In [1]:
```python
import numpy as np
```

In [2]:
```python
mu_full = np.array([0.83, 0.85, 0.96, 0.81, 0.68])

cov_full = np.array([[17.40, 15.20, 18.40, 11.94, 18.81],
             [15.20, 22.49, 18.74, 12.06, 20.96],
             [18.40, 18.74, 25.06, 13.82, 22.18],
             [11.94, 12.06, 13.82, 15.94, 14.95],
             [18.81, 20.96, 22.18, 14.95, 29.05]])
```

In [3]:
```python
mu1 = np.array([0.70, 0.90, 0.69, 0.51, 0.31])

cov1 = np.array([[14.48, 13.90, 17.27, 9.76, 19.20],
             [13.90, 22.75, 20.04, 10.52, 20.82],
             [17.27, 20.04, 26.47, 12.52, 24.42],
             [9.76, 10.52, 12.52, 13.43, 14.40],
             [19.20, 20.82, 24.42, 14.4, 32.62]])
```

In [4]:
```python
# mean vector and covariance matrix second sample period 2013:01-2022:12
mu2 = np.array([0.97, 0.79, 1.23, 1.10, 1.05])

cov2 = np.array([[20.43, 16.63, 19.62, 14.14, 18.49],
             [16.63, 22.41, 17.62, 13.74, 21.32],
             [19.62, 17.62, 23.72, 15.06, 19.93],
             [14.14, 13.74, 15.06, 18.41, 15.41],
             [18.49, 21.32, 19.93, 15.41, 25.46]])
```

In [22]:
```python
# Q1 a)

numer = np.linalg.inv(cov1) @ mu1
denom = np.ones(5) @ np.linalg.inv(cov1) @ mu1

w1_tp = np.divide(numer, denom)

numer = np.linalg.inv(cov1) @ np.ones(5)
denom = np.ones(5) @ np.linalg.inv(cov1) @ np.ones((5,1))

w1_gp = np.divide(numer, denom)

print('Tangency Portfolio Weights for period 1:\n', w1_tp,
      '\n\nMin. var. portfolio Weights for period 1:\n', w1_gp)
```

```
Tangency Portfolio Weights for period 1:
 [ 1.57874222  0.6227716  -0.37924521  0.22288215 -1.04515076]

Min. var. portfolio Weights for period 1:
 [ 1.10889977  0.23358726 -0.367573    0.51924118 -0.4941552 ]
```

In [6]:
```python
# Q1 b) summary sats of tangency portfolio

mu2_tp = w1_tp @ mu2
var2_tp = w1_tp.T @ cov2 @ w1_tp
sharpe2_tp = mu2_tp/np.sqrt(var2_tp)

print('Mean return of the TP in period 2: ', mu2_tp)
print('\nVariance of the TP in period 2: ', var2_tp)
print('\nSharpe Ratio of the TP in period 2: ', sharpe2_tp)
```

```
Mean return of the TP in period 2:  0.7046599760610504

Variance of the TP in period 2:  23.700207480868762

Sharpe Ratio of the TP in period 2:  0.14474498670951041
```

In [7]:
```
# Summary stats of min var portfolio

mu2_gp = w1_gp @ mu2
var2_gp = w1_gp.T @ cov2 @ w1_gp
sharpe2_gp = mu2_gp/np.sqrt(var2_gp)

print('Mean return of the MVP in period 2: ', mu2_gp)
print('\nVariance of the MVP in period 2: ', var2_gp)
print('\nSharpe Ratio of the MVP in period 2: ', sharpe2_gp)
```

```
Mean return of the MVP in period 2:  0.8603542508241728

Variance of the MVP in period 2:  18.339200911810877

Sharpe Ratio of the MVP in period 2:  0.20090331580689305
```

In [8]:
```
# Q1 c)

n = 12*20

rets_full = np.random.multivariate_normal(mu_full, cov_full, n) # sample from MVN
rets1 = rets_full[0:n//2, :] # take first half of sample
rets2 = rets_full[n//2:, :] # take second half of samples
# print(len(rets1), len(rets2))

mu1_samp = np.mean(rets1,axis=0)
mu2_samp = np.mean(rets2,axis=0)

cov1_samp = np.cov(rets1, ddof=1, rowvar=False)
cov2_samp = np.cov(rets2, ddof=1, rowvar=False)
```

In [21]:
```
# Calculated weights for TP & MVP implied by sample
numer = np.linalg.inv(cov1_samp) @ mu1_samp
denom = np.ones(5) @ np.linalg.inv(cov1_samp) @ mu1_samp

w1_tp_samp = np.divide(numer, denom)

numer = np.linalg.inv(cov1_samp) @ np.ones(5)
denom = np.ones(5) @ np.linalg.inv(cov1_samp) @ np.ones((5,1))

w1_gp_samp = np.divide(numer, denom)

print('Tangency Portfolio Weights for period 1:\n', w1_tp_samp,
      '\n\nMin. var. portfolio Weights for period 1:\n', w1_gp_samp)
```

```
Tangency Portfolio Weights for period 1:
 [ 0.96092531  0.51670545 -0.17733701  0.4926164  -0.79291016]

Min. var. portfolio Weights for period 1:
 [ 0.51892876  0.34989821 -0.18923686  0.68522893 -0.36481903]
```

In [19]:
```
# Calcaulte summary stats for tangency portfolio implied by sample
mu2_tp_samp = w1_tp_samp @ mu2_samp
var2_tp_samp = w1_tp_samp.T @ cov2_samp @ w1_tp_samp
sharpe2_tp_samp = mu2_tp_samp/np.sqrt(var2_tp_samp)

print('Mean return of the TP in period 2: ', mu2_tp_samp)
print('\nVariance of the TP in period 2: ', var2_tp_samp)
print('\nSharpe Ratio of the TP in period 2: ', sharpe2_tp_samp)
```

```
Mean return of the TP in period 2:  0.783914578346524

Variance of the TP in period 2:  13.72782316954373

Sharpe Ratio of the TP in period 2:  0.21157673107973482
```

In [20]:
```
# Calcaulte summary stats for min var portfolio implied by sample
mu2_gp_samp = w1_gp_samp @ mu2_samp
var2_gp_samp = w1_gp_samp.T @ cov2_samp @ w1_gp_samp
sharpe2_gp_samp = mu2_gp_samp/np.sqrt(var2_gp_samp)

print('Mean return of the MVP in period 2: ', mu2_gp_samp)
print('\nVariance of the MVP in period 2: ', var2_gp_samp)
print('\nSharpe Ratio of the MVP in period 2: ', sharpe2_gp_samp)
```

```
Mean return of the MVP in period 2:  0.757805179267897

Variance of the MVP in period 2:  13.012411074205737

Sharpe Ratio of the MVP in period 2:  0.21007708465513536
```

In [12]:
```
# Q2 calculating Bayes-Stein estimators for period 1

N = 5

mu1_gp = w1_gp @ mu1 # calculate mean return on MVP

frac1 = (N + 2)*(n/2-1)/(n/2-N-2)
frac2 = 1/((mu1-mu1_gp*np.ones(5)).T @ np.linalg.inv(cov1) @ (mu1-mu1_gp*np.ones(5)))

lamb = frac1 * frac2
gamma = lamb/(n/2+lamb)

mu1_bs = gamma * np.ones(5) * mu1_gp + (1-gamma) * mu1

frac1 = cov1 * (1 + 1/(n/2+lamb))
frac2 = lamb/(n/2*(n/2+1+lamb)) * np.ones(5) @ np.ones((5,1)) /(np.ones(5) @ np.linalg.i

cov1_bs = frac1+frac2

print('Bayes-Stein estimate of excess mean return:\n', mu1_bs,
      '\n\nBayes-Stein estimated of covariance:\n', cov1_bs)
```

```
Bayes-Stein estimate of excess mean return:
 [0.7924653  0.86444834 0.78886615 0.72408142 0.65209838]

Bayes-Stein estimated of covariance:
 [[14.75039944 14.16865985 17.54876747 10.01624278 19.48455611]
 [14.16865985 23.0452036  20.32707551 10.77852224 21.10941496]
 [17.54876747 20.32707551 26.77636097 12.78452083 24.72021242]
 [10.01624278 10.77852224 12.78452083 13.69725019 14.6701595 ]
 [19.48455611 21.10941496 24.72021242 14.6701595  32.94480662]]
```

In [13]:
```
# Calculating Bayes-Stein estimators for period 2
N = 5

numer = np.linalg.inv(cov2) @ np.ones(5)
denom = np.ones(5) @ np.linalg.inv(cov2) @ np.ones((5,1))

w2_gp = np.divide(numer, denom)

frac1 = (N + 2)*(n/2-1)/(n/2-N-2)
frac2 = 1/((mu2-mu2_gp*np.ones(5)).T @ np.linalg.inv(cov2) @ (mu2-mu2_gp*np.ones(5)))

lamb = frac1 * frac2
gamma = lamb/(n/2+lamb)

mu2_bs = gamma * np.ones(5) * mu2_gp + (1-gamma) * mu2

frac1 = cov2 * (1 + 1/(n/2+lamb))
frac2 = lamb/(n/2*(n/2+1+lamb)) * np.ones(5) @ np.ones((5,1)) /(np.ones(5) @ np.linalg.i
```

```
cov2_bs = frac1+frac2

print('Bayes-Stein estimate of excess mean return:\n', mu2_bs,
      '\n\nBayes-Stein estimated of covariance:\n', cov2_bs)
```

```
Bayes-Stein estimate of excess mean return:
 [0.8951799  0.83800835 0.97776104 0.93647047 0.92058948]

Bayes-Stein estimated of covariance:
 [[20.93754702 17.12748906 20.12540308 14.63089845 18.99241216]
 [17.12748906 22.92278774 18.12010942 14.22983972 21.8299027 ]
 [20.12540308 18.12010942 24.23625509 15.55333353 20.4362236 ]
 [14.63089845 14.22983972 15.55333353 18.91220042 15.90425992]
 [18.99241216 21.8299027  20.4362236  15.90425992 25.98086058]]
```

In [14]:

```
# Calculate TP weights for period 1 based on BS estimators
numer = np.linalg.inv(cov1) @ mu1_bs
denom = np.ones(5) @ np.linalg.inv(cov1) @ mu1_bs

w1_bs_tp = np.divide(numer, denom)
print('Weights of tangency portfolio based on Bayes-Stein estimators:\n', w1_bs_tp)
```

```
Weights of tangency portfolio based on Bayes-Stein estimators:
 [ 1.2780032   0.37366061 -0.37177401  0.41257707 -0.69246687]
```

In [15]:

```
# Q2 c) summary stats of tangency portfolio from Bayes-Stein

mu2_bs_tp = w1_bs_tp @ mu2
var2_bs_tp = w1_bs_tp.T @ cov2 @ w1_bs_tp
sharpe2_bs_tp = mu2_bs_tp/np.sqrt(var2_bs_tp)

print('Mean return of the TP in period 2: ', mu2_bs_tp)
print('\nVariance of the TP in period 2: ', var2_bs_tp)
print('\nSharpe Ratio of the TP in period 2: ', sharpe2_bs_tp)
```

```
Mean return of the TP in period 2:  0.8043175180292848

Variance of the TP in period 2:  19.686007533858866

Sharpe Ratio of the TP in period 2:  0.18127950400148277
```

In [16]:

```
print('Difference in mean returns (Traditional less BS):\n', mu2_tp-mu2_bs_tp)
print('\nDifference in Variance (Traditional less BS):\n', var2_tp-var2_bs_tp)
print('\nDifference in Sharpe Ratio (Traditional less BS):\n', sharpe2_tp-sharpe2_bs_tp)
```

```
Difference in mean returns (Traditional less BS):
 -0.0996575419682344

Difference in Variance (Traditional less BS):
 4.014199947009896

Difference in Sharpe Ratio (Traditional less BS):
 -0.03653451729197235
```

We see from the above that the Bayes-Stein estimators outperform the traditional estimation approach in all measures.

The BS estimates have a greater mean return (~0.1% higher), and a lower variance (~4% lower), making it the clear favourite in terms of mean-variance optimization.

In addition to this, it also achieves a greater Sharpe Ratio, meaning it provides a better trade-off in terms of risk-reward.

We see then that based off of the given set of returns data, that the Bayes-Stein estimates lead to much a much better performance in the estimation of our optimal portfolio.