Homework 5

Wednesday, 28 February 2024 11:13 PM

Problem 1

All the calculation function is in the quant_risk_library.py. By loading the data and importing function from this library in question_1.py, I'm able to compare my calculation outcome with the expected outcome.
To avoid floating error, I set tolerance to 1e-3 for all percentage/floating result. The function of comparing calculated result and expected result is as below:

```
def testfiles_equal(df1, df2):
    if df1.equals(df2):
        return True
    else:
        # If they are not equal, it could be due to floating point precision,
        # so we attempt a comparison that allows for a small absolute difference.
        try:
            pd.testing.assert_frame_equal(df1, df2, atol=1e-3)  # Use an appropriate tolerance
            return True
        except AssertionError:
            return False
```

And the calculation result is as below:

```
Test 1.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 1.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 1.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 1.4 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 2.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 2.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 2.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 3.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 3.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 3.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 3.4 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 4.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 5.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 5.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 5.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 5.4 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 5.5 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 6.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 6.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 7.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 7.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 7.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
```

```
Test 8.1 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 8.2 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 8.3 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 8.4 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 8.5 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 8.6 calculated Value and Expected value is indifferent with 1e-3 tolerence: True
Test 9.1 calculated Value and Expected value is indifferent with 3 tolerence: True
```

Problem 2

	VaR(percentage)	ES(percentage)
Normal distribution with EWA	0.0877	0.1094
T distribution	0.0765	0.1132
Historical simulation	0.0760	0.1168

For the 3 distribution mentioned above, we can see that ES is larger than VaR for all distribution. This means that for all three distributions, the tail is fatter. And due to the fact that VaR can only capture one single point at 5% and is not good at capturing fat tails, it tends to underestimate the risk comparing to ES.

The code calculation result is attached below:

```
VaR for normal distribution with exponential weighted average is: 0.08768106793598726
VaR for T distribution is: 0.07647602684516216
VaR for historical simulation is: 0.07598069069686242
ES for normal distribution with exponential weighted average is: 0.10940589900776566
ES for T distribution is: 0.11321790226657903
ES for historical simulation is: 0.11677669788562187
```

Problem 3

Question_3_3.py is the final code for the below calculation. Generally there are few steps to calculate VaR and ES using copulas.

Step1: calculate portfolio return for portfolio a,b,c and total. Adjuest the return matrix by removing the mean.
Step2: construct correlation matrix between portfolio returns.
Step3: use the fitted parameters based on portfolio distribution and correlation matrix calculated above to construct copula simulation
Step4: use the copula simulated returns to calculate historical VaR and ES same as last week method.

Using the portfolio data provided in week5, the 95% VaR and ES is calculated as below

	Portfolio	VaR95	ES95	VaR95_Pct	ES95_Pct
0	A	8199.612139	10547.674401	0.026440	0.034012
1	B	6455.046474	8445.105808	0.021787	0.028504
2	C	5775.415687	7234.877141	0.020707	0.025939
3	Total	22794.439830	30399.670040	0.025747	0.034338

Using the portfolio data provided in week4(for comparison), the VaR and ES is calculated as below:

	Portfolio	VaR95	ES95	VaR95_Pct	ES95_Pct
0	A	14735.000073	20520.273714	0.017044	0.023736
1	B	10167.173855	13588.430046	0.019398	0.025925
2	C	18994.703449	23805.454421	0.017334	0.021724
3	Total	51043.163337	69388.419488	0.020545	0.027929

Week04 dollar VaR for EWV is:

To sum up, the 5% dollar VaR for portfolios are listed as
A: $15284.38
B: $7786.28
C: $17826.14
Total: $38125.12

From the data above, we can see that the VaR computed using copula is generally larger than that of EWV except portfolio A.