Mandatory Assignment 1

2024-03-01

General remarks

- You are supposed to hand in one (!) .qmd (Quarto) script with commented source code in Python or R and one (!) .pdf report in which you describe your methods and results (e.g., with figures, tables, equations, etc.)
- The report (font size 12pt, use the provided template) can have maximal 7 pages including figures, tables, and other appendices
- The source code has to create every figure, table, and number used for your report
- When reporting numbers in your report, please consider a meaningful rounding scheme (there is usually no need to report six digits after the comma). Make sure you only use `round()` for the final reported results and not for any intermediate steps
- Within the report, describe every significant step and make sure that tables and figures are self-explanatory by providing meaningful captions and variable names. You cannot assume that the reader of the report goes through your code in case your description may be confusing
- You increase your chance of helpful feedback as well as your final grade if you provide meaningful comments in the code that explain the purpose of each step
- **Minimum requirement**: the code must run without error or interruption. Ensure the code loads all required packages at the beginning of the script (you can assume that the user has the packages available locally, so do not include `install.packages()` commands). Attempt to solve every task. Document your struggles if you do not finish an exercise. **Hand-ins that do not fulfill this minimum requirement will not receive peer feedback and will not be considered for the final exam**
- The assignments can be written individually or by groups of a maximum of three students
- The plagiarism rules must be complied and please be aware of the rules for co-written assignments
- All parts must be answered in English, including comments in the code.

The deadline to hand in the file on Peergrade via Absalon is **Friday, March 8th, 7 pm**.

Efficient portfolios and estimation uncertainty (inspired by Section 3 of "Portfolio Choice Problems," Brandt [2010])

How severe is the statistical error for portfolio weights? One way to answer this question is to measure the economic loss using plug-in estimates instead of the theoretical, truly optimal portfolio weights. Jobson and Korkie [1980] were among the first to document the finite-sample properties of plug-in estimates. In this assignment, you will replicate their main finding and highlight the shortcomings in real-life applications.

To define terms, we consider the returns of N risky assets, which exhibit a variance-covariance matrix Σ and a mean vector μ. Plug-in estimates are based on using the sample counterparts $\hat{\Sigma}$ and $\hat{\mu}$ to compute portfolio weights for the mean-variance allocation problem with desired expected return $\bar{\mu}$:

$$\hat{w}(\bar{\mu}) = \arg\min w'\hat{\Sigma}w \text{ s.t. } \omega'\iota = 1 \text{ and } w'\hat{\mu} \geq \bar{\mu}.$$

In the mean–variance problem above, the investor chooses the portfolio weights $\hat{w}(\bar{\mu})$ based on estimated parameters $\hat{\mu}$ and $\hat{\Sigma}$ instead of the (unknown) true parameters μ and Σ. As a result, $\hat{w}(\bar{\mu})$ may deviate from $w(\bar{\mu})$, the theoretically optimal portfolio which delivers the lowest possible portfolio volatility for a given level of desired returns $\bar{\mu}$.

To address the question of how reliable plug-in estimates of mean–variance efficient portfolio weights are for a given sample size, we consider these assets' historical sample moments to be the truth. Then, we simulate returns from a given data-generating process to obtain estimates $\hat{\mu}$ and $\hat{\Sigma}$ and the corresponding portfolio weights $\hat{w}(\bar{\mu})$. That way, we can compare the performance of the sampled portfolios against the performance of the true optimal weights.

Exercise

1. Follow the exercise sets to download daily adjusted prices for all constituents of the Dow Jones 30 index for the period from January 1st, 2000 until December 31st, 2023 from Yahoo!Finance. Remove all tickers with no continuous trading history for the entire sample (hint: you should end up with $N = 27$ assets). Compute monthly returns for each of the tickers.
2. Compute the sample mean μ and the variance-covariance matrix Σ of the monthly returns.[1] Which of the N individual assets delivered the highest Sharpe ratio (assume the risk-free rate is zero) during the sample period?[2]
3. Define a function `compute_efficient_frontier` which takes at least two inputs: a $N \times N$ variance-covariance matrix `Sigma_est` and a vector `mu_est`.[3] The function `compute_efficient_frontier` should perform each of the following steps:

- Compute the minimum variance portfolio weight ω_{mvp} for the input `Sigma_est`. The function should be able to handle positive definite variance-covariance matrices of arbitrary dimensions $N \times N$

[1] I do not use the notation $\hat{\mu}$ or $\hat{\Sigma}$ here, because in what follows we assume that these estimated parameters are the true ones.

[2] Whenever reporting performance measures, compute annualized values and clearly state how you calculated the measure.

[3] The function is allowed to require more inputs if needed.

- Compute the efficient portfolio weights $\omega_{\text{eff}}(\bar{\mu})$ for the inputs `Sigma_est` and `mu_est` that delivers two times the expected return of the minimum variance portfolio weight
- Make use of the two-mutual fund theorem to characterize a range of portfolio weights on the efficient frontier: Specifically, compute the weights of a sequence of portfolios which are combinations of the minimum variance portfolio weight and the efficient portfolio, $\omega_c = c\omega_{\text{mvp}} + (1-c)\omega_{\text{eff}}(\bar{\mu})$ where $c \in \{-0.1, \dots, 1.2\}$
- The function should return a tibble (in R) or a data.frame (in Python) with a column c and N additional columns that contain the corresponding portfolio weight ω_c

3. Use the output of the function `compute_efficient_frontier(Sigma_est, mu_est, ...)` to visualize the theoretically optimal efficient frontier in a diagram with volatility on the x-axis and expected returns on the y-axis based on the true parameters Σ and μ

4. What are the efficient tangency portfolio weights ω_{tgc} under the assumption that the risk-free rate is zero based on the true parameters μ and Σ? Do the tangency portfolio weights seem like a well-balanced portfolio? What are the potential issues when implementing this portfolio in reality? What is the maximum attainable Sharpe ratio assuming the risk-free rate is zero? Should the Sharpe ratio of ω_{tgc} be higher or lower than the Sharpe ratio of the individual assets?

Next, we want to simulate efficient frontiers which suffer from estimation uncertainty. More specifically, we assume that returns are identically and independently multivariate normal distributed with a vector of expected return μ and variance-covariance matrix Σ. We simulate hypothetical return samples from this known data-generating process and compute the sample moments $\hat{\mu}$ and $\hat{\Sigma}$. Finally, we analyze how much the estimated efficient frontier deviates from the true efficient frontier.

5. You are given the following function:

```r
# R
simulate_returns <- function(periods = 200,
                             expected_returns = mu,
                             covariance_matrix = Sigma){
  MASS::mvrnorm(n = periods, expected_returns, covariance_matrix)
}
```

```python
# Python
import numpy as np

def simulate_returns(periods=200,
                     expected_returns=mu,
                     covariance_matrix=Sigma):
    """
        periods (int): Number of periods
        expected_returns (array-like): Expected returns for each asset
        covariance_matrix (array-like): Covariance matrix of returns
    """
```

```
    returns = np.random.multivariate_normal(expected_returns,
                                            covariance_matrix,
                                            size=periods)
    return returns
```

Explain briefly what the function is doing.[4]

6. Use `simulate_returns(periods = 200)` to generate one hypothetical sample of size `periods = 200`. For this simulated sample, compute the sample mean, the sample variance-covariance matrix, and plug-in estimates of the mean–variance frontier again. Evaluate how close these estimates come to the "true" frontier based on exercise 3 by plotting them meaningfully in the figure from above. To be precise: We assume that the investor has the sample estimates available while the actual return distribution is determined by Σ and μ (which are unkown to her). Why do the simulated efficient frontiers deviate from the theoretically optimal one?
7. Repeat the simulation step 100 times and visualize *all* simulated efficient frontiers within one figure together with the theoretically optimal one. What can you conclude from the figure? How does the result depend on the sample size `periods`?

Finally, we analyze how the portfolios perform from an economic perspective.

8. Compute for each of the simulated return samples the efficient tangent portfolio $\widehat{\omega}_{tgy}(\widehat{\Sigma}, \widehat{\mu})$ under the assumption that the risk-free rate is zero. Then, for each of the simulated samples, compute the annualized Sharpe-ratio of the efficient portfolio *evaluated with the true parameters* ($SR = \sqrt{12}\widehat{\omega}'_{tgy}\mu / \sqrt{\widehat{\omega}'_{tgy}\Sigma\widehat{\omega}_{tgy}}$).
9. Visualize these Sharpe ratios in a histogram together with the Sharpe ratio of the efficient tangency portfolio based on the true parameters μ and Σ.
10. How do the results change for larger sample sizes `periods`? Discuss what you can conclude from the figure. Name at least two alternative allocation strategies that could help reduce the plug-in estimates' shortfall. Show if your proposed alternatives improve upon the benchmark we simulated above. Discuss the results.

References

Michael W. Brandt. Portfolio choice problems. In Yacine Ait-Sahalia and Lars Peter Hansen, editors, *Handbook of Financial Econometrics: Tools and Techniques*, volume 1 of *Handbooks in Finance*, pages 269–336. North-Holland, San Diego, 2010. doi: https://doi.org/10.1016/B978-0-444-50897-3.50008-0.

[4] *Note:* Your results should be reproducible. Therefore, it is essential to initialize your random number generator such that your colleagues can verify your results. You should do this in `R` by choosing an arbitrary *seed*, e.g., `2023` (can be any number, but you have to report the seed) and then calling the function `set.seed(2023)` at the beginning of your script. In python, use `np.random.seed(2023)` (after `import numpy as np`). That way, everyone using the same seed can replicate the same sequence of random numbers as you do.

J. D. Jobson and B. Korkie. Estimation for markowitz efficient portfolios. *Journal of the American Statistical Association*, 75(371):544–554, 1980. URL http://www.jstor.org/stable/2287643.