Final Project: Implementing a Statistical Arbitrage Strategy
Due: December 12, 2024

1 Project Goal

Statistical arbitrage (a.k.a stat-arb) strategies are commonly employed in quantitative asset management practices. Pairs trading is a well-known example of stat-arb strategies. The goal of this project is to implement a stat-arb strategy as proposed in [**Avellaneda and Lee 2010**] and examine the strategy performance over a universe of 40 crypto currencies (or, tokens). The hourly price data of more than 120 tokens are taken from FTX over the time period of 2021-2-19 to 2022-9-26 and saved in file "coins_all_prices.csv". Among these tokens, the 40 with largest market capitalizations in each hour based on their respective market prices are recorded in file "coins_universe_150K_40.csv".

The project is due on December 15, 2023. You may choose to complete the project using either Python or C++.

2 Implementation Procedure

Description of the implementation process:

1. **Compute factor returns of the two risk factors at time** t. Pick a starting time t, say, t =2021-03-08T05 :00 :00+00 :00. Let Ω_t denote the set of 40 largest market-cap tokens in hour t. $\Omega_t = \{S_t^{c(1)}, S_t^{c(2))}, \cdots, S_t^{c(i)}, \cdots, S_t^{c(40)}\}$ where $S_t^{c(i)}$ denotes the hour-t price of token index $c(i)$.

   - Choose a time-window size M, say $M = 240$. Use the N hourly price series $\{(S_k^{c*(1)}, S_k^{c*(2))}, \cdots, S_k^{c*(N)}) : k = t-M, t-M+1, \cdots, t-2, t-1\}$ to compute the empirical correlation matrix Σ_t of the N normalized return series $\{(Y_k^{c*(1)}, Y_k^{c*(2))}, \cdots, Y_k^{c*(i)}, \cdots, Y_k^{c*(N)}) : k = t-M, t-M+1, \cdots, t-2, t-1\}$, where N is the largest number of tokens having at least 80% valid price data in the time-window of size M. For instance, if there are no change in the 40 largest market-cap tokens within this time-window, then $N = 40$; if there are only 34 common tokens within this time window, then $N = 34$. The definition of normalized return $Y_k^{c*(i)}$ for token $S^{c*(i)}$ in hour k is given in section 2.1 (page 764) of [**Avellaneda and Lee 2010**] with hourly return $R_k^{c*(i)} = \frac{S_k^{c*(i)} - S_{k-1}^{c*(i)}}{S_{k-1}^{c*(i)}}$.

     Note: if a token has less than 20% of the prices in a time-window of size M, it shall not be included in the PCA analysis in current time-window. For sporadically missing price data, use forward-fill to fill it in.

   - Apply any Principal Component Analysis (PCA) package/library to Σ_t and get the top **two principal componet vectors** $\{v^{(1)}, v^{(2)}\}$ along with their corresponding eigenvalues $\{\lambda_1, \lambda_2\}$. The two risk factors are constructed by the respective eigenportfolios $Q^{(1)}$ and $Q^{(2)}$ where $Q^{(j)} = (\frac{v_{c(1)}^{(j)}}{\bar{\sigma}_{c(1)}}, \frac{v_{c(2)}^{(j)}}{\bar{\sigma}_{c(2)}}, \cdots, \frac{v_{c(40)}^{(j)}}{\bar{\sigma}_{c(40)}})$ represents the portfolio weight vector of eigenportfolio j ($j = 1, 2$).
   - Calculate factor return of risk factor j ($j = 1, 2$) in hour k ($k = t-M, t-M+1, \cdots, t-2, t-1, t$), denoted as $F_{j,k}$, using (the same as equation (9) in section 2.1 (page 764) of [**Avellaneda and Lee 2010**]):

$$F_{j,k} = \sum_{i=1}^{40} Q_{c(i)}^{(j)} \cdot R_k^{c(i)} = \sum_{i=1}^{40} \frac{v_{c(i)}^{(j)}}{\bar{\sigma}_{c(i)}} \cdot R_k^{c(i)}.$$

2. **Estimate residual returns of token** $S^{c(i)}$ $(i = 1, 2, \cdots, 40)$ **and its s-score at** t.

- For each token $S^{c(i)}$ $(i = 1, 2, \cdots, 40)$, apply linear regression to hourly returns $\{R_k^{c(i)}, k = t-M, t-M+1, \cdots, t-2, t-1\}$ to get regression coefficients $(\beta_{i,0}(t), \beta_{i,1}(t), \beta_{i,2}(t))$ and the residual series $\{\epsilon_k^{c(i)}\}$.

$$R_k^{c(i)} = \beta_{i,0}(t) + \beta_{i,1}(t)F_{1,k} + \beta_{i,2}(t)F_{2,k} + \epsilon_k^{c(i)}$$

- Let $X_l = \sum_{n=t-M}^{t-M+l} \epsilon_n$ $(l = 0, 1, \cdots, M-1)$. Apply linear regression to $X_{l+1} = a + bX_l + \eta_{l+1}^{c(i)}$ to get a and b. Then, using a, b, and changing the value of Δt to $1/8760$ in equation (A1) in Appendix A of [**Avellaneda and Lee 2010**] (page 781) to get $\kappa_{c(i)}$, $m_{c(i)}$, $\sigma_{c(i)}$ and $\sigma_{eq}^{c(i)}$ at time t.

3. **Generate trading signals at time** t.

   - Pick a set of trading parameter values $(\bar{s}_{bo}, \bar{s}_{so}, \bar{s}_{bc}, \bar{s}_{sc})$. For instance, $\bar{s}_{bo} = \bar{s}_{so} = 1.25$, $\bar{s}_{bc} = 0.75$, and $\bar{s}_{sc} = 0.5$.
   - For each token $S^{c(i)}$ $(i = 1, 2, \cdots, 40)$, make use of the estimated $\kappa_{c(i)}$, $m_{c(i)}$, $\sigma_{c(i)}$ and $\sigma_{eq}^{c(i)}$ to get the s-score $s^{c(i)}$ of token $S^{c(i)}$ using equation (15) in [**Avellaneda and Lee 2010**] (page 769). Compare the value of $s^{c(i)}$ against $(\bar{s}_{bo}, \bar{s}_{so}, \bar{s}_{bc}, \bar{s}_{sc})$ to generate buy/sell signals according to rules specified in equation (16) of [**Avellaneda and Lee 2010**] (page 769) for $S^{c(i)}$ at time t.

4. **Evaluate strategy performance over a testing period** $[t_{start}, t_{end}]$

   - For time t looping from a starting point t_{start} to ending point t_{end}, trade tokens at time t according to signals generated. Each transaction quantity is 1 share of a token. Record the portfolio return in each period based on the holding positions in all tokens and their respective returns in the period.
   - Compute the Sharpe ratio and maximum draw-down (MDD) (See this link for explanation on computing MDD with Python https://quant.stackexchange.com/questions/18094/how-can-i-calculate-the-maximum-drawdown-mdd-in-python).

3 Project Tasks

Consider a testing period from 2021-09-26 00:00:00 to 2022-09-25 23:00:00.

1. Compute the eigenportfolio weights corresponding to the largest two eigenvalues of the empirical correlation matrix of the top 40 tokens in each hour over the testing period. Save the eigen-vectors corresponding to the largest and second-largest eigenvalues into two separate csv files with rows indexed by time-stamp and columns corresponding to tokens. Plot the cumulative return curves of 4 assets over the testing period in ONE figure: the first eigen-portfolio, the second eigenportfolio, BTC, and ETH.

2. Plot the eigen-portfolio weights of the two eigen-portfolios at 2021-09-26T12:00:00+00:00 and then at 2022-04-15T20:00:00+00:00 (in the same way as Figure 4 and Figure 5 on pages 766-768 in [**Avellaneda and Lee 2010**]). Sort the portfolio weights from largest to smallest.

3. Plot the evolution of s-score of BTC and ETH from 2021-09-26 00:00:00 to 2021-10-25 23:00:00 in two different figures.

4. Create a csv file containing the trading signals of all 40 tokens in each hour over the entire testing period with rows indexed by time-stamp and column headers being token names. Plot the cumulative return curve of the implemented strategy and the histogram of the hourly returns (namely, two figures). Report the Sharpe ratio (using 0% as the benchmark rate) and the maximum drawdown over the testing period.

4 Project deliverables and requirements:

1. Ready-to-run codes and the three output csv files in one zip file.
2. The codes shall demonstrate the use of object-oriented design (namely, the effective use of classes) in implementation.
3. A project report which contains the following components:
   - Problem addressed by the project and technical model(s) applied.
   - Explanation of the functionalities of key classes/functions and their inputs/outputs.
   - Outcome of the implementation including required figures in "Project Tasks", performance evaluation metrics such as Sharpe value and Max Drawdown (as computed in Project Tasks), and present discussions on whether the trading strategy is good and/or practical. Draw conclusions.

References

[1] Avellaneda and Jeong-Hyun Lee (2010). "Statistical arbitrage in the US equities market", *Quantitative Finance*, vol. 10, No. 7, AugustSeptember 2010, 76178.