Problem 1:

Use the data in problem1.csv. Fit a Normal Distribution and a Generalized T distribution to this data.

Calculate the VaR and ES for both fitted distributions.

Overlay the graphs the distribution PDFs, VaR, and ES values. What do you notice? Explain the differences.

Answer:

(1) Normal Distribution:

```
def norm_ES(x,alpha):
    miu = x.mean()
    x_adj=x-miu
    sigma = x_adj.std()
    norm_VaR = -norm.ppf(alpha,loc=miu,scale=sigma)


    ES_norm =- miu + sigma*norm.pdf(norm.ppf(alpha))/alpha
    print("When alpha = "+str(alpha)+" :")
    print("The VaR fitted a Normal Distribution is "+str(norm_VaR))
    print("The Expected Shortfall is "+str(ES_norm))

    return norm_VaR, ES_norm
```

(2) T distribution:

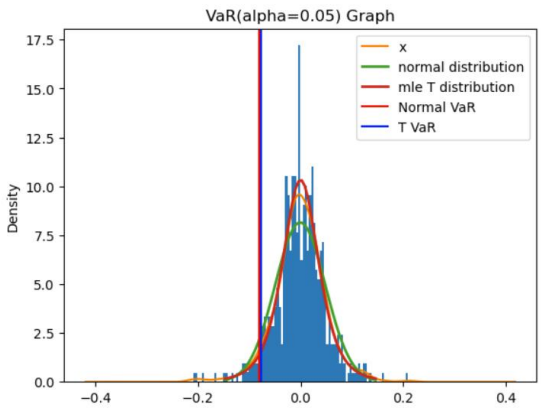
```
# t distribution:
def MLE_t(pars, x):
    df = pars[0]
    loc=pars[1]
    scale = pars[2]
    ll = np.log(t.pdf(x, df=df,loc=loc,scale=scale))
    return -ll.sum()


def t_ES(x, alpha):
    mean_x=x.mean()
    std_x=x.std()
    cons = ({'type': 'ineq', 'fun': lambda x:  x[0] - 2}, {'type': 'ineq', 'fun': lambda x:  x[2]})
    model = optimize.minimize(fun = MLE_t,  x0 = [2, mean_x,std_x ], constraints=cons, args =x).x

    df=model[0]
    loc=model[1]
    scale=model[2]
    t_sample = t.rvs(df =df, loc = loc, scale = scale, size = 10000)

    t_VaR=-t.ppf(alpha, df =df, loc = loc, scale = scale)

    ES_t=-t_sample[t_sample<-t_VaR].mean()

    print("When alpha = "+str(alpha)+" :")
    print("The VaR fitted a T Distribution is "+str(t_VaR))
    print("The Expected Shortfall is "+str(ES_t))

    return  df,loc,scale,t_VaR,ES_t
```

Result: set alpha=0.05

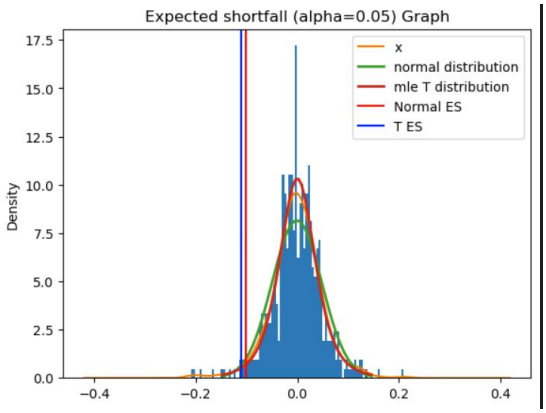
```
When alpha = 0.05 :
The VaR fitted a Normal Distribution is 0.0813353274763361
The Expected Shortfall is 0.10177426982904988
When alpha = 0.05 :
The VaR fitted a T Distribution is 0.07647580735513254
The Expected Shortfall is 0.1115321584017986
```

PDF graphs:

VaR



ES



When we set alpha=0.05, the VaR fitted a Normal Distribution is 0.0813, the Expected Shortfall is 0.1018; and the VaR fitted a T distribution is 0.0765, the Expected Shortfall is 0.1115.

The VaR fitted a Normal Distribution is higher than the VaR fitted a T distribution, the Normal Expected Shortfall is lower than T Expected Shortfall.

T distribution is a better fit for the real world scenario which is more outlier-prone. Using a distribution with large kurtosis and skewness can help us gain more information on the risks associated with this asset.

Problem 2:

In your main repository, create a Library for risk management. Create modules, classes, packages, etc as you see fit. Include all the functionality we have discussed so far in class. Make sure it includes

1. Covariance estimation techniques.

2. Non PSD fixes for correlation matrices

3. Simulation Methods

4. VaR calculation methods (all discussed)

5. ES calculation

Create a test suite and show that each function performs as expected.

Answer:

Put all the functions into 5 .py files(modules), storing them in a file called risk_manage.

I have tested all the function, which runs correctly.

<1> Covariance estimation techniques.

```
test_ewCov(0.97)
✓ 2.0s
array([[8.41106909e-05, 1.06945662e-04, 1.21760871e-04, ...,
        1.25484463e-04, 8.11331555e-05, 8.61130395e-05],
       [1.06945662e-04, 2.68752303e-04, 1.97531665e-04, ...,
        1.15658764e-04, 3.74977522e-05, 8.22220854e-05],
       [1.21760871e-04, 1.97531665e-04, 2.91157502e-04, ...,
        8.30278956e-05, 3.31844912e-05, 7.34713753e-05],
       ...,
       [1.25484463e-04, 1.15658764e-04, 8.30278956e-05, ...,
        7.47889224e-04, 2.68371109e-04, 2.00639601e-04],
       [8.11331555e-05, 3.74977522e-05, 3.31844912e-05, ...,
        2.68371109e-04, 3.08241679e-04, 8.21009546e-05],
       [8.61130395e-05, 8.22220854e-05, 7.34713753e-05, ...,
        2.00639601e-04, 8.21009546e-05, 2.62692778e-04]])
```

<2> Non PSD fixes for correlation matrices

```
test_psd(10)
✓ 0.0s
Output exceeds the size limit. Open the full output data in a text editor
The expected result is
[[1.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.9        0.43588989 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.9        0.20647416 0.38388595 0.         0.         0.
  0.         0.         0.         0.        ]
 [0.9        0.20647416 0.12339191 0.36351459 0.         0.
  0.         0.         0.         0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.35259655 0.
  0.         0.         0.         0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.34578204
  0.         0.         0.         0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.05658252
  0.34112115 0.         0.         0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.05658252
  0.04797016 0.3377314  0.         0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.05658252
  0.04797016 0.04163812 0.33515484 0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.05658252
  0.04797016 0.04163812 0.03678529 0.33313002]]
The testing result is
[[1.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.9        0.43588989 0.         0.         0.         0.
...
  0.04797016 0.04163812 0.33515484 0.        ]
 [0.9        0.20647416 0.12339191 0.08842247 0.06898628 0.05658252
  0.04797016 0.04163812 0.03678529 0.33313002]]
The imported function is correct
```

```
test_convert_psd(10)
✓ 0.0s

Higham testing:
The fixed matrix is PSD
near_psd testing:
The fixed matrix is PSD
The Frobenius_Norm of Higham method for nearest psd = 0.0015192331193787274
The Frobenius_Norm of near_psd method for nearest psd = 0.0027385481538506153
```

<3> Simulation Methods

```
Direct simulation:
[[-0.01673409  0.00083232 -0.00759468 ... -0.00077084 -0.00096411
  -0.02670777]
 [-0.02630922 -0.01851255  0.00825677 ... -0.01608659 -0.00296129
  -0.0379463 ]
 [-0.02133345 -0.01413956  0.00064282 ... -0.02192374 -0.01779261
  -0.04542748]
 ...
 [-0.03541664 -0.02407248 -0.01880319 ...  0.03181294  0.00096401
  -0.04431997]
 [-0.01376027  0.00225925 -0.00359221 ...  0.00964495 -0.01391344
  -0.04001734]
 [-0.03612949  0.03627932 -0.03669508 ...  0.00981373  0.03524604
  -0.00461439]]
PCA simulation
[[ 0.00084302  0.00792584 -0.02345503 ...  0.00446801  0.00110221
   0.01946197]
 [-0.00626658  0.00827256 -0.03202709 ...  0.00971634 -0.01259621
   0.02236809]
 [-0.00987264  0.01723051 -0.0429156  ...  0.013579    0.00479619
   0.01681443]
 ...
 [-0.00191269  0.01877185 -0.0394351  ...  0.00559977 -0.02277556
   0.05840807]
 [ 0.01872835 -0.0057785  -0.01964404 ... -0.00076426  0.00889491
...
                  mean: 111.5746070799195
                  standard deviation:0.48228533536246887
                  Expected mean: 100
                  Expected standard deviation:0.5
```

<4> VaR calculation methods (all discussed)

```
test_VaR(0.05)
✓ 0.1s

When alpha = 0.05
The VaR of using a normal distribution = 0.06560156967533283
The VaR of using a normal distribution with an Exponentially Weighted variance (λ = 0. 94) = 0.09138526093846899
The VaR of using a MLE fitted T distribution = 0.05725638549603684
The VaR of using a fitted AR(1) model = 0.06586001439007666
The VaR of using a Historic Simulation = 0.06090740598532037
                                                    + Code   + Markdown
```

<5> ES calculation

```
test_es(Problem1['x'],0.05)
✓ 0.0s

When alpha = 0.05 :
The VaR fitted a Normal Distribution is 0.0813353274763361
The Expected Shortfall is 0.10177426982904988
When alpha = 0.05 :
The VaR fitted a Normal Distribution is 0.0813353274763361
The Expected Shortfall is 0.10177426982904988
When alpha = 0.05 :
The VaR fitted a Normal Distribution is 0.07647580735513254
The Expected Shortfall is 0.11741092683547402
When alpha = 0.05 :
The VaR fitted a Normal Distribution is 0.07647580735513254
The Expected Shortfall is 0.11741092683547402
```

Problem 3:

Use your repository from #2.

Using Portfolio.csv and DailyPrices.csv. Assume the expected return on all stocks is 0.

This file contains the stock holdings of 3 portfolios. You own each of these portfolios.

Fit a Generalized T model to each stock and calculate the VaR and ES of each portfolio as well as your

total VaR and ES. Compare the results from this to your VaR form Problem 3 from Week 4.

Answer:

The result of copula fitted t distribution:

	A	B	C	Total
Simulated VaR	7717.163142	6272.646845	5658.126112	21171.263981
Simulated ES	10230.651487	9254.666443	7087.761763	26876.180282
Historical VaR	9138.874264	6205.736843	5296.980071	21370.128438
Historical ES	10234.016059	8716.987158	7432.809565	25777.296720

The previous result:

```
When alpha = 0.05:
* Delta Normal VaRs:
       portfolio A:(VaRret): 0.01890382331530241
       portfolio B:(VaRret): 0.015267725564605946
       portfolio C:(VaRret): 0.014022179383209496
       portfolio Total:(VaRret): 0.015707326971388668
       portfolio A:(VaR$): 5670.202920147335
       portfolio B:(VaR$): 4494.59841077826
       portfolio C:(VaR$): 3786.589010809051
       portfolio Total:(VaR$): 13577.075418977081
* Historical VaRs:
       portfolio A:(VaR$): 8304.068056280026
       portfolio B:(VaR$): 6201.592045270023
       portfolio C:(VaR$): 5281.497635230015
       portfolio Total:(VaR$): 18652.46984949999
```

To sum up, the result of copula fitted t distribution is relatively larger than the previous results. But the they both follow the relationship: port total > port A> port B> port C.

In my option, copula would be the better option in calculating VaR and ES as not all of the financial data are not normally distributed.