Response of Project 4

Problem 1

Fit a Normal Distribution and a Generalized T distribution to this data

I first use MLE to estimate data's mean and standard deviation.

```
def mleT(data):
    def negLogLikeForT(initialParams):
        df, mean, sigma = initialParams
        return -t(df=df, loc=mean, scale=sigma).logpdf(data).sum()
    initialParams = np.array([2, data.mean(), data.std()])
    cons = ({'type': 'ineq', 'fun': lambda x:  x[0] - 2},
            {'type': 'ineq', 'fun': lambda x:  x[2]})
    df, mean, sigma = minimize(negLogLikeForT, x0=initialParams,
constraints=cons).x
    return df, mean, sigma


def mleNormal(data):
    def negLogLikeForNormal(initialParams):
        mean, sigma = initialParams
        return -norm(loc=mean, scale=sigma).logpdf(data).sum()
    initialParams = np.array([data.mean(), data.std()])
    cons = ({'type': 'ineq', 'fun': lambda x:  x[1]})
    mean, sigma = minimize(negLogLikeForNormal, x0=initialParams,
constraints=cons).x
    return mean, sigma
```

I then constructed a VaR function that accepts any distribution and returns the quantile with specified alpha, and a ES function that accepts any distribution, simulates n draws of data, and returns the mean beyond the quantile with specified alpha. dist represents any distribution provided by scipy.stats, and **kwargs is used to deal with loc, scale, df etc,. in the quantile function.

```
def VaR_distribution(dist, alpha,  **kwargs):
    return -dist.ppf(alpha, **kwargs)

def ES_distribution(dist, alpha, size, **kwargs):
    var = - VaR_distribution(dist, alpha, **kwargs)
    numbers = dist.rvs(size=size, **kwargs)
    return - numbers[numbers < var].mean()
```

Calculate the VaR and ES for both fitted distributions

Finally, I called the MLE functions, VaR function, and ES function.

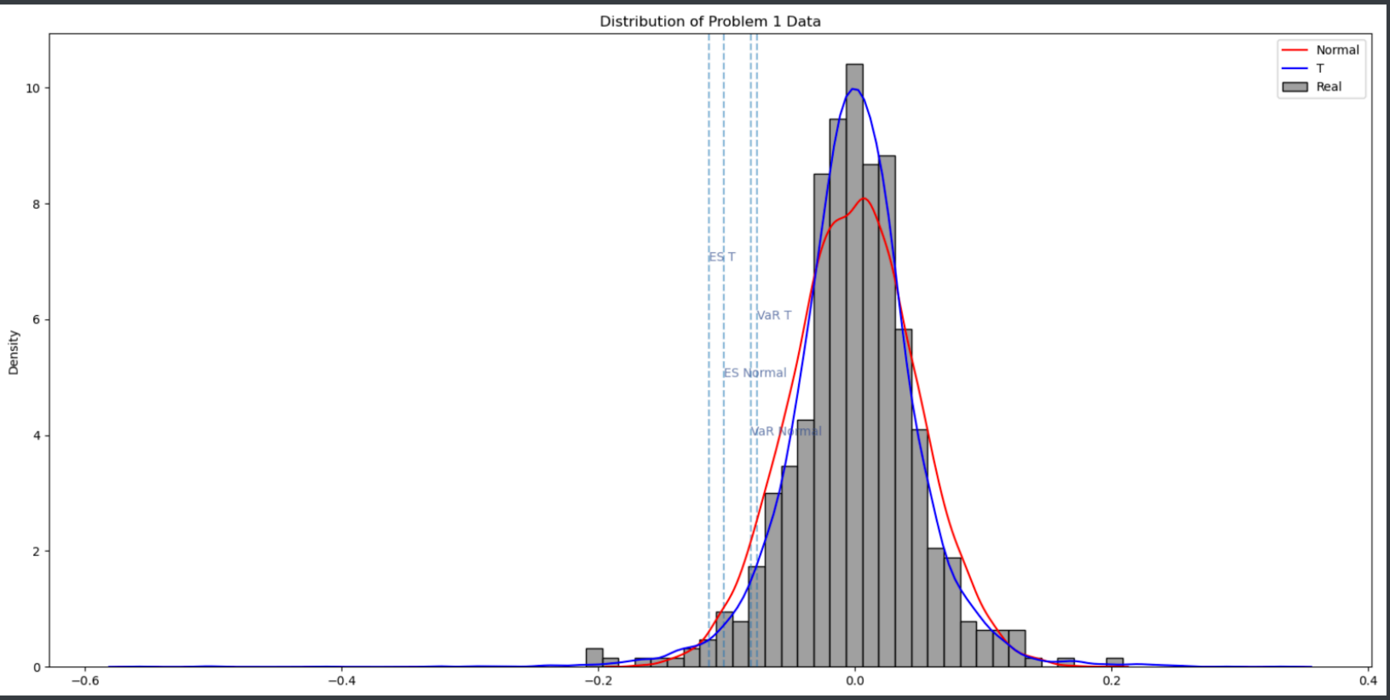
```
mean, sigma = mleNormal(data)
VaR = VaR_distribution(norm, alpha, loc=mean, scale=sigma)
ES = ES_distribution(norm, alpha, size, loc=mean, scale=sigma)

df, mean, sigma = mleT(data)
VaR = VaR_distribution(t, alpha, df=df, loc=mean, scale=sigma)
ES = ES_distribution(t, alpha, size, df=df, loc=mean, scale=sigma)
```

Results are shown below:

```
VaR with normal distribution:  0.08125483134935281
ES with normal distribution:  0.10156403066903118
VaR with t distribution:  0.07647580320576054
ES with t distribution:  0.1151056088947588
```

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



First, t distribution fits the data better. It has fatter tails and a higher peak than the normal distribution.

Second, t distribution gives more pessimistic ES and VaR than those given by the normal distribution. ES T is larger than ES Normal, while VaR T is larger than VaR Normal.

Third, ES gives larger losses than those given by VaR. ES T is larger than VaR T, while ES Normal is larger than VaR Normal.

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

- Covariance estimation techniques.
- Non PSD fixes for correlation matrices
- Simulation Methods
- VaR calculation methods (all discussed)
- ES calculation

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

See RiskManagementPackage and tests.

Problem 3

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.

My steps:

1. Given DailyPrice.csv, transform prices to returns using return_calculate function
2. Input stock returns to copulaSimulation function. The function outputs simulated returns. Specifically in the copulaSimulation function:
    1. For each stock ***Xi***,
        - Use MLE to find the parameters for each ***Xi*** under t distribution
        - transform ***Xi*** into a uniform vector ***Ui***, where ***Ui*** is ***Xi***'s CDF following t distribution
    2. Calculate the Spearman correlation given matrix ***U***
    3. Input the Spearman correlation to Cholesky factorization function. The function returns the root of the covariance matrix (here correlation matrix equals to covariance matrix since std is 1)
    4. Input the root to dataSimulation function. The function returns n draws of data following multivariate normal distribution
    5. For each ***Ui***,
        - transform ***Ui*** into a uniform variable using the standard normal CDF ***Ui'***
        - transform ***Ui'*** into ***Xi***'s quantile function following t distribution to backout ***Xi'***
3. Calculate the current prices, holdings, and current values for each portfolio
4. Multiply current values with simulatedReturns element-wise, sum at columns to get portfolio loss
5. Pass in portfolio loss to VaR_raw, VaR_distribution, ES_raw, ES_distribution given in problem 1to calculate VaR and ES.

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

Results given below:

	historicVaR	monteCarloVaR	historicES	monteCarloES
A	5828.615711	5943.576003	7734.717622	7745.748892
B	4671.423424	4608.387004	6432.190991	6461.287835
C	3416.638676	3478.926346	4838.452787	4842.090148
ABC	13159.146889	13275.911589	17969.186963	17799.386151

Results in Week04:

	historicVaR	deltaNormalVaR
A	5298.490894	6003.221296
B	5576.130254	4886.596045
C	3307.758237	3679.556066
Total	12460.873738	14100.550125

The results are similar. Very generally speaking, after considering the dependency structure among different stocks and setting the marginal distribution of the portfolio to be t distributions, VaRs tend to become larger than those deriving from historical distributions.