QF600 HOMEWORK4 WANGHAOTONG

Q1.a.Estimate the expected deviation from market return, for the ten industry portfolios.

	Expected deviation from market return
NoDur	0.154750
Durbl	-0.014750
Manuf	0.264750
Enrgy	0.483083
HiTec	0.018167
Telcm	0.133333
Shops	0.168250
Hlth	0.035750
Utils	0.159083
Other	-0.259000

Q1.b Estimate the covariance matrix of return deviations.

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Shops	Hlth	Utils	Other
NoDur	5.439696	-6.073035	-1.396192	-1.200533	-1.883151	1.538885	1.140741	3.815137	4.272002	-1.768738
Durbl	-6.073035	26.628901	4.908024	-3.481055	1.891577	-1.707625	-0.354335	-8.082946	-9.617490	4.385865
Manuf	-1.396192	4.908024	2.950499	1.666133	0.065267	-0.626416	-1.154597	-2.288900	-1.901412	0.358904
Enrgy	-1.200533	-3.481055	1.666133	19.274911	-1.516972	-1.040525	-3.710439	-2.485796	4.454368	-3.864826
HiTec	-1.883151	1.891577	0.065267	-1.516972	5.098746	-0.773294	-0.245350	-1.936284	-2.342839	-1.404050
Telcm	1.538885	-1.707625	-0.626416	-1.040525	-0.773294	4.682567	0.463797	0.693157	2.721477	-1.271778
Shops	1.140741	-0.354335	-1.154597	-3.710439	-0.245350	0.463797	4.452628	0.764510	-0.176666	-0.256987
Hlth	3.815137	-8.082946	-2.288900	-2.485796	-1.936284	0.693157	0.764510	7.820446	3.496136	-1.726842
Utils	4.272002	-9.617490	-1.901412	4.454368	-2.342839	2.721477	-0.176666	3.496136	12.267476	-4.055112
Other	-1.768738	4.385865	0.358904	-3.864826	-1.404050	-1.271778	-0.256987	-1.726842	-4.055112	4.503204

Q1.c Plot the minimum-tracking-error frontier generated by the ten industry portfolios.

First,I calculate the zeta,alpha,delta and Rmv follow the following code

```
zeta = tran_mean@inverse_cov@mean_array
zeta
```

0.20474497351130078

```
alepha = tran_mean@inverse_cov@identity_matrix
alepha
```

2.932127882630633

```
delta = tran_identity@inverse_cov@identity_matrix
delta
```

58.55025437639917

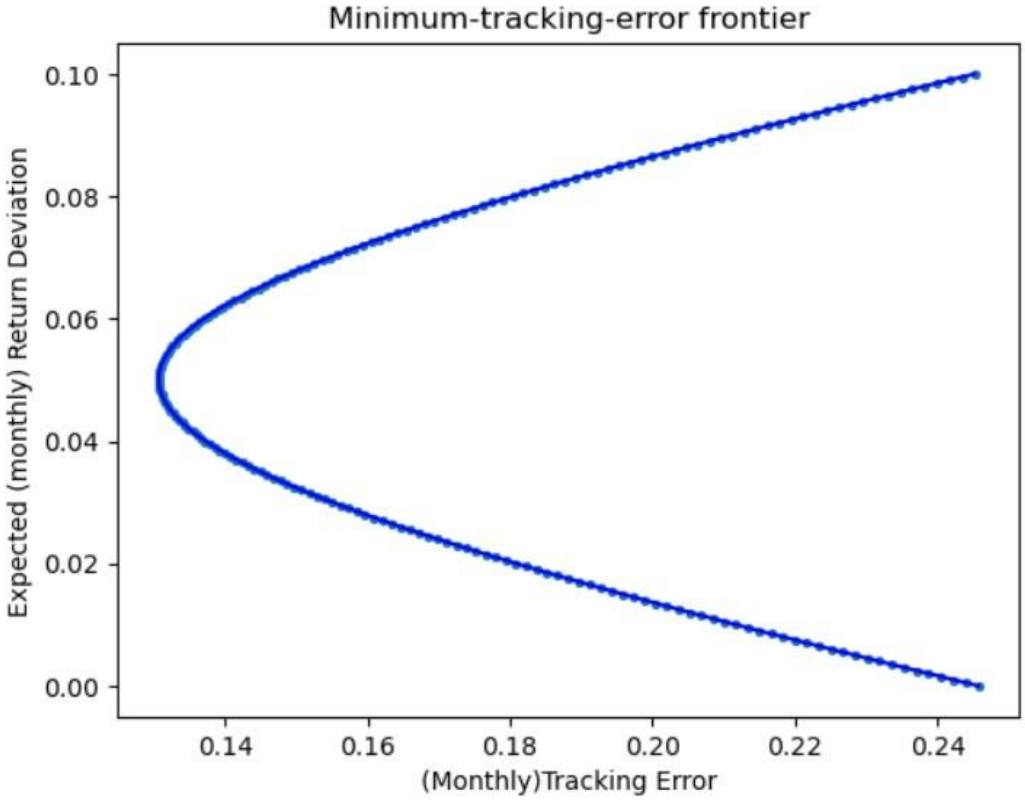
```
Rmv = alepha/delta
Rmv
```

0.050078823975400776

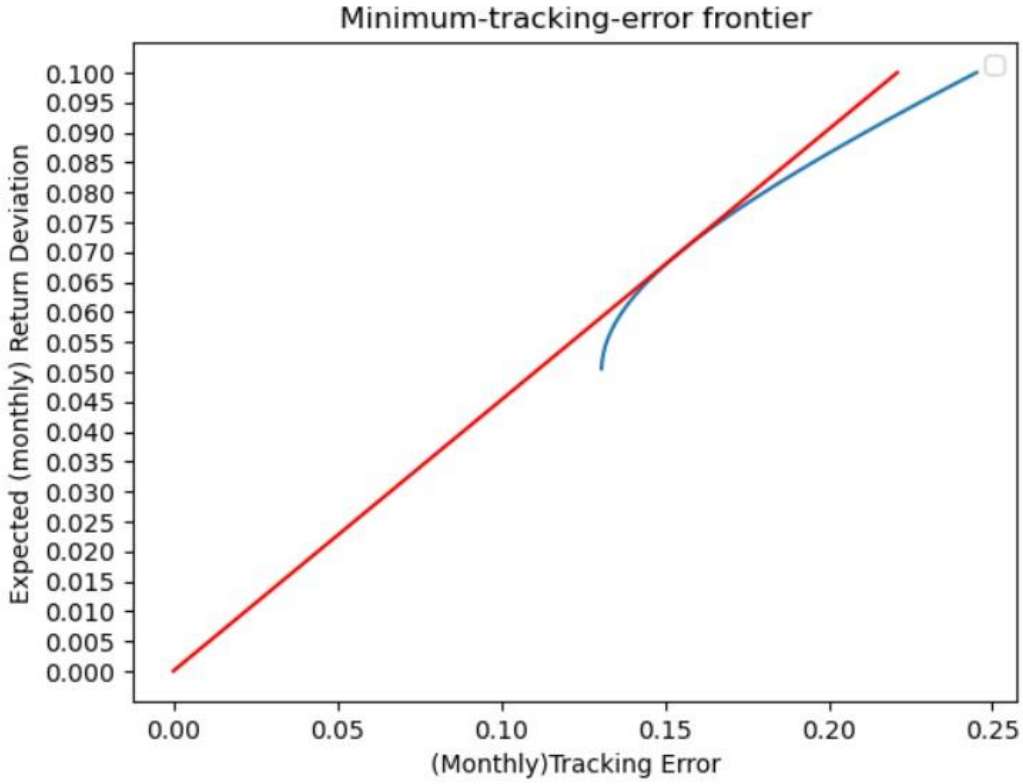
And then I defined a function to generate the tracking error

```
def variance_computing(item):
    return 1/delta+[delta/(zeta*delta-alepha*alepha)]*(item-Rmv)*(item-Rmv)
```

And then plot the minimum-tracking-error frontier generated by the ten industry portfolios. The graph has shown below.



Q1.d Plot the line starting from the origin that is tangent to the upper half of the minimum-tracking-error frontier.



Q1.e Calculate the information ratio and portfolio weights for the "tangency" portfolio.

I calculate the information ratio follow the following code.

```
information_ratio = linregress(sqrt_result1, rp)[0]
information_ratio
```

```
0.45248753961993377
```

The information ratio = 0.4525

And the i calculate the weight follow the equation below

```
weight = (delta*rtg-alepha)/(zeta*delta-alepha**2)*inverse_cov@mean_array+(zeta-alepha*rtg)/(zeta*delta-alepha**2)
*inverse_cov@identity_matrix
```

And finally the weight for the ten portfolios

	Weight
NoDur	0.052634
Durbl	0.000153
Manuf	0.137627
Enrgy	0.087032
HiTec	0.179353
Telcm	0.071074
Shops	0.106884
Hlth	0.102776
Utils	0.040162
Other	0.222304

Q2.a Plot the data points with mean return on the vertical axis vs standard deviation of return on the horizontal axis.

First I write a for loop as follows

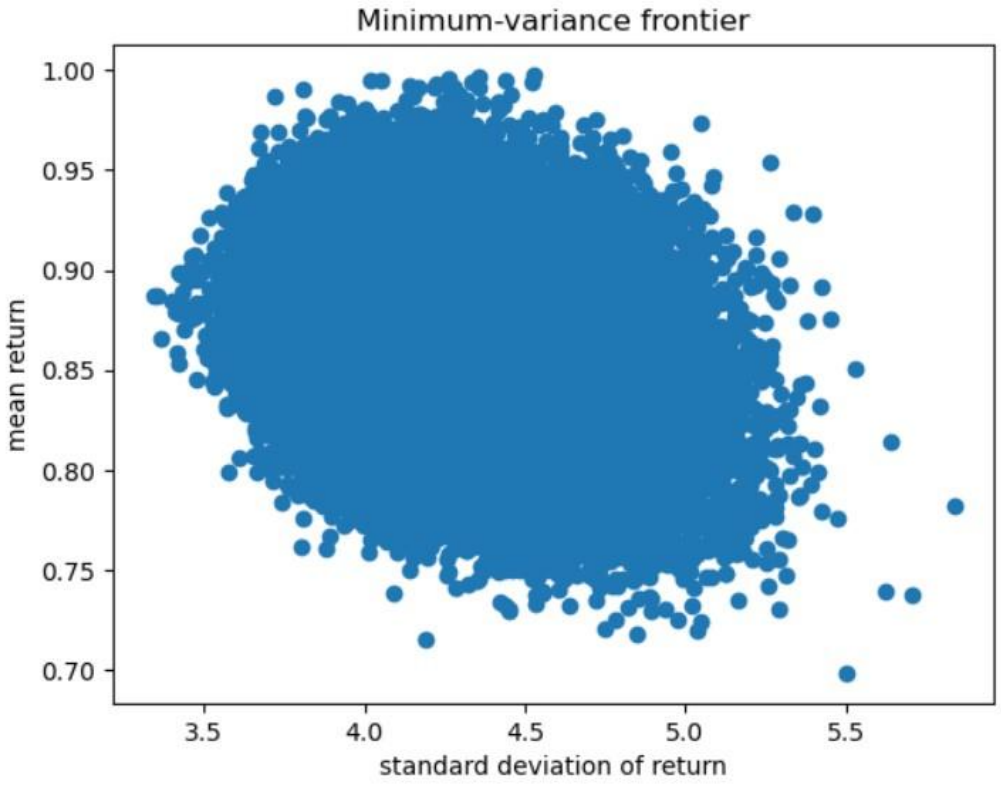
```
i = 0
for i in range(100000):
    w = np.random.uniform(0, 1, 10)
    weight = w/w.sum()
    mean = weight@R
    variance = weight.T@V@weight
    std = np.sqrt(variance)
    df.iloc[i]=[mean, std]
    i +=1
```

and I get a table which contains the mean and the standard deviation

	mean	std_dev
0	0.83651	4.647248
1	0.863407	4.438094
2	0.839684	4.405279
3	0.821917	4.505118
4	0.888967	4.597201
...	...	...
99995	0.821194	4.285456
99996	0.835056	4.427498
99997	0.817492	4.545744
99998	0.856945	4.473665
99999	0.926587	4.126091

100000 rows × 2 columns

And finally I plot the data points with mean return on the vertical axis vs standard deviation of return on the horizontal axis.



Q2.b Plot the new data points (on a separate graph) with mean return on the vertical axis vs standard deviation of return on the horizontal axis.

I rewrite the for loop as below

```
i = 0
for i in range(100000):
    inverse_w = np.random.uniform(0,1,10)
    w = 1/inverse_w
    weight = w/w.sum()
    mean = weight @ R
    variance = weight.T @ V @ weight
    std_dev = np.sqrt(variance)
    df_2.iloc[i]=[mean,std_dev]
    i +=1
```

And then get the table contains mean and standard variation

	mean	std_dev
0	1.201501	5.799654
1	0.871569	4.249907
2	0.813338	4.323019
3	0.743006	4.854808
4	0.733011	8.196149
...	...	...
99995	0.851838	4.279513
99996	0.779474	3.75758
99997	0.937132	3.938221
99998	0.889491	4.002769
99999	0.916539	3.848326

100000 rows × 2 columns

And finally plot the new data points (on a separate graph) with mean return on the vertical axis vs standard deviation of return on the horizontal axis.