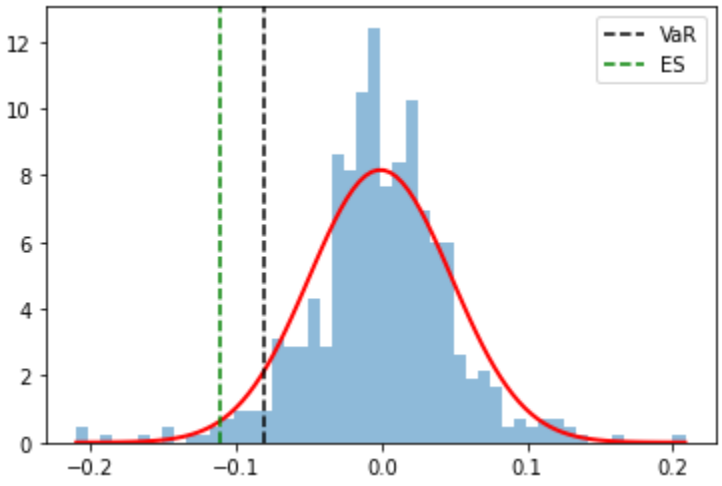
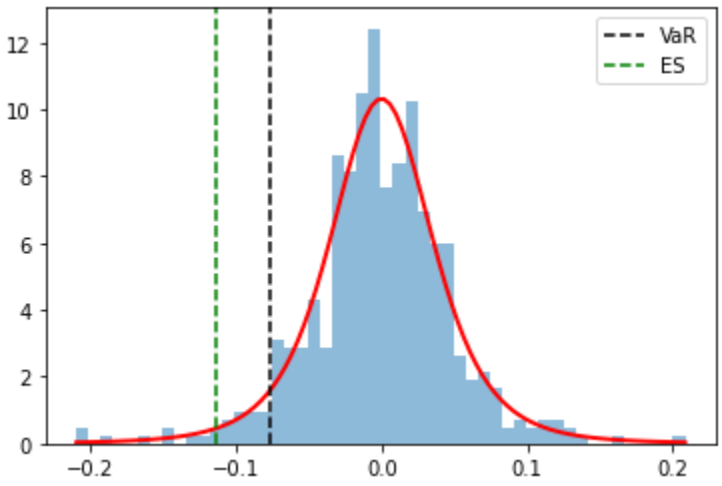
Problem1

With problem 1, for the normal fitted I got the values of VaR: 8.1255 ,ES: 11.1715 with the graph being shown below:



For the general t fitted distribution I got the values of: VaR: 7.6476 ES: 11.4446 with the graph looking like:



The major thing that I noticed and which can be explained by the rule of anything over a nvalue over 30 is that the VaR and ES of the normal fitted and T fitted were almost identical. Only difference I can see is that the normal fitted had fatter tails but only slightly.

Problem 2

For problem 2 I ran into multiple issues trying to compile everything. It became a nightmare that cascaded into problems for number 3. At first I tried to write my own package which can be found in the QuantRiskBookTools folder of the repo. I got it to fully build, and installed but when I went to call the package, the module couldn't be found. So then because of this I had to scramble to re build everything into functions in QuantRiskTools.py. This then caused multiple headaches where using the code created headaches with jupyter where I had to restart kernels and rerun everything as I had to change. That's why it looks like a mess in number 3. some functions I built before then bundled multiple functions into a new function for QuantRiskTools. The newer functions kept breaking with simulations using PCA and fit_general_t() function

Problem 3

This became very problematic from what I thought would be a simple build. I initially went the path of trying to use my fit_general t function, then finding the spearman correlation. After that I used the Simulate_pca to run simulations. From here I thought I was home free but realized I had spearman correlation of 100k sims. This is where things got messy. I needed a return matrix for my var function that calculated Delta VaR .I tried calculating figuring out how to get from correlations sims to get to returns from correlations but realized I needed to build U matrix and use gaussian copula which I am still struggling to grasp how to do.

Which led me back to square one. Next best option I had was to use my monte carlo VaR function which had simulation capabilities. I thought I had quick fix by changing the normal.multivariate function which was used to calculate VaR to _multivariate_t but this arose a massive headache. At first my system couldn't build it cause the package didn't exist which led to me having to uninstall scipy and reinstall it to the newest version. This then corrupted old a few old functions causing more problems. Once I got it to run, I kept getting error that "rvs() received unexpected parameter scale=" which was mind boggling because that was a parameter needed, and this got me stuck for a long time.

Eventually I tried running it without scale and got values in the 10s of thousands instead of the expected 1000s. At first I thought the code was broken and wandered through thinking its broken. Eventually I thought as a temporary fix to just divide the outputs by 10 in the function. At first this didn't make sense but once I looked at the output from when I was trying to do the method with simulate_PCA and saw the t distribution was between -1,1 for the most part. Even though I'm 80% sure this isn't right, it was the quickest fix that got some values close to my monte carlo simulation last week depending on what random sim output was given. Below is the example of the output of the tDist

```
PortfolioATDist
```

✓ 0.0s Python

	AAPL	TSLA	JPM	HD	BAC	XOM	AVGO	PEP	TMO	CMCSA	...	SBUX	GE	ISRG
	Price	Price	Price	Price	Price	Price	Price	Price	Price	Price	...	Price	Price	Price
Date														
2022-02-15	1.086783	0.553900	1.348196	1.272150	1.284249	1.090247	1.066632	1.874866	1.236357	1.277080	...	1.185401	1.127375	0.965169
2022-02-16	-0.057427	-0.032865	-0.050829	-0.102707	-0.032219	-0.079524	-0.045153	-0.147221	-0.042221	-0.112509	...	-0.040300	-0.091172	-0.068056
2022-02-17	-0.986178	-0.508227	-1.190066	-1.211881	-1.109890	-1.026618	-0.947537	-1.776222	-1.085674	-1.231248	...	-1.042623	-1.075326	-0.904650
2022-02-18	-0.436234	-0.223612	-0.517602	-0.571292	-0.477629	-0.464852	-0.410322	-0.877516	-0.472344	-0.583493	...	-0.456768	-0.500005	-0.414114
2022-02-22	-0.829586	-0.425757	-0.999561	-1.035881	-0.932490	-0.866200	-0.793293	-1.543242	-0.913055	-1.052612	...	-0.879126	-0.914531	-0.766980
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023-02-03	1.141809	0.583606	1.414313	1.332315	1.344529	1.147107	1.121366	1.950170	1.295569	1.338641	...	1.240756	1.183264	1.013001
2023-02-06	-0.834910	-0.428539	-1.006053	-1.041956	-0.938561	-0.871647	-0.798518	-1.551461	-0.918955	-1.058768	...	-0.884734	-0.920042	-0.771690
2023-02-07	0.911742	0.460739	1.136930	1.076097	1.090117	0.909874	0.893633	1.621147	1.046179	1.077041	...	1.006509	0.947332	0.811467
2023-02-08	-0.822291	-0.421947	-0.990663	-1.027547	-0.924167	-0.858737	-0.786135	-1.531944	-0.904968	-1.044167	...	-0.871436	-0.906974	-0.760522
2023-02-09	-0.320548	-0.165110	-0.375205	-0.429719	-0.342042	-0.347111	-0.298562	-0.661478	-0.341335	-0.441058	...	-0.330195	-0.375839	-0.308866

After multiple trials and errors for this problem, the best answer I could get for all the portfolios are:

Portfolio A VaR$: $8759.08

Portfolio A VaRret: 0.03

Portfolio A ES$: $11297.51

Portfolio A ESret: 0.04

Portfolio B VaR$: $8895.14

Portfolio B VaRret: 0.03

Portfolio B ES$: $11145.48

Portfolio B ESret: 0.04

Portfolio C VaR$: $8422.09

Portfolio C VaRret: 0.03

Portfolio C ES$: $10735.81

Portfolio C ESret: 0.04

Portfolio Total VaR$: $15341.25

Portfolio Total VaRret: 0.02

Portfolio Total ES$: $18785.47

Portfolio Total ESret: 0.02

Going forward, I will try to improve the QRT build and try to rebuild the deployed package.