Exercise7Solution

October 6, 2022

[ ]:
```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import optimize
from scipy.stats import gmean
```

0.0.1 1) Collect 10 years of weekly data for McDonalds, Coca Cola and Microsoft for the period Jan. 1, 2011 to Jan. 1, 2021.

[ ]:
```
TICKERS = [
    "MCD",
    "KO",
    "MSFT"
]
START = "2011-01-01"
END = "2021-01-01"
INTERVAL = "1wk"
stock_data = yf.download(
    tickers = TICKERS,
    start = START,
    end = END,
    interval = INTERVAL
).dropna()
stock_data.tail()
```

```
[*********************100%***********************]  3 of 3 completed
```

[ ]:
```
            Adj Close                              Close              \
                   KO         MCD        MSFT         KO         MCD
Date
2020-11-30  50.666431  201.447357  211.198227  53.849998  210.740005
2020-12-07  50.589577  199.779144  210.114456  53.349998  207.759995
2020-12-14  50.959400  206.817978  215.365829  53.740002  215.080002
2020-12-21  50.674915  203.269714  219.464462  53.439999  211.389999
2020-12-28  52.002476  206.337204  219.139328  54.840000  214.580002

                             High                                Low  \
```

```
                   MSFT         KO        MCD       MSFT         KO
Date
2020-11-30  214.360001  53.869999  218.929993  217.320007  51.080002
2020-12-07  213.259995  53.779999  209.529999  216.949997  52.700001
2020-12-14  218.589996  54.220001  217.509995  220.889999  52.619999
2020-12-21  222.750000  53.549999  213.429993  225.630005  51.980000
2020-12-28  222.419998  54.930000  215.779999  227.179993  53.730000

                                         Open                          \
                   MCD        MSFT         KO        MCD        MSFT
Date
2020-11-30  209.130005  210.839996  52.090000  216.460007  214.100006
2020-12-07  206.190002  209.110001  53.759998  208.529999  214.369995
2020-12-14  210.210007  212.240005  53.650002  210.729996  213.100006
2020-12-21  208.009995  217.279999  52.680000  210.619995  217.550003
2020-12-28  210.779999  219.679993  53.849998  212.990005  224.449997

                 Volume
                     KO         MCD         MSFT
Date
2020-11-30  112526800.0  20745700.0  137480700.0
2020-12-07   62940500.0  15273500.0  138057400.0
2020-12-14  102322700.0  21713100.0  186693000.0
2020-12-21   34921900.0   9821800.0   89044300.0
2020-12-28   33978800.0   8681700.0   76551100.0
```

(a) Weekly returns

[ ]:
```
weekly_returns = stock_data["Adj Close"].pct_change().dropna()
weekly_returns.tail()
```

[ ]:
```
                  KO       MCD      MSFT
Date
2020-11-30  0.021822 -0.034764 -0.004042
2020-12-07 -0.001517 -0.008281 -0.005132
2020-12-14  0.007310  0.035233  0.024993
2020-12-21 -0.005583 -0.017156  0.019031
2020-12-28  0.026198  0.015091 -0.001481
```

(b) Annualised mean and covariance

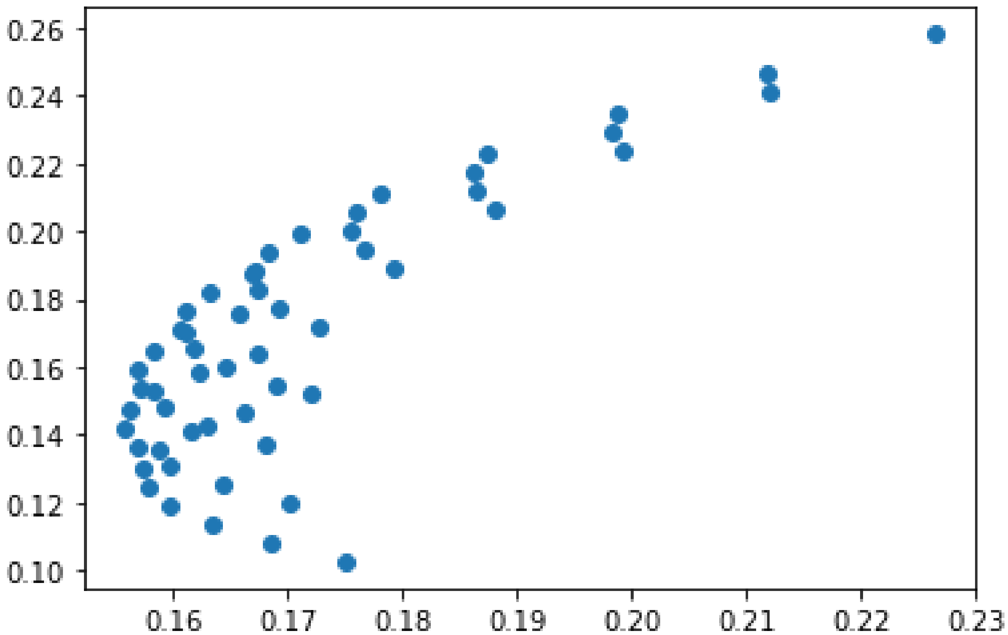
[ ]:
```
mean_weekly_returns = gmean(weekly_returns + 1) - 1
mean_annual_returns = (1 + mean_weekly_returns)**52 - 1
weekly_cov_matrix = weekly_returns.cov()
annual_cov_matrix = weekly_cov_matrix * 52
print("=== Mean weekly returns ===")
print(pd.Series(mean_weekly_returns, weekly_returns.columns))
print("=== Mean annual returns ===")
```

```
print(pd.Series(mean_annual_returns, weekly_returns.columns))
print("=== Annual Covariance Matrix ===")
print(annual_cov_matrix)
```

```
=== Mean weekly returns ===
KO      0.001576
MCD     0.002529
MSFT    0.004425
dtype: float64
=== Mean annual returns ===
KO      0.085340
MCD     0.140377
MSFT    0.258106
dtype: float64
=== Annual Covariance Matrix ===
            KO       MCD      MSFT
KO    0.033404  0.019477  0.017287
MCD   0.019477  0.032205  0.016835
MSFT  0.017287  0.016835  0.051306
```

(c) Portfolio weights In this exercise, we will try different combination of portfolios where a multiple of 1/10 is invested in each asset. There are many more options, but this gives us an idea of possible risk and return combinations

[ ]:
```
portfolio_means = []
portfolio_stds = []
portfolio_weights = []
for x in range(0, 10, 1):
    for y in range(0, 10 - x, 1):
        i = x/10
        j = y/10
        k = 1 - i - j
        w = np.array([[i, j, k]])
        portfolio_means.append((i * mean_annual_returns[0] + j *␣
↪mean_annual_returns[1] + k * mean_annual_returns[2]).squeeze())
        portfolio_stds.append(np.sqrt(w @ annual_cov_matrix @ w.T).squeeze())
        portfolio_weights.append(w)
portfolio_means = np.array(portfolio_means)
portfolio_stds = np.array(portfolio_stds)
_ = plt.scatter(portfolio_stds, portfolio_means)
```



(d) Maximal mean Maximal mean === 100% allocated to asset with highest return

[ ]:
```
idx = np.argmax(portfolio_means)
max_mean = portfolio_means[idx]
max_weights = portfolio_weights[idx]

print(max_mean, max_weights)
```

```
0.25810634543846667 [[0. 0. 1.]]
```

(e) Minimum std Minimum risk is non-trivial as covariance can be used to lower the overall risk (hence why the plot in (c) has a 1-to-many relation ship between std and mean)

[ ]:
```
idx = np.argmin(portfolio_stds)
min_std = portfolio_stds[idx]
min_weights = portfolio_weights[idx]

print(weekly_returns.columns.values)
print(min_weights)
print(min_std)
```

```
['KO' 'MCD' 'MSFT']
[[0.4 0.4 0.2]]
0.15569808070854343
```

(f) Highest (approximate) Sharpe-Ratio (Risk-Free Rate assumed to be 0) Most interesting portfolio as it has the best risk adjusted return.

[ ]:
```
idx = np.argmax(portfolio_means / portfolio_stds)
max_sharpe = (portfolio_means / portfolio_stds)[idx]
max_weights = portfolio_weights[idx]

print(weekly_returns.columns.values)
print(max_weights)
print(max_sharpe)
```

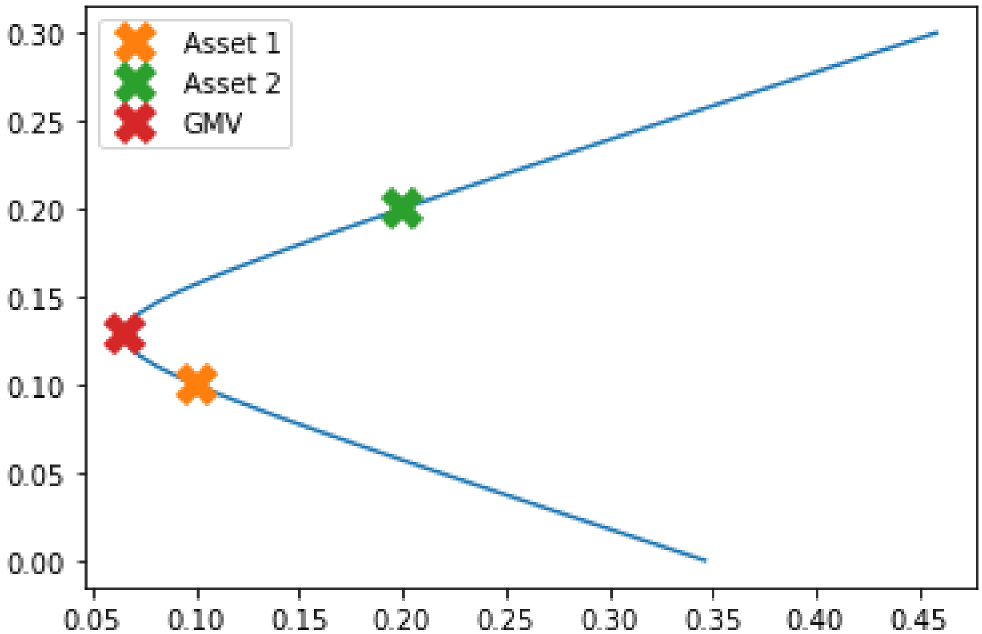
```
['KO' 'MCD' 'MSFT']
[[0.  0.3 0.7]]
1.1889941786455571
```

0.0.2 2) Portfolios, diversification, efficient frontier

Calculate using the analytical expressions derived from the constrained optimization problem. Change rho as needed

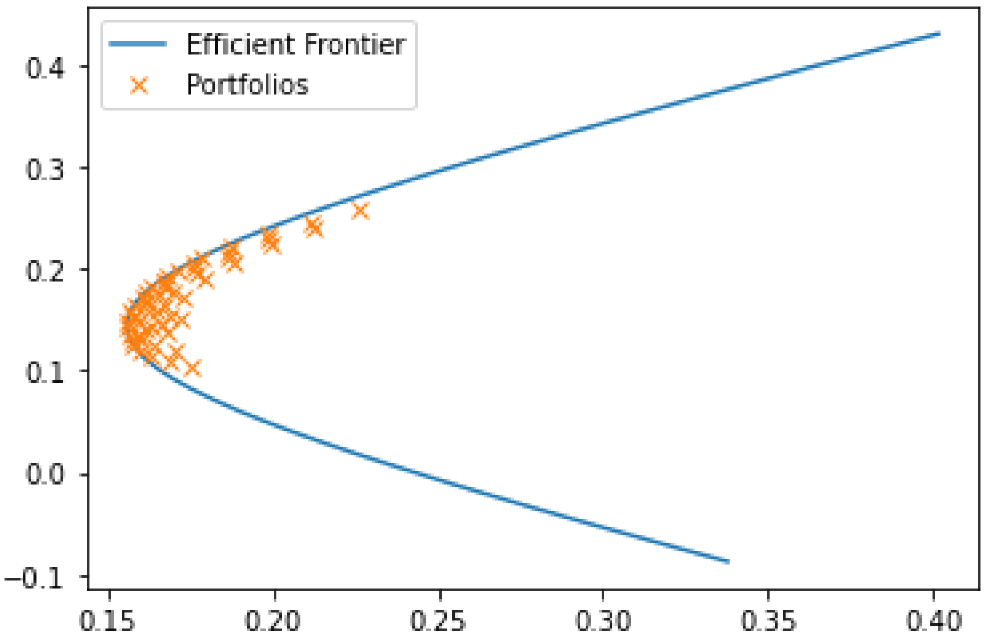
[ ]:
```
rho = -0.5

mu = np.array([[0.1, 0.2]]).T
Sigma = np.array([
    [0.1**2, 0.1*0.2*rho],
    [0.1*0.2*rho, 0.2**2]
])
Sigma_inv = np.linalg.inv(Sigma)
# Sigma_inv = np.linalg.pinv(Sigma) # Moore-Penrose pseudo-inverse (As Sigma is␣
 ↪singular for -1 and 1)

# Solve GMV
a = mu.T @ Sigma_inv @ mu
b = mu.T @ Sigma_inv @ np.ones_like(mu)
c = np.ones_like(mu).T @ Sigma_inv @ np.ones_like(mu)

mu_gmv = b/c
var_gmv = 1/c
w_gmv = 1/c * Sigma_inv * np.ones_like(mu)

# Calculate Efficient Frontier
mus = np.linspace(np.min(mu) - np.ptp(mu), np.max(mu) + np.ptp(mu))
sigmas = np.sqrt((c*mus**2 - 2*b*mus + a) / (a*c - b**2)).squeeze()

# Plots
plt.plot(sigmas, mus)
plt.plot(0.1, 0.1, "x", ms=12, mew=6, label="Asset 1")
plt.plot(0.2, 0.2, "x", ms=12, mew=6, label="Asset 2")
```

```
plt.plot(var_gmv**0.5, mu_gmv, "x", ms=12, mew=6, label="GMV")
plt.legend()
None
```



0.0.3 3) Efficient frontier for question 1

[ ]:
```
mu = mean_annual_returns.reshape(-1,1)
Sigma = annual_cov_matrix
Sigma_inv = np.linalg.inv(Sigma)
a = mu.T @ Sigma_inv @ mu
b = mu.T @ Sigma_inv @ np.ones_like(mu)
c = np.ones_like(mu).T @ Sigma_inv @ np.ones_like(mu)
mu_gmv = b/c
var_gmv = 1/c

mus = np.linspace(np.min(mu) - np.ptp(mu), np.max(mu) + np.ptp(mu))
sigmas = np.sqrt((c*mus**2 - 2*b*mus + a) / (a*c - b**2)).squeeze()

# Plots
plt.plot(sigmas, mus, label="Efficient Frontier")
plt.plot(portfolio_stds, portfolio_means, "x", label="Portfolios")
plt.legend()
None
```


Efficient Frontier
Portfolios
0.4
0.3
0.2
0.1
0.0
−0.1
0.15
0.20
0.25
0.30
0.35
0.40