Dylan Wong Xuan Kai **H00280633**

F79PA Portfolio Theory and Asset Models Assignment 2

Note that work uses term format as variance%%, covariance%%, and total returns%

1. Task 1 requires creating invertible variance-covariance matrix and 4 populations generated from random normal sample each with different mean and each having invertible variance-covariance matrix.

Line 7 is a list of all required packages for the code below. Line 10-29 installs and loads R packages that have yet to be acquired. Line 34-65 reads the 'Task_1_func' function which has the option of returning variance covariance matrix or matrix Y (matrix XA as stated in question). Line 67 and 69 calls the function with input being

- 'n' number of random samples generated for 'n' row matrix
- returns_matrix being either '1' for Y matrix or '0' for variance-covariance matrix as return value (Answers all of Task 1)

Line 72 is a byte code compiler function for 'Task 1 func' function which enables reused function above in later questions to be executed more quickly.

Output gives Y matrix of 100x4 dimension:

```
> Task_1_func(100,1)
            [,1]      [,2]       [,3]        [,4]
  [1,] -56.17676 -17.43550  -7.987523  102.822463
  [2,] 170.37406 169.93777 236.716295  204.961894
  [3,] 193.94141 219.19609 188.623031  132.819545
  [4,]  11.15006 229.23832 205.128298  186.492868
  [5,]  20.95643 134.63835 186.510320  -14.941587
  [6,] 232.63347 200.44770 245.742185  216.517060
  [7,]  85.20304 -62.48147  73.658322   82.080373
  [8,] 181.69625 215.32975 -33.982101 -146.070588
  [9,] 244.10027  50.32359 146.961701  307.485625
```

Output gives variance-covariance matrix of 4x4 dimension:

```
> Task_1_func(100,0)
           [,1]       [,2]       [,3]       [,4]
[1,] 10998.7659  1338.2907  1287.2869   535.7521
[2,]  1338.2907 10800.3837   872.4367   425.7972
[3,]  1287.2869   872.4367 10143.1049  1215.1272
[4,]   535.7521   425.7972  1215.1272 12386.4591
```

2. Task 2 requires estimating variance covariance matrix of previous 4 populations with different by generating different range of random samples thus determining if estimation improves as generated random sample increases.

Line 76-118 reads the 'est_cov_matrix' function which has the option of returning maximum absolute value of the estimated variance covariance matrix or estimated variance covariance matrix. Line 119 and 120 calls the function with input being

- 'n' number of random samples generated for 'n' to run and retrieve previous variance covariance matrix from 'Task_1_func' function
- with_max_abs_value being either '1' for maximum absolute value of the estimated variance covariance matrix or '0' for estimated variance-covariance matrix as return value (Answers all of Task 2)

Line 123 is a byte code compiler function for 'est_cov_matrix' function which enables reused function above in later questions to be executed more quickly.

Output gives for maximum absolute value of the estimated variance covariance matrix:

```
> est_cov_matrix(100,1)
max absolute value of all matrix entries is
[1] 12234.11
```

Output gives variance covariance matrix of 4x4 dimension:

```
> est_cov_matrix(100,0)
          asset1    asset2   asset3     asset4
asset1 8884.453 2779.5777 1031.734  3859.6357
asset2 2779.578 8622.9479 2192.218  -720.7563
asset3 1031.734 2192.2178 7458.479  3378.9958
asset4 3859.636 -720.7563 3378.996 12234.1059
```

Line 125-133 compares value difference between estimation of variance of variance covariance matrix of different number of random samples with the variance covariance matrix found in Task 1. As seen below difference reduces as number of generated random samples increases

Differences in values:

```
> sum(est_cov_matrix(100,0))-sum(cmp_Task_1_func(100,0))
[1] 6564.701
> # 62242.8%%
> sum(est_cov_matrix(500,0))-sum(cmp_Task_1_func(100,0))
[1] 3774.94
> # 59453.04%%
> sum(est_cov_matrix(1000,0))-sum(cmp_Task_1_func(100,0))
[1] 786.925
> # 56465.02%%
> sum(est_cov_matrix(10000,0))-sum(cmp_Task_1_func(100,0))
[1] 170.3221
> # Compare with covariance_matrix matrix
> sum(est_cov_matrix(100,0))-sum(cmp_Task_1_func(100,0))
[1] 6564.701
> # Difference: 6564.701
> sum(est_cov_matrix(500,0))-sum(cmp_Task_1_func(100,0))
[1] 3774.94
> # Difference: 3774.94
> sum(est_cov_matrix(1000,0))-sum(cmp_Task_1_func(100,0))
[1] 786.925
> # Difference: 786.925
> sum(est_cov_matrix(10000,0))-sum(cmp_Task_1_func(100,0))
[1] 170.3221
```

3. Task 3 vi) requires providing weights of Portfolio 1 of 25% return and Portfolio 2 of 35% return based on previous 4 populations as 4 assets. (From this point onwards, estimated variance covariance matrix from 10000 generated random samples will be used since its difference from variance covariance matrix of Task 1 is smaller.)

   Line 140-211 reads the 'portfolio_1_2_vi' function which has the option of returning the minimum variance or returns or weights of portfolio 1 or portfolio 2. Line 212 to 226 calls the function with input being:

   - '1', or '2', or '3' for minimum variance, or returns or weights of portfolio 1 respectively and '4', or '5', or '6' for minimum variance, or returns or weights of portfolio 2 respectively.

Line 229 is a byte code compiler function for 'portfolio_1_2_vi' function which enables reused function above in later questions to be executed more quickly.

<u>Returns of output as mentioned above(first 4 row values of weight being weights of asset 1,2,3 and 4 respectively) :</u>

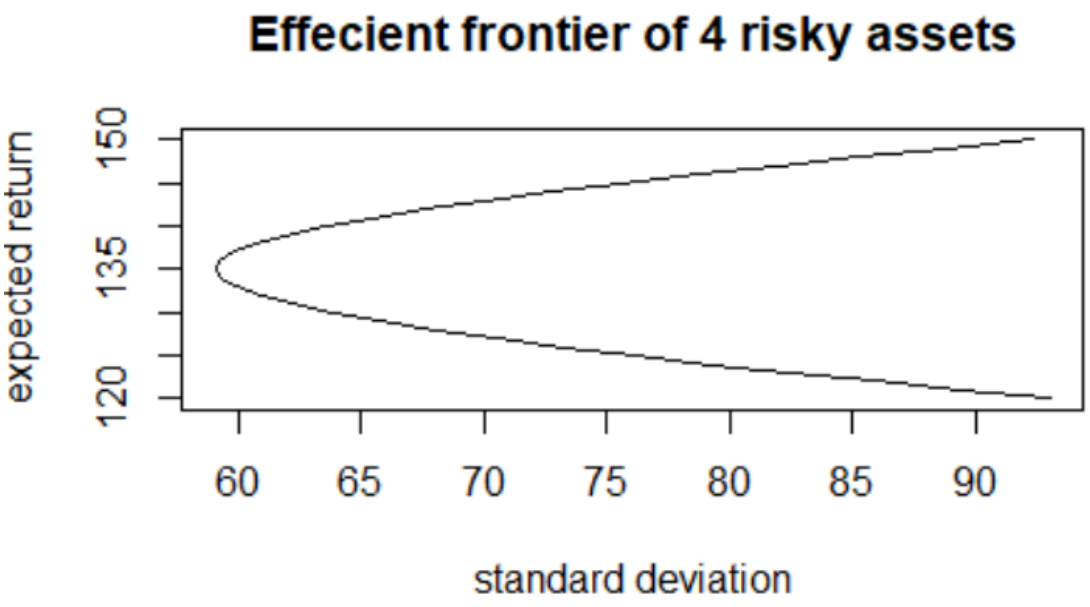
```
> portfolio_1_2_vi(1)
         [,1]
[1,] 5796.278
> # Returns
> portfolio_1_2_vi(2)
     [,1]
[1,]  125
> # Asset weights
> portfolio_1_2_vi(3)
             [,1]
asset1  0.5688668
asset2  0.3350953
asset3  0.1232092
asset4 -0.0271712
> # Portfolio 2
> # Variance
> portfolio_1_2_vi(4)
         [,1]
[1,] 3489.257
> # Returns
> portfolio_1_2_vi(5)
     [,1]
[1,]  135
> # Asset weights
> portfolio_1_2_vi(6)
            [,1]
asset1 0.2473506
asset2 0.2577209
asset3 0.2425066
asset4 0.2524220
```

4. Task 3 vii) requires forming a efficient frontier with expected return of portfolio ranging from 0.2 to 0.5.

Line 183-202 reads the 'effecient_frontier_vii' function which creates the efficient frontier by incrementing the expected return of portfolio by 1 from 233 to 259 (by total return%). Line 260 calls the function with no additional inputs. Line 263 is a byte code compiler

function for ‘effecient_frontier_vii’ function which enables reused function above in later questions to be executed more quickly.

Plot of the efficient frontier:



5. Task 3 viii) requires calculating expected returns and standard deviation of portfolio if risk-free rste 5% is included by first calculating the weights of tangency portfolio of the risky assets.

Line 267-297 reads the ‘weight_or_value_tangency_port’ function which returns weights, or expected returns, or standard deviation of the tangency portfolio. Line 299-301 calls the function with input being:

- ‘1’, or ‘2’, or ‘3’ for weights, or expected returns, or standard deviation of the tangency portfolio respectively

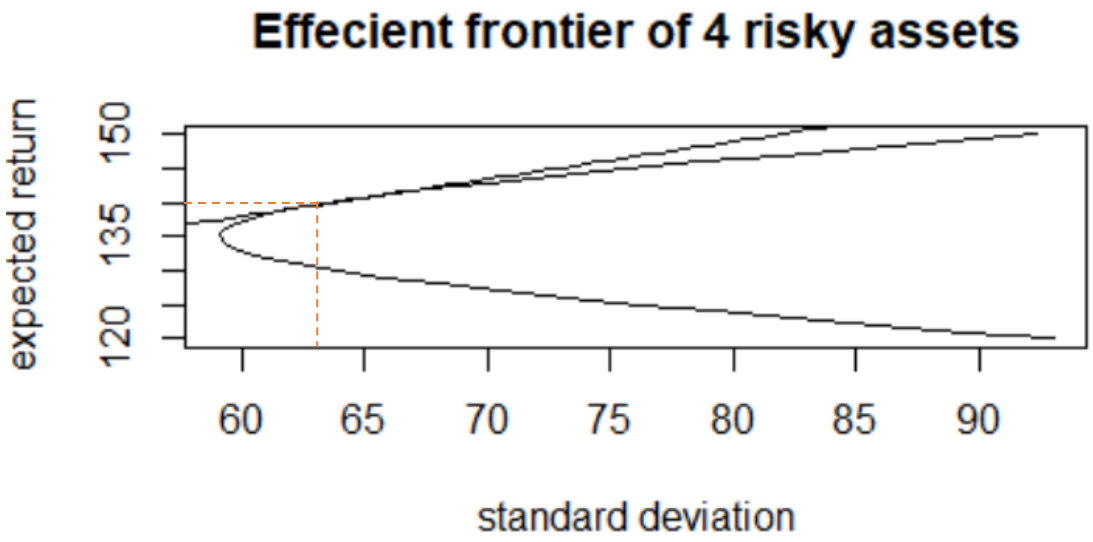
Output of code:

```
> weight_or_value_tangency_port(1)
              [,1]
asset1 0.08003362
asset2 0.21745524
asset3 0.30458898
asset4 0.39792216
> weight_or_value_tangency_port(2)
         [,1]
[1,] 140.204
> weight_or_value_tangency_port(3)
          [,1]
[1,] 63.89262
```

6. Task 3 ix) requires drawing Capital Market Line (CML)in a same graph with previous efficient frontier.

Line 308-332 reads the' CML' function which plots the CML by incrementing the standard deviation of portfolio by 0.5 from 0 to 100 (by %) along with the efficient frontier before. Line 333 calls the function with no additional inputs.

Output shows that it matches the expected returns and standard deviation from Task 3 viii) which corresponds to the tangency of portfolio:



7. Task 3 x) requires determining if weights of global minimum variance portfolio matches the formula of calculating the weights given in question and if not, explain origin of difference and also find the minimum variance of all portfolio.

Line 337-356 reads the 'est_weight_lagrange' function which returns the weights of portfolio based on the written Lagrange method below. Line 357 then calls the function with no additional inputs.

Lagrange equations and associated system of equations:

Minimize:

$$f(\pi_1, \pi_{2,}\pi_3, \pi_4) = \sum_{i=1}^{4} sd_i^2 \pi_i^2 + \sum_{i=1}^{4} \sum_{j=1, j!=i}^{4} sd_{ij}\pi_{ij}$$

Constraints:

$$g_1(\pi_1, \pi_{2,}\pi_3, \pi_4) = \sum_{i=1}^{4} \pi_i = 1$$

Form Larangian:

$$L(\pi_1, \pi_2, \pi_3, \pi_4, alpha, beta) = \sum_{i=1}^{4} sd_i^2 \pi_i^2 + \sum_{i=1}^{4} \sum_{j=1, j!=i}^{4} sd_{ij} \pi_{ij} - alpha(\sum_{i=1}^{4} \pi_i - 1)$$

Then differentiate above Lagrangian to arrive at the required equations:

$$\frac{dL}{d\pi_1} = 2sd_1^2\pi_1 + 2sd_{12}\pi_2 + 2sd_{13}\pi_3 + 2sd_{14}\pi_4 - alpha = 0$$

$$\frac{dL}{d\pi_2} = 2sd_2^2\pi_2 + 2sd_{12}\pi_1 + 2sd_{23}\pi_3 + 2sd_{24}\pi_4 - alpha = 0$$

$$\frac{dL}{d\pi_3} = 2sd_3^2\pi_3 + 2sd_{13}\pi_1 + 2sd_{32}\pi_2 + 2sd_{34}\pi_4 - alpha = 0$$

$$\frac{dL}{d\pi_4} = 2sd_4^2\pi_4 + 2sd_{41}\pi_1 + 2sd_{42}\pi_2 + 2sd_{43}\pi_3 - alpha = 0$$

$$\frac{dL}{dg_1} = \sum_{i=1}^{4} \pi_i = 1$$

Estimated variance and covariance of n=10000 made in Task 2 v) from cmp_est_cov_matrix(10000,0):

	asset1	asset2	asset3	asset4
asset1	11189.6945	1274.99952	1289.852	436.57532
asset2	1274.9995	11355.96847	1003.664	54.89911
asset3	1289.8525	1003.66380	10770.033	1147.69820
asset4	436.5753	54.89911	1147.698	12117.34426

Estimated weights from Lagrange method :

```
> asset_final_weight
             [,1]
asset1 0.2445032
asset2 0.2570356
asset3 0.2435631
asset4 0.2548980
```

Line 360-383 reads the 'est_weight_formula' function which returns weights of portfolio based on formula provided in question. Line 385-387 calls the function with input 'weight_or_variance' as 0 for weights and 1 for variance%%.

Estimated weights and variance%% from formula provided in question :

```
> # weights
> est_weight_formula_or_var(0)
             [,1]
asset1 0.2445032
asset2 0.2570356
asset3 0.2435631
asset4 0.2548980
> # Min variance
> est_weight_formula_or_var(1)
         [,1]
[1,] 3489.079
```

Line 390-395 runs the 'weight_diff' function which checks thr difference in weight values for Lagrangian method and the formula. Line 396 calls the function with no additional inputs. Indeed it seems similar for 7 decimal points but when subtracted from each other small difference exist :

```
> weight_diff()
                 [,1]
asset1  2.775558e-17
asset2 -5.551115e-17
asset3 -2.775558e-17
asset4 -5.551115e-17
```

Lagrange method solves the matrix in one step while the formula provided in the question requires calculating weights of portfolio (nominator of equation), then calculating the value to be divided by(denominator of equation) which is a two step method, therefore leading to small approximation difference for the weights since some rounding up must have been made for the nominator of equation(weights) before being divided by the value of denominator.

8. Task 3 xi) requires finding weights of portfolio of the 4 assets with additional constraints of weight of asset 1 = weight asset 2.

Line 400-419 reads the 'portfolio_weight_xi' function which returns the weights of each asset of portfolio. Line 420 calls the function with no additional inputs.

Lagrange equations and associated system of equations:

Minimize:

$$f(\pi_1, \pi_2, \pi_3, \pi_4) = \sum_{i=1}^{4} sd_i^2 \pi_i^2 + \sum_{i=1}^{4} \sum_{j=1, j!=i}^{4} sd_{ij} \pi_{ij}$$

Constraints:

$$g_1(\pi_1, \pi_{2,}\pi_3, \pi_4) = \pi_1 - \pi_2 = 0$$

$$g_2(\pi_1, \pi_{2,}\pi_3, \pi_4) = \sum_{i=1}^{4} \pi_i = 1$$

Form Lagrangian:

$$L(\pi_1, \pi_{2,}\pi_3, \pi_4, alpha, beta) = \sum_{i=1}^{4} sd_i^2 \pi_i^2 + \sum_{i=1}^{4} \sum_{j=1,j!=i}^{4} sd_{ij}\pi_{ij} - alpha(\pi_1 - \pi_2) - beta(\sum_{i=1}^{4} \pi_i - 1)$$

Then differentiate above Lagrangian to arrive at the required equations:

$$\frac{dL}{d\pi_1} = 2sd_1^2\pi_1 + 2sd_{12}\pi_2 + 2sd_{13}\pi_3 + 2sd_{14}\pi_4 - alpha - beta = 0$$

$$\frac{dL}{d\pi_2} = 2sd_2^2\pi_2 + 2sd_{12}\pi_1 + 2sd_{23}\pi_3 + 2sd_{24}\pi_4 + alpha - beta = 0$$

$$\frac{dL}{d\pi_3} = 2sd_3^2\pi_3 + 2sd_{13}\pi_1 + 2sd_{32}\pi_2 + 2sd_{34}\pi_4 + 0 - beta = 0$$

$$\frac{dL}{d\pi_4} = 2sd_4^2\pi_4 + 2sd_{41}\pi_1 + 2sd_{42}\pi_2 + 2sd_{43}\pi_3 + 0 - beta = 0$$

$$\frac{dL}{dg_1} = \pi_1 - \pi_2 = 0$$

$$\frac{dL}{dg_2} = \sum_{i=1}^{4} \pi_i = 1$$

Output of the code:

```
> portfolio_weight_xi()
             [,1]
asset1 0.2508855
asset2 0.2508855
asset3 0.2434583
asset4 0.2547706
```