FinTech545 - hw5

Samuel Fuller

March 2024

1 Problem 1

Please see below the outputs after running each individual test case using the functions I implemented as part of my library (all located in Library.functions.py). Matrix outputs have been compared directly as well as by their norms. All parameter, VaR, and ES tests have been compared directly as well. The code used to run the code is in tests.py and all the outputs can be found in the test_outputs folder.

- Section 1: Matrix Fixes
  - Test 1.1: Covariance Missing data, skip missing rows

Realized

	x1	x2	x3	x4	x5
	2.148513	-1.389470	-0.516466	-0.129327	-1.056814
	-1.389470	1.035342	0.339993	0.193888	0.626876
	-0.516466	0.339993	0.942388	0.947887	0.051788
	-0.129327	0.193888	0.947887	1.113436	-0.204731
	-1.056814	0.626876	0.051788	-0.204731	0.592027

Expected

	x1	x2	x3	x4	x5
	2.148513	-1.389470	-0.516466	-0.129327	-1.056814
	-1.389470	1.035342	0.339993	0.193888	0.626876
	-0.516466	0.339993	0.942388	0.947887	0.051788
	-0.129327	0.193888	0.947887	1.113436	-0.204731
	-1.056814	0.626876	0.051788	-0.204731	0.592027

Realized Norm: 4.2198060746438015
Expected Norm: 4.219806074643802

  - Test 1.2: Correlation Missing data, skip missing rows

Realized

	x1	x2	x3	x4	x5
	1.000000	-0.931618	-0.362959	-0.083616	-0.937042
	-0.931618	1.000000	0.344202	0.180583	0.800698
	-0.362959	0.344202	1.000000	0.925357	0.069333
	-0.083616	0.180583	0.925357	1.000000	-0.252163
	-0.937042	0.800698	0.069333	-0.252163	1.000000

Expected

	x1	x2	x3	x4	x5
	1.000000	-0.931618	-0.362959	-0.083616	-0.937042
	-0.931618	1.000000	0.344202	0.180583	0.800698
	-0.362959	0.344202	1.000000	0.925357	0.069333
	-0.083616	0.180583	0.925357	1.000000	-0.252163
	-0.937042	0.800698	0.069333	-0.252163	1.000000

Realized Norm: 3.493299901940298
Expected Norm: 3.493299901940298

– Test 1.3: Covariance Missing data, Pairwise

Realized

x1	x2	x3	x4	x5
1.173986	-0.629631	-0.278932	-0.081448	-0.735140
-0.629631	1.318197	0.018090	0.446047	0.139309
-0.278932	0.018090	0.918102	0.360836	0.258613
-0.081448	0.446047	0.360836	0.894764	-0.235190
-0.735140	0.139309	0.258613	-0.235190	0.522607

Expected

x1	x2	x3	x4	x5
1.173986	-0.629631	-0.278932	-0.081448	-0.735140
-0.629631	1.318197	0.018090	0.446047	0.139309
-0.278932	0.018090	0.918102	0.360836	0.258613
-0.081448	0.446047	0.360836	0.894764	-0.235190
-0.735140	0.139309	0.258613	-0.235190	0.522607

Realized Norm: 2.831484302766953
Expected Norm: 2.831484302766953

– Test 1.4: Correlation Missing data, pairwise

Realized

x1	x2	x3	x4	x5
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Expected

x1	x2	x3	x4	x5
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Realized Norm: 2.8139740361993284
Expected Norm: 2.8139740361993284

- Section 2: Exponentially Weighted Evaluation

– Test 2.1: EW Covariance, lambda=0.97

Realized

0	1	2	3	4
0.855911	0.127559	0.186929	0.081415	0.052412
0.127559	1.087350	0.032715	0.112515	-0.432729
0.186929	0.032715	0.744771	0.131065	0.065806
0.081415	0.112515	0.131065	0.868810	0.113836
0.052412	-0.432729	0.065806	0.113836	1.139180

Expected

x1	x2	x3	x4	x5
0.855911	0.127559	0.186929	0.081415	0.052412
0.127559	1.087350	0.032715	0.112515	-0.432729
0.186929	0.032715	0.744771	0.131065	0.065806
0.081415	0.112515	0.131065	0.868810	0.113836
0.052412	-0.432729	0.065806	0.113836	1.139180

Realized Norm: 2.2614680002352796
Expected Norm: 2.2614680002352796

- Test 2.2: EW Correlation, lambd=0.94

Realized

0	1	2	3	4
1.000000	0.109711	0.218511	0.116902	0.059677
0.109711	1.000000	-0.046716	0.191773	-0.444896
0.218511	-0.046716	1.000000	0.184148	0.089927
0.116902	0.191773	0.184148	1.000000	0.122028
0.059677	-0.444896	0.089927	0.122028	1.000000

Expected

x1	x2	x3	x4	x5
1.000000	0.109711	0.218511	0.116902	0.059677
0.109711	1.000000	-0.046716	0.191773	-0.444896
0.218511	-0.046716	1.000000	0.184148	0.089927
0.116902	0.191773	0.184148	1.000000	0.122028
0.059677	-0.444896	0.089927	0.122028	1.000000

Realized Norm: 2.3961597869136297
Expected Norm: 2.3961597869136293

- Test 2.3: Covariance with EW Variance (l=0.94), EW Correlation (l=0.97)

Realized

0	1	2	3	4
0.855911	0.105840	0.174461	0.100809	0.058928
0.105840	1.087350	-0.042040	0.186396	-0.495153
0.174461	-0.042040	0.744771	0.148129	0.082832
0.100809	0.186396	0.148129	0.868810	0.121399
0.058928	-0.495153	0.082832	0.121399	1.139180

Expected

x1	x2	x3	x4	x5
0.855911	0.105840	0.174461	0.100809	0.058928
0.105840	1.087350	-0.042040	0.186396	-0.495153
0.174461	-0.042040	0.744771	0.148129	0.082832
0.100809	0.186396	0.148129	0.868810	0.121399
0.058928	-0.495153	0.082832	0.121399	1.139180

Realized Norm: 2.2985123369892113
Expected Norm: 2.2985123369892118

- Section 3: PSD/PD matrix fixes

- Test 3.1:

Realized

0	1	2	3	4
1.173986	-0.617989	-0.284559	-0.065152	-0.688287
-0.617989	1.318197	0.017092	0.445696	0.139176
-0.284559	0.017092	0.918102	0.354147	0.246056
-0.065152	0.445696	0.354147	0.894764	-0.218717
-0.688287	0.139176	0.246056	-0.218717	0.522607

Expected

x1	x2	x3	x4	x5
1.173986	-0.617989	-0.284559	-0.065152	-0.688287
-0.617989	1.318197	0.017092	0.445696	0.139176
-0.284559	0.017092	0.918102	0.354147	0.246056
-0.065152	0.445696	0.354147	0.894764	-0.218717
-0.688287	0.139176	0.246056	-0.218717	0.522607

Realized Norm: 2.7961523098556302

Expected Norm: 2.7961523098556293

– Test 3.2:

Realized

0	1	2	3	4
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Expected

x1	x2	x3	x4	x5
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Realized Norm: 2.8139740361993293
Expected Norm: 2.8139740361993275

– Test 3.3:

Realized

0	1	2	3	4
1.186714	-0.628321	-0.283719	-0.073916	-0.714017
-0.628321	1.318332	0.017597	0.446822	0.141484
-0.283719	0.017597	0.919902	0.358003	0.250669
-0.073916	0.446822	0.358003	0.899221	-0.222690
-0.714017	0.141484	0.250669	-0.222690	0.557664

Expected

x1	x2	x3	x4	x5
1.173986	-0.623870	-0.294335	-0.057677	-0.693888
-0.623870	1.318197	0.016449	0.448579	0.143703
-0.294335	0.016449	0.918102	0.354067	0.246866
-0.057677	0.448579	0.354067	0.894764	-0.217062
-0.693888	0.143703	0.246866	-0.217062	0.522607

Realized Norm: 2.830965956835944
Expected Norm: 2.8044534385842383

– Test 3.4:

Realized

0	1	2	3	4
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Expected

x1	x2	x3	x4	x5
1.000000	-0.483199	-0.241787	-0.067767	-0.714761
-0.483199	1.000000	0.015446	0.405660	0.178286
-0.241787	0.015446	1.000000	0.488250	0.336248
-0.067767	0.405660	0.488250	1.000000	-0.322136
-0.714761	0.178286	0.336248	-0.322136	1.000000

Realized Norm: 2.813974036199328
Expected Norm: 2.8139740361993275

- Section 4: Cholesky

Realized

0	1	2	3	4
1.083506	0.000000	0.000000	0.000000	0.000000
-0.570360	0.996437	0.000000	0.000000	0.000000
-0.262628	-0.133175	0.911807	0.000000	0.000000
-0.060130	0.412871	0.431384	0.731160	0.000000
-0.635240	-0.223938	0.054179	-0.256892	0.000000

Expected

x1	x2	x3	x4	x5
1.083506	0.000000	0.000000	0.000000	0.000000
-0.570360	0.996437	0.000000	0.000000	0.000000
-0.262628	-0.133175	0.911807	0.000000	0.000000
-0.060130	0.412871	0.431384	0.731160	0.000000
-0.635240	-0.223938	0.054179	-0.256892	0.000000

Realized Norm: 2.1971928150679045
Expected Norm: 2.197192815067904

- Section 5: Normal/PCA Simulation (all cov matrices)
  - Test 5.1:

Realized

0	1	2	3	4
0.085334	0.088514	0.042605	0.009057	0.003913
0.088514	0.162094	0.058743	0.012484	0.005397
0.042605	0.058743	0.037668	0.006018	0.002596
0.009057	0.012484	0.006018	0.001699	0.000553
0.003913	0.005397	0.002596	0.000553	0.000317

Expected

x1	x2	x3	x4	x5
0.085347	0.087720	0.042334	0.009020	0.003888
0.087720	0.160529	0.058067	0.012351	0.005324
0.042334	0.058067	0.037468	0.005974	0.002574
0.009020	0.012351	0.005974	0.001692	0.000549
0.003888	0.005324	0.002574	0.000549	0.000315

Realized Norm: 0.24865812988951916
Expected Norm: 0.2466132264784241

  - Test 5.2

Realized

0	1	2	3	4
0.084909	0.116685	0.042084	0.008991	0.003870
0.116685	0.160352	0.057833	0.012356	0.005318
0.042084	0.057833	0.037271	0.005940	0.002564
0.008991	0.012356	0.005940	0.001690	0.000547
0.003870	0.005318	0.002564	0.000547	0.000314

Expected

x1	x2	x3	x4	x5
0.085347	0.117287	0.042374	0.009028	0.003892
0.117287	0.161180	0.058232	0.012406	0.005349
0.042374	0.058232	0.037524	0.005989	0.002581
0.009028	0.012406	0.005989	0.001695	0.000550
0.003892	0.005349	0.002581	0.000550	0.000316

Realized Norm: 0.2690991895520819

Expected Norm: 0.2705620778801814

– Test 5.3:

Realized

0	1	2	3	4
0.084744	0.008502	0.037672	0.007985	0.003458
0.008502	0.160414	0.051833	0.011034	0.004750
0.037672	0.051833	0.037307	0.005989	0.002583
0.007985	0.011034	0.005989	0.001681	0.000550
0.003458	0.004750	0.002583	0.000550	0.000314

Expected

x1	x2	x3	x4	x5
0.085347	0.008391	0.037693	0.008108	0.003462
0.008391	0.160773	0.051755	0.011086	0.004753
0.037693	0.051755	0.037418	0.006061	0.002535
0.008108	0.011086	0.006061	0.001690	0.000558
0.003462	0.004753	0.002535	0.000558	0.000315

Realized Norm: 0.2078243609286466
Expected Norm: 0.20834574581085424

– Test 5.4:

Realized

0	1	2	3	4
0.086017	0.000664	0.041080	0.008239	0.003523
0.000664	0.160498	0.056991	0.011750	0.005064
0.041080	0.056991	0.040026	0.007470	0.003269
0.008239	0.011750	0.007470	0.002552	0.000947
0.003523	0.005064	0.003269	0.000947	0.000499

Expected

x1	x2	x3	x4	x5
0.085347	0.012810	0.038889	0.008288	0.003568
0.012810	0.160737	0.053380	0.011333	0.004898
0.038889	0.053380	0.037418	0.006219	0.002681
0.008288	0.011333	0.006219	0.001690	0.000571
0.003568	0.004898	0.002681	0.000571	0.000315

Realized Norm: 0.2127487304774782
Expected Norm: 0.2100846981531551

– Test 5.5:

Realized

0	1	2	3	4
0.085161	0.117031	0.042192	0.008997	0.003879
0.117031	0.160828	0.057982	0.012363	0.005330
0.042192	0.057982	0.037297	0.006002	0.002576
0.008997	0.012363	0.006002	0.001096	0.000471
0.003879	0.005330	0.002576	0.000471	0.000203

Expected

x1	x2	x3	x4	x5
0.085227	0.117122	0.042212	0.009002	0.003881
0.117122	0.160953	0.058009	0.012371	0.005334
0.042212	0.058009	0.037190	0.005993	0.002572
0.009002	0.012371	0.005993	0.001095	0.000471
0.003881	0.005334	0.002572	0.000471	0.000203

Realized Norm: 0.2698650088630311
Expected Norm: 0.27004289099447176

- Section 6: Returns Calculator

  - Test 6.1: Arithmetic Returns
    The returns dataframe is too cumbersome to include (the output is available in the test_outputs folder should you choose to view it but the norms of the matrix prove the accuracy of my code):

    Realized Norm: 3.027524614084936
    Expected Norm: 3.02752461408493

  - Test 6.2: Log Returns

    Realized Norm: 3.020464728931918
    Expected Norm: 3.0204647289319126

- Section 7:

  - Test 7.1: Normal Distribution Fit

    Realized

mu	sigma
0.046026	0.046545

    Expected

mu	sigma
0.046026	0.046780

  - Test 7.2: T Distribution Fit

    Realized

mu	sigma	nu
0.045940	0.045443	6.336867

    Expected

mu	sigma	nu
0.045940	0.045443	6.336875

  - Test 7.3: T Regression

    Realized

mu	s	nu	betas
0.000518	0.048549	4.598590	0.042116
0.000518	0.048549	4.598590	0.974906
0.000518	0.048549	4.598590	2.041195
0.000518	0.048549	4.598590	3.154776

    Expected

mu	sigma	nu	Alpha	B1	B2	B3
0.000000	0.048548	4.598293	0.042634	0.974889	2.041192	3.154801

- Section 8: VaR (all absolute and alpha = 5%) and ES
  Results:

Table 1: Summary of VaR and ES Tests

Test	Realized	Expected
8.1: Normal VaR	0.030920	0.030920
8.2: T VaR	0.041530	0.041530
8.3: Historical VaR	0.045171	0.040212
8.4: Normal ES	0.049984	0.050468
8.5: T ES	0.075232	0.075232
8.6: Historical ES	0.081409	0.076906

- Section 9: VaR/ES using Copula
  After running the Copula VaR code and modeling security B using a T distribution and A as normally distributed (as well as $\alpha = 5\%$), the outputs were (please run tests.py to see these outputs):

  Total Portfolio VaR: 150.7413848187147
  Total Portfolio ES: 199.7851873376829

  The expected values were:

  Total Portfolio VaR: 152.5656843426060
  Total Portfolio ES: 199.7045322301790

Overall, my implemented risk management library is able to capably complete all the tasks required by all the tests demonstrated by the raw outputs and norms as a heuristic.

2 Problem 2

VaR is the minimum amount ($ or %) a portfolio is expected to lose on a day given that day is a $\alpha = \%$ "bad" day. After filtering the price data and implementing a "return_calculate" function in Python, we calculate arithmetic returns for the given price dataset, remove the mean, and then calculate VaR through various methods. For this problem, we will assume $\alpha = 5\%$. We report our VaR as an absolute percentage. Expected shortfall is our expected value (loss) conditional on the fact that the loss is beyond the VaR. For the EW variance normal distribution, we can use a closed form formula, for the T distribute we can use the well defined CDF, and for the Monte Carlo simulation we will calculate the average of the values below VaR.

- **Normal distribution with EW variance:**
  To evaluate VaR using the normal distribution, we calculate $-\Phi^{-1}(\alpha) = -1.645$ and do not factor in the. We then use the EW variance (using our EW function from previous homework with $\lambda = 0.97$) to calculate our VaR as follows (note that the mean of the returns is 0.08% which we subtract from the relative VaR: $\Phi^{-1}(\alpha)) * \sigma_{EW}$ to leave us with in absolute VaR):

  $$VaR = -(\Phi^{-1}(\alpha)) * \sigma_{EW} - \mu = 9.03\%$$

  In calculating ES, we use:

  $$ES_\alpha(X) = -\mu + \sigma \frac{f(\Phi^{-1}(\alpha))}{\alpha} = 11.41\%$$

- **MLE fitted T distribution:**
  Using scipy we fit a T distribution to the meta returns data with the parameters determined by MLE. Once that T distribution is fit ($df = 4.25, loc = -9.37e-05, scale = 0.0364$), we then take these parameters (and the subsequent t CDF function $F(x)$) to calculate the value at α percentile of that distribution to yield the VaR return as a percentage (note that we also subtract the mean from our relative VaR to yield absolute VaR but in this case the mean is negligible).

  $$-F_x^{-1}(\alpha) = VaR = 7.64\%$$

  Calculating ES, we use the following where $f(x)$ is the t-distribution density function

  $$ES(X) = -\frac{1}{\alpha} \int_{-\infty}^{-VaR(X)} x f(x)\, dx = 11.32\%$$

- **Historic simulation:**
  Sampling 10000 times with replacement from the given historical returns, we get the bootstrapped distribution of returns for our security of interest. Then evaluating the 5th percentile of those samples using the np.percentile function, we achieve the minimum percentage loss on the worst 5% of days which equals 7.59%.
  In calculating ES, we take the average return of all the sampled returns below VaR (7.59%). This yields:

  $$VaR = 7.59\% \quad ES = 11.66\%$$

In summary:

Method	Calculated VaR ($)	Calculated ES
Using a normal distribution (EW variance $\lambda = 0.97$)	9.03%	11.41%
Using a MLE fitted T distribution.	7.64%	11.32%
Using a Historic Simulation.	7.59%	11.66%

In theory, the exponential weighted VaR being higher could indicate more volatility in recent markets for the security in question relative to the unweighted methods (T and historic). It is interesting that the normal distribution has such a higher VaR value but still a comparable expected shortfall indicating a relatively higher probability of returns closer to the VaR value (relatively thinner tails and skewed to the right extreme probability distribution relative to the T and historical distributions meaning given recent volatility there could be some underestimation in ES, ie ES would be higher if the VaR values were 9% for the T and historic distributions).

3 Problem 3

The high-level steps of the algorithm for evaluating porfolio VaR as well as ES (considering securities in portfolios A and B to be T distributed and securities in portfolio C to be normally distributed and using a copula) I implemented are below (where $\alpha = 0.05$), please see the full algorithms in my repo:

- Create most recent price dataframe using the returns_calculate function and create new dataset consisting of each ticker and its most recent close price.
- Join most recent close data frame to portfolio data frame (so that we have quantity and price in the portfolio dataset to calculate the value of each holding.
- Fit T distributions to the securities in portfolios A and B and normal distributions to the securities in C (when analysing each individual portfolio separately, fit the requested distribution).
- Using the fitted T and normal CDF functions, transform the returns into a uniform vector by applying the fit CDF to the observed values.
- Then standardize these uniform values using the inverse normal CDF function and applying that function to the uniform values.
- Use the Spearman correlation function to calculate the correlation matrix of all of the standard normal returns values for each security (over the whole observation period).
- Simulate 5000 draws from the multivariate normal distribution given the Spearman calculated correlation matrix.
- Run these simulated multivariate normal values through the normal cdf to transform the observations back into the uniform variable preserving the correlations between distributions/variables.
- Reapply the CDF functions fit for each security at the beginning of the process to transform the uniform values into simulated returns (5000 simulations for all securities)
- After the copula has been fit and simulated from, we then have 5000 simulated returns for each security which reflect the overall correlation between each security, we can then combine these simulated values with our portfolio statistics joined by security (so that for every portfolio we have every security return simulated 5000 times as well as the original number of shares held and share price). Given we have original share price, simulated return, and number of shares held, we can then calculate the current holding value and simulated holding value and pnl (simulated - current) for each simulation.
- Given we now have each simulated position's holding value, we then sum up the holding value for each position grouped by simulation to calculate each portfolio's total value by simulation (essentially summing the parts of each portfolio on a simulation by simulation basis resulting in a Monte Carlo Simulation of 5000 trials for the total portfolio value). When analyzing all portfolios, we simply sum up each portfolio (A,B,C) and its total value to get the simulated total portfolio value. We save the pnl of each simulated portfolio total value and then analyze the pnl for VaR and ES.
- To calculate VaR we take the 5th percentile of the simulated pnl distribution and for ES we then take the average of each pnl simulated with value less than the VaR.

The rigorous implementation can be found in the q3, q3a, q3b, and q3c python scripts in the github.

The algorithm was run on all of the portfolios (A,B,C) as well as the overall portfolio

Portfolio	Calculated VaR (pnl $)	Expected Shortfall (pnl $)
A	7,171.29	9,901.99
B	6951.70	9,211.27
C	5,835.78	7,270.86
Total	20,760.44	26,300.89

Table 2: Portfolio Value at Risk (VaR) and Expected Shortfall

The holding values (number of securities owned by each portfolio) differ from this week to last, thus, any comparison between the two weeks will be effectively meaningless. One would imagine this way of approaching VaR and ES better preserves the correlations between securities and obviously is able to model each security to a more granular level than the delta normal approach.