Portfolio Optimization

Changlin Yi

cy2578@nyu.edu

Consider the daily closing prices of the following companies from January 1, 2024 until December 31, 2024:

- Microsoft (MSFT)
- JPMorgan (JPM)

A. Determine the average daily return and daily volatility of these two stocks.

In [3]:
```python
import yfinance as yf
import numpy as np
import pandas as pd

tickers = ["MSFT", "JPM"]
data = yf.download(tickers, start="2024-01-01", end="2024-12-31")["Close"]
returns = data.pct_change().dropna()

# Compute metrics
avg_daily_ret = returns.mean()
daily_vol = returns.std()


print("avg daily return", avg_daily_ret)
print("daily_volatility",daily_vol)
```

```
[                       0%                       ]
YF.download() has changed argument auto_adjust default to True
[*********************100%***********************]  2 of 2 completed
avg daily return Ticker
JPM     0.001523
MSFT    0.000653
dtype: float64
daily_volatility Ticker
JPM     0.014845
MSFT    0.012581
dtype: float64
```

B. Annualize these numbers (252 trading days, rf = 0).

In [5]:
```
ann_return = 252 * avg_daily_ret
ann_vol = np.sqrt(252) * daily_vol

print("annual return is", ann_return)
print("annual_vol is",ann_vol)
```

```
annual return is Ticker
JPM     0.383907
MSFT    0.164483
dtype: float64
annual_vol is Ticker
JPM     0.235658
MSFT    0.199714
dtype: float64
```

C. Construct a long-only portfolio that maximizes the Sharpe Ratio (rf = 0).

In [7]:
```
import scipy.optimize as sco

mu = ann_return.values
cov = returns.cov().values * 252

def neg_sharpe(w):
    ret = w.dot(mu)
    vol = np.sqrt(w.dot(cov).dot(w))
    return -ret/vol

cons = ({'type':'eq','fun': lambda w: np.sum(w)-1})
bnds = tuple((0,1) for _ in tickers)
init = np.array([1/2,1/2])

opt = sco.minimize(neg_sharpe, init, bounds=bnds, constraints=cons)
w_opt = opt.x

for ticker, weight in zip(tickers, w_opt):
    print(f"{ticker}: {weight}")
```

```
MSFT: 0.6927027754999318
JPM: 0.30729722450006824
```

D. What are the Risk, Return, and Sharpe Ratio of this portfolio?

In [9]:
```
port_ret = w_opt.dot(mu)
port_vol = np.sqrt(w_opt.dot(cov).dot(w_opt))
port_sharpe = port_ret/port_vol

print("annual return:", port_ret)
print("annual risk:", port_vol)
print("Sharpe Ratio:", port_sharpe)
```

```
annual return: 0.316478414717738
annual risk: 0.18341165007992843
Sharpe Ratio: 1.7255087917251757
```

E. For an investor tolerating up to 18% annual vol, build the max-return portfolio with vol ≤ 18%.

In [11]:
```
def neg_return(w):
    return -w.dot(mu)

cons2 = (
    {'type':'eq','fun': lambda w: np.sum(w)-1},
    {'type':'ineq','fun': lambda w: 0.18 - np.sqrt(w.dot(cov).dot(w))}
)
opt2 = sco.minimize(neg_return, init, bounds=bnds, constraints=cons2)
w_18 = opt2.x

for ticker, weight in zip(tickers, w_18):
    print(f"{ticker}: {weight}")
```

```
MSFT: 0.6647406727014394
JPM: 0.3352593272985605
```

F. What are the Risk, Return, and Sharpe Ratio of this new portfolio?

In [13]:
```
ret2 = w_18.dot(mu)
vol2 = np.sqrt(w_18.dot(cov).dot(w_18))
sr2  = ret2/vol2

print("annual return:", ret2)
print("annual risk:", vol2)
print("Sharpe Ratio:", sr2)
```

```
annual return: 0.310342856515781
annual risk: 0.18000000067954036
Sharpe Ratio: 1.7241269741342617
```

G. How would portfolios in parts (c) and (e) have performed from January 1, 2025 to April 1, 2025?

In [15]:
```
# download out-of-sample
oos = yf.download(tickers, start="2025-01-01", end="2025-04-01")["Close"]
oos_ret = oos.pct_change().dropna()

# compute realized performance
def perf(weights):
    port = oos_ret.dot(weights)
    ann_ret = np.mean(port) * 252
    ann_vol = np.std(port) * np.sqrt(252)
    return ann_ret, ann_vol, ann_ret/ann_vol
```

```
perf_c = perf(w_opt)
perf_e = perf(w_18)
```

```
[*********************100%***********************]  2 of 2 completed
```

In [16]:

```
# Calculate and print the performance of the optimal weights portfolio
perf_c = perf(w_opt)
print("Optimal Weights Portfolio Performance:")
print(f"Annualized Return: {perf_c[0]:.2f}%")
print(f"Annualized Volatility: {perf_c[1]:.2f}%")
print(f"Annualized Sharpe Ratio: {perf_c[2]:.2f}")
print("\n")

# Calculate and print the performance of the volatility-constrained portfoli
perf_e = perf(w_18)
print("Constrained 18%——vola Portfolio Performance:")
print(f"Annualized Return: {perf_e[0]:.2f}%")
print(f"Annualized Volatility: {perf_e[1]:.2f}%")
print(f"Annualized Sharpe Ratio: {perf_e[2]:.2f}")
```

```
Optimal Weights Portfolio Performance:
Annualized Return: -0.03%
Annualized Volatility: 0.21%
Annualized Sharpe Ratio: -0.14


Constrained 18%——vola Portfolio Performance:
Annualized Return: -0.04%
Annualized Volatility: 0.21%
Annualized Sharpe Ratio: -0.21
```

In [ ]: