MF796 Homework4
Xuyang Liu
xyangliu@bu.edu

Problem Set # 4

Problem 1: Covariance Matrix Decomposition:

(a) I choose randomly 100 stocks' five year daily close price from S&P 500. And here's a brief view of these data:

```
In [3]: adj.head()
Out[3]:
                    C        CAG          CB  ...         STZ        XOM
XRAY
Date                                          ...
2018-03-12  67.195992  33.805389  130.919647  ...  216.828690  60.095406
56.582001
2018-03-13  66.214821  33.787621  129.867462  ...  214.746628  59.528336
55.981510
2018-03-14  64.941986  33.281208  128.787796  ...  214.228424  58.785522
55.497242
2018-03-15  64.827065  32.321690  129.062347  ...  213.757370  59.440456
55.158260
2018-03-16  64.941986  32.526031  128.989105  ...  215.311859  59.999565
54.557762

[5 rows x 100 columns]
```

(b) I compute the log return and get the results:

```
In [4]: returns
Out[4]:
                   C       CAG        CB  ...       STZ       XOM      XRAY
Date                                      ...
2018-03-13 -0.014709 -0.000526 -0.008069  ... -0.009649 -0.009481 -0.010669
2018-03-14 -0.019410 -0.015102 -0.008348  ... -0.002416 -0.012557 -0.008688
2018-03-15 -0.001771 -0.029254  0.002130  ... -0.002201  0.011079 -0.006127
2018-03-16  0.001771  0.006302 -0.000568  ...  0.007246  0.009362 -0.010947
2018-03-19 -0.010536 -0.007402 -0.004835  ... -0.009629 -0.012997 -0.015565
...              ...       ...       ...  ...       ...       ...       ...
2022-03-07 -0.018549 -0.020559 -0.032641  ... -0.021736  0.035399 -0.020564
2022-03-08 -0.012317 -0.085771 -0.006405  ...  0.007764  0.007547 -0.020797
2022-03-09  0.019492  0.008050  0.031328  ...  0.000373 -0.058526  0.017101
2022-03-10 -0.019309 -0.016492 -0.012434  ... -0.005043  0.030570 -0.012354
2022-03-11  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000

[1008 rows x 100 columns]
```

(c) Here's the covariance matrix:

```
In [5]: covmat
Out[5]:
             C       CAG            CB  ...       STZ       XOM      XRAY
C     0.000646  0.000087  3.218911e-04  ...  0.000254  0.000372  0.000298
CAG   0.000087  0.000391  8.952491e-05  ...  0.000104  0.000078  0.000069
CB    0.000322  0.000090  3.557998e-04  ...  0.000176  0.000222  0.000200
CCI   0.000153  0.000097  1.465678e-04  ...  0.000133  0.000099  0.000106
CEG  -0.000004 -0.000003  9.955416e-07  ...  0.000002  0.000004 -0.000004
...        ...       ...           ...  ...       ...       ...       ...
RE    0.000304  0.000083  2.772352e-04  ...  0.000187  0.000209  0.000196
SCHW  0.000416  0.000068  2.470547e-04  ...  0.000186  0.000287  0.000222
STZ   0.000254  0.000104  1.757017e-04  ...  0.000381  0.000188  0.000169
XOM   0.000372  0.000078  2.220414e-04  ...  0.000188  0.000456  0.000207
XRAY  0.000298  0.000069  2.001190e-04  ...  0.000169  0.000207  0.000482
```

And test the negative or positive characteristic of the eigenvalues:

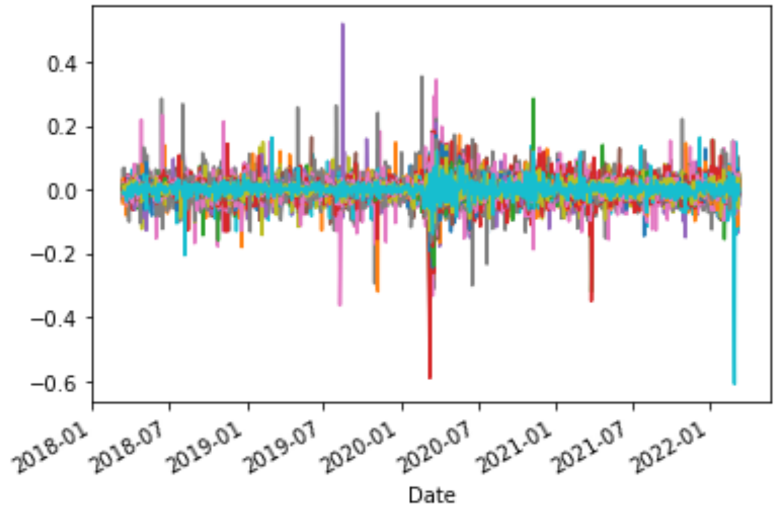
```
In [8]: sum(egv<=0)
Out[8]: 0
```

I find that all the eigenvalues are **positive**. That's reasonable. If there're any negative, maybe my datasets have some errors or the data itself has a strong autocorrelation.

(d) The first **2** eigen values account for 50% of the variance. And **43** eigenvalues account for 90% in my datasets.

It makes sense because according to CAPM model, the market risk would account for returns. Although there're some other factors, they're not majority. That's why our first couple of eigenvalues would have larger proportion.

(e) Here's the required plot:



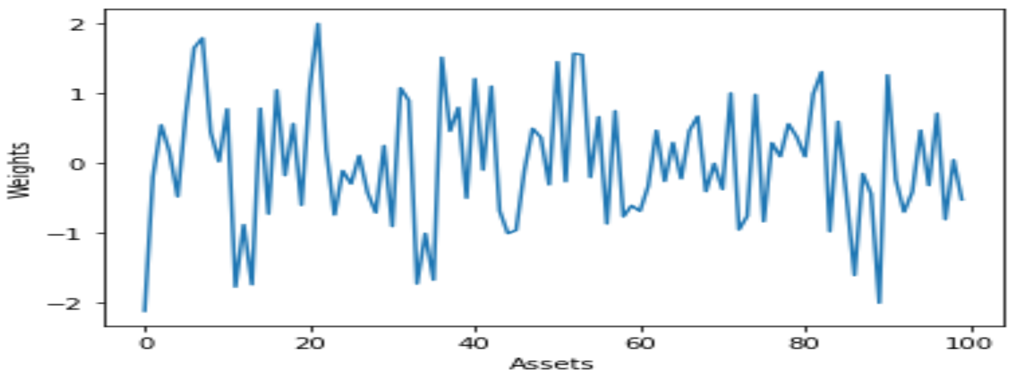
From the plot we can figure out that the residual is around 0 with a normal distribution.

Problem 2: Portfolio Construction:

(a) I just use the numpy package, the function np.linalg.inv () to invert this in a stable way. And the invert result is here:

```
array([[ 2.99274793e-05, -8.35501311e-06],
       [-8.35501311e-06,  1.14185212e-05]])
```

(b) I suppose the risk aversion is about 1. And I got the result matrix here, to make is more visible, I also plot it.



Unluckily, there're lots of negative position, which means we must short in daily practice. However, it's unacceptable in mutual funds. I think we should add a constrain such as making sure all the weight is above zero.

MF796 Homework4
Xuyang Liu
xyangliu@bu.edu

Problem 3: Portfolio Stability:

(a) Here I guess that the expected return equals the mean of their historical returns. Then I use scipy.minimize to get the result:

```
array([1.63384228e-20, 0.00000000e+00, 6.90550754e-01, 1.45184875e-01,
       1.36989349e-20, 0.00000000e+00, 1.64264372e-01, 6.59031346e-21,
       1.53750198e-20, 0.00000000e+00])
```

Some of weights are too light so we may ignore them. That is, Sec3: 69.1%, Sec4:14.5%, Sec7:16.4%

(b) Using the same method, I got the result:

```
array([0.00000000e+00, 0.00000000e+00, 3.39515162e-02, 9.55451449e-20,
       1.20627405e-20, 0.00000000e+00, 0.00000000e+00, 9.66048484e-01,
       1.16825595e-18, 0.00000000e+00])
```

That is, Sec3:3.40%, Sec8: 96.6%

(c) Using the same method, I got the result:

```
array([1.25756856e-22, 1.02506072e-18, 1.88830576e-21, 3.58782559e-22,
       2.66451836e-19, 2.66451836e-19, 4.83080512e-19, 1.00000000e+00,
       4.63814906e-24, 2.63407389e-19])
```

That is, invest all the money on Sec8

(d) It would be stable when I change add or minus the same values on all the sec. From the picture below, we can tell that the majority of weights are stable.

```
In [195]: optw
Out[195]:
array([0.00000000e+00, 0.00000000e+00, 3.39515162e-02, 9.55451449e-20,
       1.20627405e-20, 0.00000000e+00, 0.00000000e+00, 9.66048484e-01,
       1.16825595e-18, 0.00000000e+00])

In [196]: optw2
Out[196]:
array([6.82261791e-17, 1.70996270e-16, 3.39514431e-02, 5.61829403e-17,
       3.81650636e-17, 4.50107853e-17, 2.71841504e-16, 9.66048557e-01,
       8.04310934e-18, 3.08386776e-16])

In [197]: optw3
Out[197]:
array([0.00000000e+00, 4.57710084e-19, 3.39514930e-02, 1.02506766e-18,
       0.00000000e+00, 3.11317546e-19, 0.00000000e+00, 9.66048507e-01,
       2.02343928e-19, 2.84792813e-19])
```

If we just change one of them, the result still reveals stable.

```
In [209]: optw
Out[209]:
array([0.00000000e+00, 0.00000000e+00, 3.39515162e-02, 9.55451449e-20,
       1.20627405e-20, 0.00000000e+00, 0.00000000e+00, 9.66048484e-01,
       1.16825595e-18, 0.00000000e+00])

In [210]: optw4
Out[210]:
array([0.00000000e+00, 6.21358454e-19, 3.39515587e-02, 0.00000000e+00,
       3.26040239e-19, 3.80945685e-19, 0.00000000e+00, 9.66048441e-01,
       0.00000000e+00, 1.90514968e-19])
```