Fintech 545 Final Exam

NetID: wz172

Name: Wenyue Zhao

Hashed Name: c9f2b0dcaddaa9901ff35ecdc3b99c70cfa940ac

Problem1

Answer1

Using the data in “problem1.csv”

1.Calculate log returns

```
julia> returns = return_calculate(prices,method="LOG",dateColumn="Date")
19×4 DataFrame
 Row │ Date        Price1           Price2           Price3         …
     │ Date        Float64?         Float64?         Float64?       …
─────┼──────────────────────────────────────────────────────────────────
   1 │ 2023-04-13      -0.011375       -0.00781393      -0.00885814  …
   2 │ 2023-04-14      -0.00144296     -0.00205868       0.000724714
   3 │ 2023-04-15       0.00603908      0.0062936        0.00582721
   4 │ 2023-04-16       0.000941397     0.000130434     -0.00119776
   5 │ 2023-04-17      -0.00147132     -0.00306872   missing         …
   6 │ 2023-04-18       0.00547465      0.00543251   missing
   7 │ 2023-04-19      -0.0100864      -0.00673178      -0.00500889
   8 │ 2023-04-20       0.0058507       0.00425145       0.00597521
   9 │ 2023-04-21       0.00316581      0.00179861      -0.000211902 …
  10 │ 2023-04-22  missing          missing             -0.00612103
  11 │ 2023-04-23  missing          missing              0.00599717
  12 │ 2023-04-24      -0.00657676     -0.00673489      -0.00625978
  13 │ 2023-04-25      -0.0041556      -0.00168234      -0.00153254  …
  14 │ 2023-04-26       0.0160147       0.0068833        0.000705494
  15 │ 2023-04-27      -0.00796495     -0.00249421      -0.000138515
  16 │ 2023-04-28       0.00206413     -0.000336567      7.06645e-5
  17 │ 2023-04-29       0.00167294      0.00410485       0.00443365  …
  18 │ 2023-04-30  missing              0.004058         0.00685603
  19 │ 2023-05-01  missing          missing          missing
```

2.Calculate Pairwise Covariance

```
julia> col = [:Price1, :Price2, :Price3]
3-element Vector{Symbol}:
 :Price1
 :Price2
 :Price3

julia> x = Matrix(prices[!,col])
20×3 Matrix{Union{Missing, Float64}}:
 90.1722    109.669     86.1736
 89.1523    108.816     85.4136
 89.0237    108.592     85.4755
 89.563     109.277     85.9751
 89.6473    109.292     85.8721
 89.5155    108.957       missing
 90.0069    109.55      85.9043
 89.1037    108.815     85.4751
 89.6265    109.279     85.9873
 89.9107    109.476     85.9691
   missing     missing  85.4445
 89.783     109.353     85.9585
 89.1944    108.619     85.4221
 88.8245    108.436     85.2912
 90.2585    109.185     85.3514
 89.5424    108.913     85.3396
 89.7275    108.877     85.3456
 89.8777    109.324     85.7249
   missing  109.769     86.3146
 88.969        missing    missing

julia> pairwise = missing_cov(x,skipMiss=false,fun=cov)
3×3 Matrix{Float64}:
 0.180147   0.128689  0.0690949
 0.128689   0.148581  0.116019
 0.0690949  0.116019  0.10821
```

3.Is this Matrix PSD? If not, fix it with the “near_psd” method

```
julia> eVal = eigvals(pairwise)
3-element Vector{Float64}:
 0.0012355406048242
 0.0748303352882249
 0.3608719839229988
```

This Matrix is PSD, because all the eigen values are significantly larger than 0, or larger than -1e-7. No need to fix it.

4.Discuss when you might see data like this in the real world.

This data is likely to be seen in the stock market, or the financial market. The are likely to be the daily prices of financial instruments, it goes up and down like "random walk". And the value of log returns of the given data seem to be real in the market.

There are missing data in the prices and it is common in the financial world. Because not all markets are open at the same time on the same days. A holiday in one market is not necessarily a holiday in another, even in the same country. The three data might be instruments with different trading schedule. Here, we use pairwise method to calculate the covariance of the prices. We find the matching rows for each pair, and build the covariance matrix piece by piece. Though this method will not guarantee the covariance matrix be a "PSD" matrix, we checked it above, and luckily, it is a PSD matrix here.

Problem2

“problem2.csv” contains data about a call option. Time to maturity is given in days. Assume 255 days in a year.

Answer2

1.Calculate the call price

```
julia> option_data = CSV.read("C:/Users/17337/Desktop/FinTech-545-Spring2023
-main/Final/problem2.csv",DataFrame)
1×6 DataFrame
 Row │ Underlying  Strike   IV       TTM    RF       DivRate
     │ Float64     Float64  Float64  Int64  Float64  Float64
─────┼──────────────────────────────────────────────────────
   1 │    109.536  98.5297     0.23    151    0.045  0.0470394
```

```
julia> val = gbsm(true,S,strike,ttm,rf,b,ivol,includeGreeks=true)
GBSM(13.641968027031453, 0.7310987190572306, 0.015881737975622263, 25.95248727198594,

julia> println("Value: $(val.value)")
Value: 13.641968027031453

julia> println("Delta: $(val.delta)")
Delta: 0.7310987190572306

julia> println("Gamma: $(val.gamma)")
Gamma: 0.015881737975622263

julia> println("Vega: $(val.vega)")
Vega: 25.95248727198594

julia> println("Theta: $(val.theta)")
Theta: -4.262906824862621

julia> println("Rho: $(val.rho)")
Rho: 39.34284345364541
```

call price is 13.641968027031453

2.Calculate Delta

0.7310987190572306

3.Calculate Gamma

0.015881737975622263

4.Calculate Vega

25.95248727198594

5.Calculate Rho

39.34284345364541

6.Calculate VaR at 5%

```
julia> S0 = S
109.5363281813056

julia> for j in 1:1000
           d = Normal(0,ivol/sqrt(255))
           r = rand(d,1000)
           S = S0 .* (r .+ 1)
           p =  [gbsm(true,S[i],strike,150/255,rf,b,ivol).value for i in 1:1000]
           pnl = (S .- S0) - (p .- val.value)

           vars[j] = VaR(pnl)
           ess[j] = ES(pnl)
       end

julia> println("VaR Range: $(mean(vars)) 95% confidence [$(quantile(vars,.025)), $(quantile(vars,.975))]")
VaR Range: 0.7343787075732929 95% confidence [0.6734576587547085, 0.7938533529944296]

julia> println("ES Range : $(mean(ess)) 95% confidence [$(quantile(ess,.025)), $(quantile(ess,.975))]")
ES Range : 0.943705953610806 95% confidence [0.8697672793087581, 1.0176096023605479]
```

VaR Range: 0.7343787075732929 95% confidence [0.6734576587547085, 0.7938533529944296]

7.Calculate ES at 5%

ES Range : 0.943705953610806 95% confidence [0.8697672793087581, 1.0176096023605479]

8.This portfolio’s payoff structure most closely resembles what?

This portfolio’s payoff structure most closely resembles CoveredCall. CoveredCall is one long position of stock and one short position of call, which replicates(behaves like) a put option.

Problem3

Data in “problem3_cov.csv” is the covariance for 3 assets. “problem3_ER.csv” is the expected return for each asset as well as the risk free rate

Answer3

1.Calculate the Maximum Sharpe Ratio Portfolio

```
julia> wop = round.(value.(w),digits=4)
3-element Vector{Float64}:
 0.3292
 0.32
 0.3509
```

asset1:0.3292 asset2: 0.32 asset3: 0.3509

2.Calculate the Risk Parity Portfolio

```
julia> wrp = round.(value.(w),digits=4)
3-element Vector{Float64}:
 0.3292
 0.32
 0.3509
```

3.Compare the differences between the portfolio and explain why

```
julia> println("ER Optimal: $(wop'*er)")
ER Optimal: 0.12809674489104267

julia> println("SD Optimal: $(sqrt(wop'*covar*wop))")
SD Optimal: 0.16259668065295393

julia> println("SR Optimal: $((wop'*er - rf) / sqrt(wop'*covar*wop))")
SR Optimal: 0.5110605244666969

julia> println(" ")


julia> println("ER RP: $(wrp'*er)")
ER RP: 0.12809674489104267

julia> println("SD RP: $(sqrt(wrp'*covar*wrp))")
SD RP: 0.16259668065295393

julia> println("SR Optimal: $((wrp'*er - rf) / sqrt(wrp'*covar*wrp))")
SR Optimal: 0.5110605244666969
```

These two portfolios are equal, with same weights, expected returns, maximum sharpe, and volatility.

This is because the sharpe ratio for each asset is the same and the correlation between each asset is the same. Here they are.

```
julia> corr = cov2cor(covar, vols)
3×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
 1.0  1.0  1.0
```

```
julia> (0.12913882911464453-0.045)/sqrt(0.04568741078765268)
0.39363906439194135

julia> (0.13155535179860567-0.045)/sqrt(0.04834944241137626)
0.3936390643919413

julia> (0.12392855449115958-0.045)/sqrt(0.04020424552161158)
0.39363906439194135
```

Correlations and Sharpe ratios are equal -> risk parity is the maximum sharpe ratio portfolio. These two portfolios have same weights.

Problem 4

Data in “problem4_returns.csv” is a series of returns for 3 assets. “problem4_startWeight.csv” is the starting weights of a portfolio of these assets as of the first day in the return series.

Answer4

1.Calculate the new weights for the start of each time period

```
julia> lastW
3-element Vector{Float64}:
 0.14205200994273412
 0.17159569996180485
 0.6863522900954612
```

The updated weights for the three assets are: asset1 0.14, asset2 0.17, asset3 0.69

2.Calculate the ex-post return attribution of the portfolio on each asset

3.Calculate the ex-post risk attribution of the portfolio on each asset

Question 2 and 3 see the Attribution dataframe below.

```
julia> println(Attribution)
3×5 DataFrame
 Row │ Value              Asset1       Asset2       Asset3     Portfolio
     │ String             Float64      Float64      Float64    Float64
─────┼───────────────────────────────────────────────────────────────────
   1 │ TotalReturn        -0.434708    -0.153261    0.377433   0.0501684
   2 │ Return Attribution -0.121462    -0.0314611   0.203092   0.0501684
   3 │ Vol Attribution     0.00159946  -0.00132575  0.0336552  0.0339289
```

Extra Credit

Input prices in “problem5.csv” are for a portfolio. You hold 1 share of each asset. Using arithmetic returns, fit a generalized T distribution to each asset return series. Using a Gaussian Copula:

Answer extra credit

1.Calculate VaR (5%) for each asset

99.81310593244524, 97.61960896020994, 99.43383205301558, 117.05357034682892

2.Calculate VaR (5%) for a portfolio of Asset 1 &2 and a portfolio of Asset 3&4

VaR (5%) for a portfolio of Asset 1 &2
196.8581380067783

VaR (5%) for a portfolio of Asset 3 &4
197.6596579311149

3.Calculate VaR (5%) for a portfolio of all 4 assets.

VaR (5%) for a portfolio of all 4 assets
399.6634682466357

```
VaR (5%) for each asset
[99.29863581905839, 97.09292412393731, 96.65794999303886, 100.735925433756]
VaR (5%) for a portfolio of Asset 1 &2
196.8581380067783
VaR (5%) for a portfolio of Asset 3 &4
197.6596579311149
VaR (5%) for a portfolio of all 4 assets
399.6634682466357
```