# Fintech545 Project Week5

Kaiye Chen

October 2022

## 1 Question1

### 1.1 Expected Shortfall Calculation

```
#ES Calculation
def ES(ret_arr, alpha):
    VaR_t = -np.percentile(ret_arr, alpha*100)
    return_before_alpha = ret_arr[ret_arr < -VaR_t]
    ES = return_before_alpha.mean()
    print(f"Expected Shortfall: {round(-ES*100,2)}%")
    return -ES
```

In this part, I defined function to calculate expected shortfall.

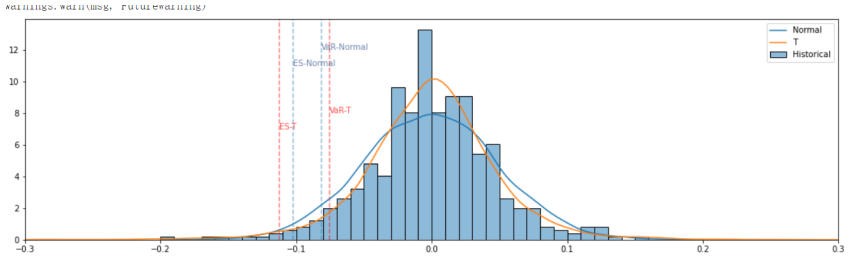
### 1.2 Results

```
normal_results = simulate_normal(data, 5, 10000)
normal = Return_norm_simulation(data, 10000)
normal_es = ES(normal, 0.05)
```

```
Normal distribution VaR: 8.12%
Expected Shortfall: 10.2%
```

```
[68] t_results = simulate_MLE_t(data, 5, 10000)
     t_es = ES(t_results[1], 0.05)
```

```
T distribution VaR: 7.52%
Expected Shortfall: 11.22%
```




In this part, I used the formerly defined VaR functions and the ES function below to calculate the results under normal distribution and t distribution. As shown in the graph, t distribution has a lower VaR than Normal Distribution, but higher ES. The reason is that t distribution has more fat-tail and sharp-peak characteristics than the normal distribution.

# 2 Question2

This is a refactoring question. All my refactoring and tests are shown in my code. If some functions are used in this weeks questions, I won't test it individually in test part.

# 3 Question3

## 3.1 Copula Method

```
def copula_t_simulation(ret, nOfDarws):
    n = ret.shape[1]
    stock_cdf = pd.DataFrame()
    t_params = []

    for col in ret.columns:
        ret[col] -= ret[col].mean()
        df, mean, scale = Return_T_params(ret[col])
        t_params.append([df, mean, scale])
        stock_cdf[col] = t.cdf(ret[col], df=df, loc=mean, scale=scale)

    Corr_spearman = spearmanr(stock_cdf)[0]
    cholesky = chol_psd(Corr_spearman)
    simuNormal = pd.DataFrame(norm.rvs(size=(n, nOfDarws)))
    simulatedT = (cholesky @ simuNormal).T
    Simu_data = pd.DataFrame()
    for i in range(n):
        simu = norm.cdf(simulatedT.iloc[:, i])
        Simu_data[ret.columns[i]] = t.ppf(simu, df=t_params[i][0], loc=t_params[i][1], scale=t_params[i][2])

    return Simu_data
```

In this part, I defined function to do copula simulation.

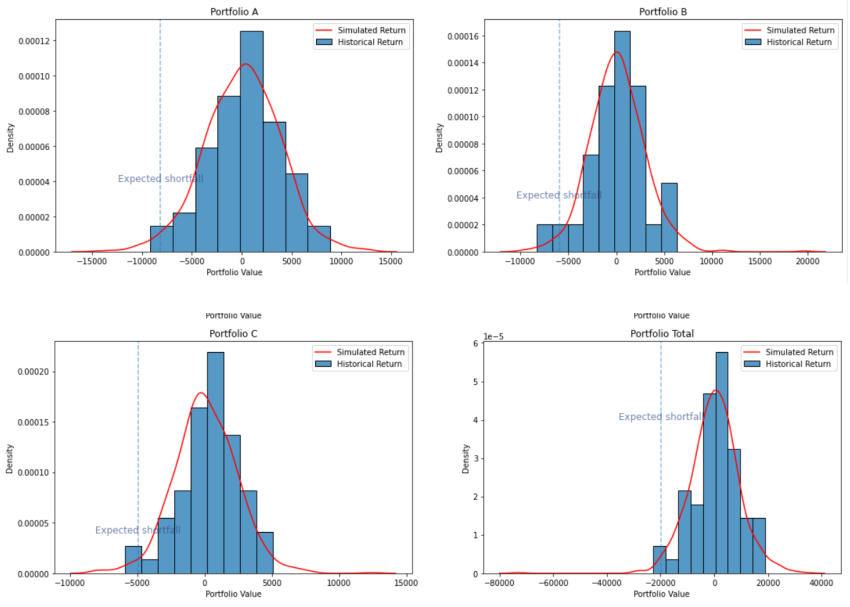
## 3.2 Data Manipulation

```
[84]  #import  and  modify  data
      portfolio  =  pd.read_csv("portfolio.csv")
      portfolio[portfolio["Portfolio"]=="A"]
      total  =  portfolio.groupby('Stock').sum('Holding')
      total['Portfolio']  =  'total'
      total.reset_index(inplace=True)
      portfolio  =  portfolio.append(total)

      prices  =  pd.read_csv("DailyPrices.csv",index_col=0)
      return_set  =  prices.pct_change().dropna()
      current  =  pd.DataFrame({"Price":prices.iloc[-1]})
```

In this part, I do some adjustment to the raw data in order to analyst it more easily.

## 3.3 Results




| | A | B | C | Total |
|---|---|---|---|---|
| **MC VaR** | 6282.60 | 4667.75 | 3393.31 | 15053.18 |
| **MC ES** | 8157.96 | 5926.41 | 4941.10 | 19759.65 |
| **Historical VaR** | 5920.57 | 4676.30 | 3768.38 | 14619.22 |
| **Historical ES** | 7622.93 | 7231.08 | 4881.32 | 19735.33 |

In this part, the graphs and plots show the final results of 4 portfolios. I think copula model is a better fit, because not all stock returns follow the exact standard normal distribution.