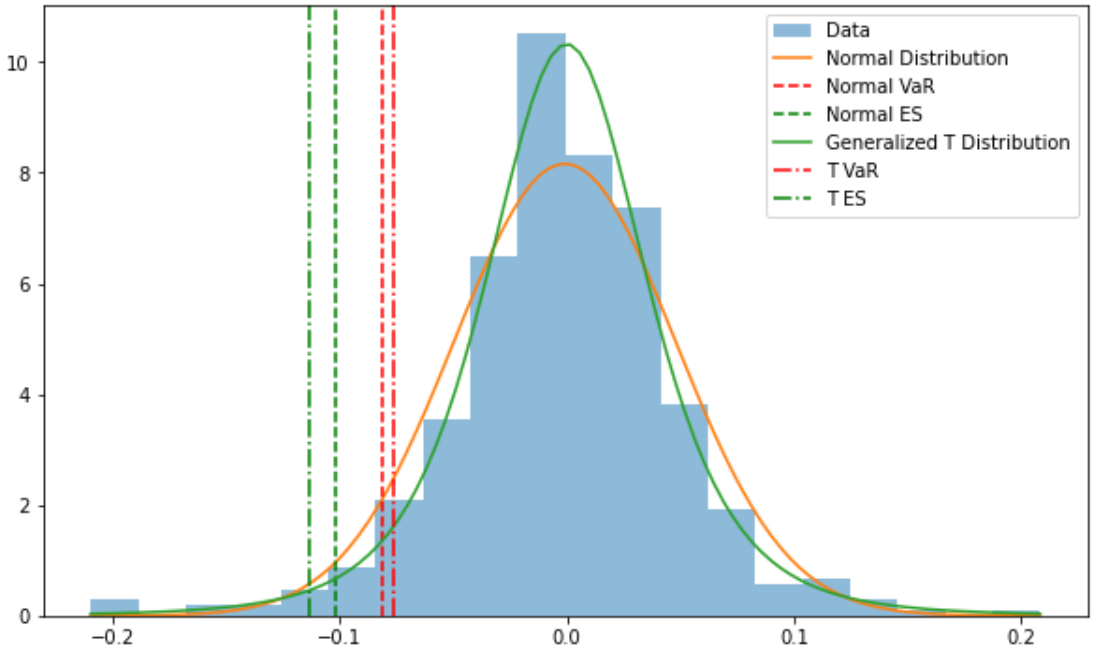
Problem 1

I use stats.norm and stats.t from scipy to fit the data into normal and generalized T distribution. Then I calculate the VaR and ES by definition (Expected Shortfall is the average of the left tail of the distribution, where the tail is defined as values whose CDF is less than or equal to Alpha):

- Normal Distribution:
  - VaR: 0.08
  - ES: 0.10
- Generalized T Distribution:
  - VaR: 0.08
  - ES: 0.11

I plot PDFs and VaR/ES values for both distributions:



Problem 2

In this problem, I test exp_weighted_cov, near_psd, chol.psd, and direct_simulation, and they work out as expected. The function of calculating VaR and ES will be tested in the next problem.

Problem 3

For this problem, I used a copula to do a joint simulation of portfolio returns using the t-distribution and principal component analysis (PCA). Here are the steps:

1. Daily returns calculation: Firstly I calculate daily returns for a set of stock prices using the pct_change() function and subtracting the mean using np.mean(). The resulting dataframe returns contains the daily returns for each stock in the original prices dataframe.
2. Portfolio pricing: The portfolio_price function takes a set of stock prices and a portfolio of stocks as inputs and returns the current value of each stock in the portfolio. The function first creates an empty list pv, then iterates over each stock in the portfolio, retrieves its current price using iloc[-1], and appends it to the pv list. The resulting pv list contains the current value of each stock in the portfolio.
3. Portfolio creation: I then create four different portfolios (A, B, C, and total) based on a portfolio dataframe that contains information on each stock's portfolio and holding. The tolist() method is used to convert the dataframe columns to lists.
4. Portfolio daily returns: The code calculates the daily returns for each portfolio using the logarithmic returns of the stocks in each portfolio. The np.diff() function calculates the difference between each day's logarithmic return and the previous day's return.
5. cal_t_pVals function: This function takes a portfolio, its daily returns, and the current prices for each stock in the portfolio as inputs. The function first fits a t-distribution to each stock's returns using t.fit(), which returns the degrees of freedom, location, and scale parameters. It then calculates the cumulative distribution function (CDF) for each stock's returns using t.cdf(). The resulting return_cdf dataframe contains the CDF values for each stock in the portfolio.

I use 95% PCA to simulate new returns based on the correlation between the stocks in the portfolio. The s.simulate_pca() function from my simulation_methods module is used to simulate 10,000 sets of new returns based on the correlation matrix of the return_cdf dataframe. The resulting sample dataframe contains the simulated returns for each stock in the portfolio.

I then calculate the CDF for each simulated return using norm.cdf(), which assumes a normal distribution of returns. The resulting sample_cdf list contains the CDF values for each simulated return.

Next, the function calculates the simulated returns using t.ppf(), which takes the CDF values, the degrees of freedom, location, and scale parameters, and returns the simulated returns. The resulting simu_return array contains the simulated returns for each stock in the portfolio.

Finally, the function calculates the potential portfolio values using the simulated returns and current prices for each stock in the portfolio. It multiplies each simulated return by

the current price and adds them up using matrix multiplication. The resulting pVals list contains the potential portfolio values for each simulation, sorted in ascending order.

The last step is to calculate the VaR and ES using my var library. Here are the results:

- Simulating with 22 PC Factors: 95.01 % total variance explained
- Simulating with 22 PC Factors: 95.01 % total variance explained
- portfolio A
- VaR = 8119.592085576267
- ES = 10699.212707121353

- Simulating with 23 PC Factors: 95.47 % total variance explained
- Simulating with 23 PC Factors: 95.47 % total variance explained
- portfolio B
- VaR = 6707.263129159866
- ES = 8815.59925480243

- Simulating with 23 PC Factors: 95.27 % total variance explained
- Simulating with 23 PC Factors: 95.27 % total variance explained
- portfolio C
- VaR = 5580.703319037799
- ES = 7376.550165893091

- Simulating with 52 PC Factors: 95.03 % total variance explained
- Simulating with 52 PC Factors: 95.03 % total variance explained
- portfolio total
- VaR = 20293.365268705296
- ES = 26671.293641570177

This result is higher than my VaR from Problem 3 from Week 4.