfolios at specific timestamps were plotted and sorted to visualize the weight distribution across different tokens.
- **s-score Evolution Plotting:** The evolution of the s-score for BTC and ETH was plotted over a one-month period to observe the mean reversion signals.
- **Trading Signal Recording and Strategy Evaluation:** A CSV file containing hourly trading signals for all 40 tokens was created, and the cumulative return curve of the strategy was plotted. The histogram of hourly returns was

5. Implementation & program structure overview:

a. Classes:

- **Functions:** We first define a library Functions class that will contain the following components and provides critical utility functions and supports the analysis.
- It encapsulates a series of methods dedicated to data analysis and visualization pertinent to financial time series and regression analysis, which are crucial in understanding market behaviors and validating trading strategies.
- Methodologies and Approaches
- Time Series Visualization (time_plot): The time_plot method offers a dynamic approach to visualizing financial time series data. By melting the input DataFrame, it enables the plotting of multiple variables over time in a single figure. The inclusion of summary statistics—mean and standard deviation—directly on the plot provides immediate insight into the data's behavior, aiding in quick decision-making.
- Eigen Portfolio Weights Analysis (eigenweights): This method focuses on visualizing the weights of eigen portfolios at specific points in time. By transposing the data and sorting the eigenweights, it gives a clear picture of the token distribution within the portfolio. This is especially important when assessing the concentration and diversification of the investment strategy.
- Distribution Analysis (plot_hist): The ability to plot histograms allows for the examination of the distribution of returns, which is vital in risk management and strategy validation. Customization options such as axis labels and bin sizes make this method versatile for various data explorations.
- Principal Component Analysis (pca): The PCA method is a critical component of risk factor modeling. By standardizing the data and computing covariance matrices, it isolates the principal components that explain the most variance within the dataset, which is essential in constructing eigen portfolios and identifying underlying market factors.
- Regression Analysis (regress_part1 and regress_part2): These methods conduct regression analysis to understand the relationship between token returns and risk factors, and to calculate the residual series. This analysis is a fundamental aspect of pairs trading strategies, where understanding and exploiting these relationships can lead to profitable opportunities.
- Results and Analysis
- The implementation of these methods provides a multi-faceted view of the market:

- Time Series Analysis: The time_plot method's output reveals trends and volatility patterns in the financial time series data. It could potentially uncover periods of market stress or stability, which are crucial for a stat-arb strategy.
- Eigen Portfolio Construction: The eigenweights plots inform the strategy's exposure to market movements. Large weights in certain tokens may indicate a higher dependency on their performance.
- Return Distribution Insights: The histograms generated shed light on the expected return profiles and the presence of outliers, which can affect the risk assessment of the trading strategy.
- Factor Model Efficiency: The PCA method confirms the number of significant factors driving the market, which validates the dimensionality reduction aspect of the strategy.
- Residual Behavior and Strategy Feasibility: Regression analysis evaluates the effectiveness of the strategy in isolating noise from the market factors, which is critical for identifying mean-reversion opportunities.
- **CryptoDataProcessor**:
- The CryptoDataProcessor class serves as the foundation for the extraction and initial processing of cryptocurrency data. The read_and_process_data method is an integral part of the workflow, designed to ingest data from CSV files containing price and ticker information. This method demonstrates a meticulous approach to data preparation by converting timestamps to a consistent datetime format and merging price data with ticker information. A significant aspect of this procedure ensures the inclusion of prominent cryptocurrencies such as Bitcoin (BTC) and Ethereum (ETH) in the analysis, which are pivotal in the cryptocurrency market.
- Eigenvector and Weight Calculation
- The eigen method exemplifies a critical analytical step in the statistical arbitrage process. It applies principal component analysis (PCA) to the dataset to determine the primary risk factors influencing the portfolio. This method calculates standard deviations, eigenvectors, and eigenvalues, which are then used to construct eigen portfolios. The calculated eigenvectors are adjusted by the standard deviation to determine the weights (w_eig and w_eig2) of each token in the eigen portfolios. Subsequently, the method computes the first and second risk factors, represented by Factor1 and Factor2. These risk factors are essential for understanding the systematic risk in the cryptocurrency market.
- Coefficient Calculation
- The coef method is a testament to the thoroughness of the class's analytical capabilities. It iteratively calculates coefficients for a list of tickers, representing the sensitivity of each cryptocurrency to the identified risk factors. This method takes a rigorous approach by accounting for each token's residual returns and cumulative sums. It then applies regression analysis to these values to derive coefficients that capture the mean-reversion tendency and volatility of each token. This information is vital for identifying trading opportunities and assessing the risk profile of individual tokens within the arbitrage strategy.
- Analytical Rigor
- Throughout its procedures, the class displays a consistent emphasis on data integrity and error handling, replacing any non-finite values with zeros to maintain the robustness of subsequent analyses. The use of lambda functions and DataFrame operations showcases the class's capability to handle complex data manipulations efficiently.

- Integration with Strategy Implementation
- While the class is focused on data processing, its outputs are directly integrated into the broader strategy implementation. The calculated eigenvectors, weights, and coefficients are essential inputs for the trading signal generation and performance evaluation components of the strategy. The class functions as a vital intermediary, transforming raw data into actionable insights and inputs for further stages of the stat-arb strategy implementation.
- The **MeanReversionStrategy** class encapsulates the logic for a mean reversion trading strategy, tailored specifically for cryptocurrency markets. This strategy is pivotal in identifying and capitalizing on temporary inefficiencies in asset prices.
- Key Features of the Strategy
- **Threshold-Based Signal Generation:**
- The strategy uses predefined threshold levels to generate trading signals.
- Buy and sell signals are determined based on these threshold levels, indicating potential mean reversion opportunities.
- **Dynamic Signal Updates:**
- The class iterates through a DataFrame (**df_coeff**) containing coefficients for each ticker.
- For each ticker, it checks the current signal and updates it based on the s-score (**S**) and the defined threshold levels.
- **Threshold Levels:**
- **Sell Open Threshold (1.25):** A signal to open sell positions when the s-score exceeds this level.
- **Buy Open Threshold (-1.25):** A signal to open buy positions when the s-score drops below this level.
- **Sell Close Threshold (0.75):** A signal to close sell positions when the s-score falls below this level.
- **Buy Close Threshold (-0.5):** A signal to close buy positions when the s-score rises above this level.
- Signal Generation Logic
- **Zero Signal State:**
- If there's no current signal for a ticker, the strategy checks the s-score against the open thresholds.
- A buy signal (1) is generated if the s-score is lower than the buy open threshold.
- A sell signal (-1) is generated if the s-score is higher than the sell open threshold.
- **Existing Buy Signal:**
- For tickers with an existing buy signal, the strategy checks if the s-score has risen above the buy close threshold.
- If so, it closes the buy position by resetting the signal to zero.
- **Existing Sell Signal:**
- For tickers with an existing sell signal, the strategy checks if the s-score has dropped below the sell close threshold.
- If so, it closes the sell position by resetting the signal to zero.
- Application in Trading
- The strategy is designed to exploit small, temporary deviations from an asset's intrinsic value, a common occurrence in the volatile cryptocurrency market.

- By using a systematic, threshold-based approach, the strategy aims to automate the decision-making process in trading, reducing emotional biases and improving reaction times to market changes.
- Integration with Data Analysis
- The mean reversion signals generated by this class are dependent on the output of previous data processing stages, particularly the coefficients calculated for each ticker.
- This integration highlights the interconnected nature of the various components within the statistical arbitrage strategy, underscoring the importance of accurate data analysis in informing trading decisions.

6. Analysis and Discussion:

- **Performance Metrics:** The Sharpe ratio, a measure of risk-adjusted return, was calculated using a 0% benchmark rate, providing an indication of the excess return per unit of risk taken. The maximum drawdown metric offered insight into the potential losses from peak to trough, suggesting the strategy's risk during adverse market movements.

- **Strategy Practicality:** The strategy's practicality was considered in the context of the cryptocurrency market's volatility and liquidity. The analysis accounted for transaction costs, market impact, and the capacity of the strategy to adapt to market changes.

- **Conclusions:** The strategy's effectiveness was measured by its ability to generate positive excess returns while minimizing drawdowns. The results indicated that while the strategy had periods of profitability, it also exhibited significant risks, as evidenced by the maximum drawdowns.

- **High Frequency = High Cost:** As noted by Avellaneda and Lee themselves, one of the primary downsides to the type of strategy employed is the high number of trades incurring ever-increasing transaction cost, further diminishing profits and already slim margins. This strategy was implemented in a costless hypothetical, but that means that the slim alpha generated would be diminished even further, probably to overall losses.

- **Expected (average) negative return:** despite overall cumulative return in the positives, the expected overall return (and Sharpe Ratio) was negative, indicating that a strategy built in this manner would likely not be considered “investment-grade” and more of a high-risk strategy.

- **This is further supported by -**4.77 MDD: sharp downturns in the overall portfolio indicate that the returns of this strategy are not stable, although, in the context of cryptocurrency (and the few years that it has had) makes this

- **Importance of filtration criteria:** it became readily apparent that one of the key constraints of the strategy, especially with regards to the cryptocurrency market, was constraining the sample data to the largest market cap coins. This is likely necessary as the depth of market beyond those coins is poor enough (and collapses have been frequent) for the strategy to be significantly worse if allowed to access all coins.

- **Returns clustering around 0:** the returns of the strategy were normally distributed around 0, a very pleasing result with regards to the efficient market hypothesis, if not for generating alpha.

- **Dimensionality Reduction:** PCA reduces the number of variables while retaining the essential information, making it easier to analyze and visualize complex data sets, like those found in cryptocurrency markets.
- **Simpler Graphical Representation:** PCA facilitates the visualization of complex, multi-dimensional cryptocurrency data in two or three dimensions, making it easier to detect patterns and outliers.This allows for a clearer understanding of how different coins are correlated with each other.
- **Efficient Computational Processing:** PCA can significantly reduce computational complexity, which is beneficial given the vast amount of data in the cryptocurrency market. Faster processing makes it more feasible to perform real-time or near-real-time analysis, a critical factor in the fast-paced crypto trading environment.
- **Reduces Overfitting:** Identifies Unique Coin Characteristics: PCA can help segment the market into different niches or categories based on underlying characteristics, aiding in targeted investment strategies.
- **Standardizes Comparison**: By reducing dimensions, PCA standardizes the comparison among various cryptocurrencies, making it easier to evaluate relative performance. PCA helps in reducing the risk of overfitting in models by eliminating redundant features.

7. Project Deliverables:

- **Codebase:** The implementation was carried out in Python, leveraging its rich ecosystem of libraries for data analysis and scientific computing. Object-oriented programming principles were applied to structure the code effectively, with key classes encapsulating the functionality of data processing, factor analysis, signal generation, and performance evaluation.
- **CSV Outputs:** Three CSV files were produced, containing the eigenportfolio weights, trading signals, and other relevant metrics calculated during the testing period.
- **Project Report:** This document serves as the project report, detailing the problem addressed, the methodologies applied, the outcomes of the implementation, and the conclusions drawn regarding the strategy's viability.