




Problem Set #1

⚙ Status	Done
➤ Course	 <u>BUSN 32520: Advanced Investment</u>
📅 Due date	@January 20, 2024 8:30 AM
📎 Assignment document	PS1_2024.pdf
📁 Type	Problem Set

Problem 1

(a)

- `ret` stores the stock index return.
- `rf` stores the T-bill yield.

(b)

The distribution will be log normal because from the code below:

```
# Simulate data
rng = np.random.default_rng(1000) # Set seed for random num
T = 1000000 # Number of time periods to simulate
e = rng.standard_normal(size = (T,1)) # Random numbers from
rf = np.exp(np.ones((T, 1)) * 0.04 / 12) - 1 # Risk-free ra
ret = np.exp(rf + 0.04 / 12 + e * 0.15 / np.sqrt(12)) - 1 #
data = 'sim'
```

We know that `ret = np.exp(rf + 0.04 / 12 + e * 0.15 / np.sqrt(12)) - 1`

is calculated by exponentialize the normally distributed values, which is e in this case. Therefore, we would see a log-normal distribution for ret.

(c)

The reason for multiplying 12 is for annualization. It is a common practice for working with monthly or daily returns to measure the return in a yearly basis.

(d)

The purpose for multiplying 12/hor is to properly annualize the compounding monthly return.

For this method, we calculate the compounding return by adding 1 to all the return values of index and risk-free rate, and then for each new value, we calculate the compounded product of the preceding 60 values (rolling window of 60 months) by multiplying them together. Once we get all the compound returns with rolling window 60, we subtract 1 to revert to the original form, annualize by 12 (because we are dealing with monthly return) and divide it by 60 (because we calculate each new value by calculate compound returns in the window of 60).

Since using compounding method delivers higher variance compared to simply getting the average of returns and annualize it, and almost the same mean returns, we will end up having a higher weights for the compounding method.

(e)

- w represents the wealth for each candidate portfolio weight.
- U represents the utility of each portfolio.
- $\text{mean}U$ represents the mean utility across all candidate portfolio weights.

Problem 2

With actual data:

Based on 1-month returns, we have:

Mean	Variance	StDev	Optimal Weight
0.081	0.0343	0.1853	0.4721

Based on long-horizon returns, we have:

Mean	Variance	StDev	Optimal Weight
0.102	0.0674	0.2596	0.3026

Numerical evaluation of expected utility from cell #8:

- Optimal weight without approximation, 1-month: 0.46
- Optimal weight without approximation, long horizon: 0.42

For actual return data, 1-month return optimal portfolio share is quite close to the optimal weight without approximation (0.4721 and 0.46), however, for long-horizon, it is not that close (0.3026 and 0.42).

Rerun everything with simulation:

Based on 1-month returns

Mean	Variance	StDev	Optimal Weight
0.052	0.0229	0.1512	0.4547

Based on long-horizon returns

Mean	Variance	StDev	Optimal Weight
0.0721	0.0598	0.2445	0.2413

Numerical evaluation of expected utility from cell #8:

- Optimal weight without approximation, 1-month: 0.46
- Optimal weight without approximation, long horizon: 0.46

For simulated return data, 1-month return optimal portfolio share is quite close to the optimal weight without approximation (0.4547 and 0.46), however, for long-horizon, it is even significantly different (0.2413 and 0.46).

Observations:

- The optimal weight based on 1-month returns is consistently close to the value obtained from numerical evaluation, both with actual and simulated data.
- While for long-horizon returns, there is a notable discrepancy between actual and simulated optimal weights compared to the numerical evaluation.
- The returns in our data may not align with the IID assumption, but it should not be the reason for this discrepancy as investors might only care about mean and variance.
- Long-horizon investor might not need to choose a different portfolio, the reason is not just because they are long-horizon investor, there are several other reasons other than that.
- The approximation error in the optimal portfolio formula is a likely reason for the observed discrepancies in optimal portfolio weights, especially when comparing actual and simulated data for different investment horizons.

Problem 3

```
# Step 1: Read Data
data = pd.read_excel("PS1data.xlsx")
data.set_index("month", inplace=True)

# Step 2: Calculate Portfolio Returns
portfolio_weights = np.array([0.4, 0.6]) # 40% Treasury bills,
portfolio_returns = np.dot(data[['rf', 'rvwind']], portfolio_weights)
data['Portfolio'] = portfolio_returns

# Step 3: Calculate VaR
alpha = 0.05 # 95% confidence level
```

```
var_1month = np.percentile(portfolio_returns, alpha * 100)

# Step 4: Explanation
print(f"One-month 95% VaR of the portfolio: {var_1month:.4f}")
```

The one-month 95% VaR of the portfolio of -0.0452 means with the confidence level of 95%, there is a 5% chance that the portfolio's will experience a loss greater or equal to 4.52% over the course of one month.