Master Level 2

24_M2_NI_FENG_S1_CCO_QUANT_3682

EMPIRICAL METHODS IN FINANCE

ASSIGNMENT

START DATE: 11/10/2023 4PM (Paris time)

DUE DATE: 23/10/2023 6PM (Paris time)

Professor/Lecturer: Vincent MILHAU

INSTRUCTIONS:

- **APPENDIX(CES) PROVIDED :**
- **SafeAssign : THIS ASSIGNMENT IS PROTECTED AGAINST PLAGIARISM**

- **Assignment Type : Individual assignment**
- **Assignment File(s) Type and Number : Python notebook (IPYNB file) to submit**
- **Submission Attempts : 1 attempt**
- **Instructions for students : see next page.**

Page **1** sur **1**

Empirical Methods in Finance
Assignment

Instructions

This assignment consists of 3 independent problems to be solved with Python. You are expected to write your code in a single Jupyter notebook and to add text in Markdown cells to describe each piece of code and answer the questions that do not require you to write code.

Every student must submit a **personal** file. If two files are identical or identical up to cosmetic changes (e.g., renaming a variable while otherwise keeping the exact same code), they will be regarded as evidence of academic misconduct, and the case will be handled according to the School's policy.

Your work must be submitted exclusively **via BlackBoard**. Late submissions and submissions by any other method than BlackBoard will not be accepted. The submitted files must include the Jupyter notebook with your Python code (a file with the .ipynb extension) and any other file that is required for your code to run (e.g., a Pickle file or a comma-separated value file). You are not allowed to use other Python packages than those used in this course, i.e., `numpy`, `scipy`, `pandas`, `matplotlib`, `scikit-learn`, `statsmodels`, `arch`, `cvxopt` and `yfinance`.

Evaluation criteria:

- Correcteness of code;
- Explanations around code: every cell of code should be introduced by at least one sentence explaining what the code does. For a function, write a more detailed explanation of the algorithm, but keep it concise. Divide your notebook into sections to separate exercises by using appropriate Markdown formatting (see Markdown documentation here). Add axis labels to plots to make them self-explanatory;
- Clarity of code: use descriptive but reasonably short variable names, keep comments within code to a minimum, keep code concise (but this is not a code golf competition, code should stay clear).

1. Maximum Time Under Water

The following definition of the maximum time under water is reproduced from the Glossary of Scientific Beta website:

> "Using daily return series, a daily price index is obtained for the strategy portfolio. We define 'High Water Marks' (or HWM) as the points in the price index at which the index

value is greater than any previous index value and is followed by a loss. For each HWM, which is also the starting point of a drawdown, we calculate 'Time Under Water' (or TUW) as the number of business days required to reach the same index value at the HWM or, in other words, the time required to reach the point at which the cumulative losses for that drawdown are equal to zero. If the HWM value is never reached in the future, which could be case with the latest HWM, we define TUW as the number of business days from HWM till the end of the data series. The maximum across all TUW values is reported as the 'Maximum Time Under Water'. It should be noted that this 'Maximum Time Under Water' corresponds to the drawdown that lasted for the longest period of time and not to the maximum drawdown."

We are calculating the maximum time under water for the 6 funds analyzed in class (data in the file funds.csv) by following this definition, except that we are counting calendar days rather than business days. Formally, we define a high water mark (HWM) as follows: if T is the number of quotes in the sample and P_t is the value at time t, P_t is a HWM if $t < T$ and

$$P_t \geq P_s \quad \text{for all date } s \leq t \quad \text{and} \quad P_{t+1} < P_t.$$

1. Write a Python function that calculates the first hitting index for a Pandas Series. This function should take a Pandas Series and a number and return the first index where the Series is greater than or equal to the number. If the target is never hit, the function should return some default value, e.g. an empty object.
2. Write a Python function that returns the longest "period under water" for a Pandas Series and the duration in days of that period. A possible approach is to calculate all HWMs in the Series, as defined previously, and to find the first "recovery date" after each HWM. The start date of the "period under water" is the date when the HWM is first reached, and the end date is the date when it is reached for the second time. The function should return the start and the end dates of the longest period under water and the duration in calendar days.
3. Find the longest periods under water for the 6 funds.

2. Factor-Based Covariance Matrix Estimation

Many portfolio optimization programs require an estimate for the covariance matrix of the constituents, e.g. the variance minimization program:

$$\min_{\mathbf{w}} \sigma_p^2 \quad \text{subject to } \mathbf{w}'\mathbf{1} = 1.$$

In this program, $\sigma_p^2 = \mathbf{w}'\mathbf{\Sigma}\mathbf{w}$ is the portfolio variance, $\mathbf{w}$ is the column vector of percentage weights, $\mathbf{w}'$ is the transposed vector of weights, $\mathbf{1}$ is a column vector of 1 with the same length as $\mathbf{w}$, and $\mathbf{\Sigma}$ is the covariance matrix of constituents. The portfolio that solves this program is called the *global minimum variance* (GMV) portfolio.

The covariance matrix is often estimated as the sample covariance matrix of past returns, but this estimator treats all covariances as independent parameters. When the number of constituents (N) is large relative to the sample size (T), this gives rise to the "curse of dimensionality problem": we have many independent covariances to estimate from too few datapoints. One way to mitigate this problem is to use a *structured* covariance matrix estimator, e.g. the one implied by the 3-factor model of Fama and French. To construct this estimator, the returns on the N constituents in the investment universe are regressed against the market, the size (SMB) and the value (HML) factors, as per the following equation:

$$r_{t,t+1,i} = \alpha_i + \beta_{\mathrm{MKT},i}\mathrm{MKT}_{t+1} + \beta_{\mathrm{SMB},i}\mathrm{SMB}_{t+1} + \beta_{\mathrm{HML},i}\mathrm{HML}_{t+1} + \varepsilon_{t+1,i}.$$

The estimated betas are stacked in a $3 \times N$ matrix $\hat{\boldsymbol{\beta}}$, and the covariance matrix of the 3 factors is estimated as the sample covariance matrix of factor values, $\hat{\boldsymbol{\Sigma}}_F$. For each portfolio, the variance of the residuals $\boldsymbol{\varepsilon}_{.,i}$ is estimated as

$$\hat{\sigma}^2_{\varepsilon,i} = \frac{1}{T-K-1}\sum_{t=0}^{T-1}\hat{\varepsilon}^2_{t+1,i},$$

where $\hat{\varepsilon}_{t+1,i}$ is the estimated residual at time $t+1$, T is the number of returns, and $K = 3$ is the number of regressors excluding the intercept. Then, the diagonal covariance matrix of estimated residual variances is constructed:

$$\hat{\boldsymbol{\Sigma}}_\varepsilon = \begin{bmatrix} \hat{\sigma}^2_{\varepsilon,1} & 0 & 0 & \cdots & 0 \\ 0 & \hat{\sigma}^2_{\varepsilon,2} & 0 & \cdots & 0 \\ \cdots & & & & \\ 0 & 0 & 0 & \cdots & \hat{\sigma}^2_{\varepsilon,N} \end{bmatrix}.$$

Finally, the *factor-based covariance matrix* is calculated as

$$\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\beta}}'\hat{\boldsymbol{\Sigma}}_F\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\Sigma}}_\varepsilon.$$

In this problem, you are writing a function that calculates the factor-based covariance matrix, and you are calculating the GMV portfolio with two covariance matrix estimates, namely the sample matrix and the factor-based one. Our reference investment universe consists of the 100 portfolios formed on size and book-to-market ratio from Ken French's data library.

1. Import the weekly factor values (market, size and value) and the daily portfolio returns.
2. Eliminate from the investment universe the portfolio that have at least one missing return (represented as -99.99 in the data) between January 2, 1970 and July 28, 2023. How many portfolios does the universe contain? The analysis in what follows is restricted to the period from January 9, 1970 to July 28, 2023.
3. Calculate the daily portfolio values assuming an initial value of $1 on January 2, 1970.
4. From the daily portfolio values, calculate the weekly portfolio returns over the same weeks as in the factor dataset.
5. Write a function that takes a DataFrame of asset returns (T rows and N columns) and a DataFrame of factor values (T rows and K columns) and returns the factor-based covariance matrix estimate.
6. Calculate the GMV portfolio with the following two covariance matrix estimates: the sample covariance matrix estimated from January 1, 2021 to January 1, 2023, and the factor-based matrix estimated over the same period. Which portfolio has the bigger range of weights?

7. Compare the leverage amounts of the two portfolios. The leverage of a portfolio is defined as

$$\frac{1}{2}\left[\sum_{i=1}^{N} |w_i| - 1\right],$$

where $|w_i|$ denotes the absolute value of weight w_i.

8. Simulate the weekly values of a minimum variance strategy based on the sample covariance matrix and rebalanced every 12 weeks. At each rebalance date, the portfolio weights are the minimum variance weights calculated from the sample covariance matrix estimated over the past 3 years of weekly data.

9. Repeat the previous question by using the factor-based covariance matrix estimate.

10. Compare the annualized volatilities of the previous two portfolios and conclude on the benefits of using the factor-based matrix over the sample matrix.

3. Asian Option Pricing

An Asian call is a call option on the average price of an asset over a given period. Suppose the option is written at time 0 and expires at time t, the strike price is K, and the value of the underlying asset (here, a stock index) at time t is S_t. Let $t_1, \ldots, t_n$ be the monitoring dates, all of which are less than or equal to t. The payoff of the call option is:

$$C_t = \left[\frac{1}{n}\sum_{i=1}^{n} S_{t_i} - K\right]^+, \tag{1}$$

where the notation x^+ stands for $\max(x, 0)$. The option price is:

$$C_0 = e^{-rt}\mathbb{E}[C_t], \tag{2}$$

where the expectation $\mathbb{E}$ is taken under the *equivalent martingale measure*, and r is the risk-free rate. Under the equivalent martingale measure, the underlying asset price evolves as

$$\frac{dS_t}{S_t} = rdt + \sigma dz_t,$$

where z is a Brownian motion.

We use the following parameter values:

$$t = 6 \text{ months in questions 1 to 3}, \quad r = 2\%, \quad , \sigma = 18\%, \quad S_0 = \$100, \quad K = S_0.$$

The objective of this exercise is to calculate the option price by a Monte-Carlo method for various monitoring frequencies and maturities.

1. Simulate 10,000 paths for the underlying asset price under the equivalent martingale measure over 6 months with a daily time step, assuming that there are 22 business days in a month, hence $22 \times 12 = 264$ business days in a year. (You can reuse code from the class examples.)

2. Simulate 10,000 values for the option payoff (1) using daily monitoring dates (that is, all the simulation dates). Then, estimate the option price using (2).

3. Repeat the previous question using weekly monitoring dates (that is, one simulation date out of five).

We now return to daily monitoring and want to plot the option price as a function of the maturity when the maturity ranges from 1 day to 6 months, or 132 business days, with a daily time step.

4. Write a Python function that takes the array of underlying asset values obtained in Question **1** and returns the estimated option price for every maturity and the 95% confidence bounds for the price.

5. Plot the option price and the bounds as a function of the maturity.