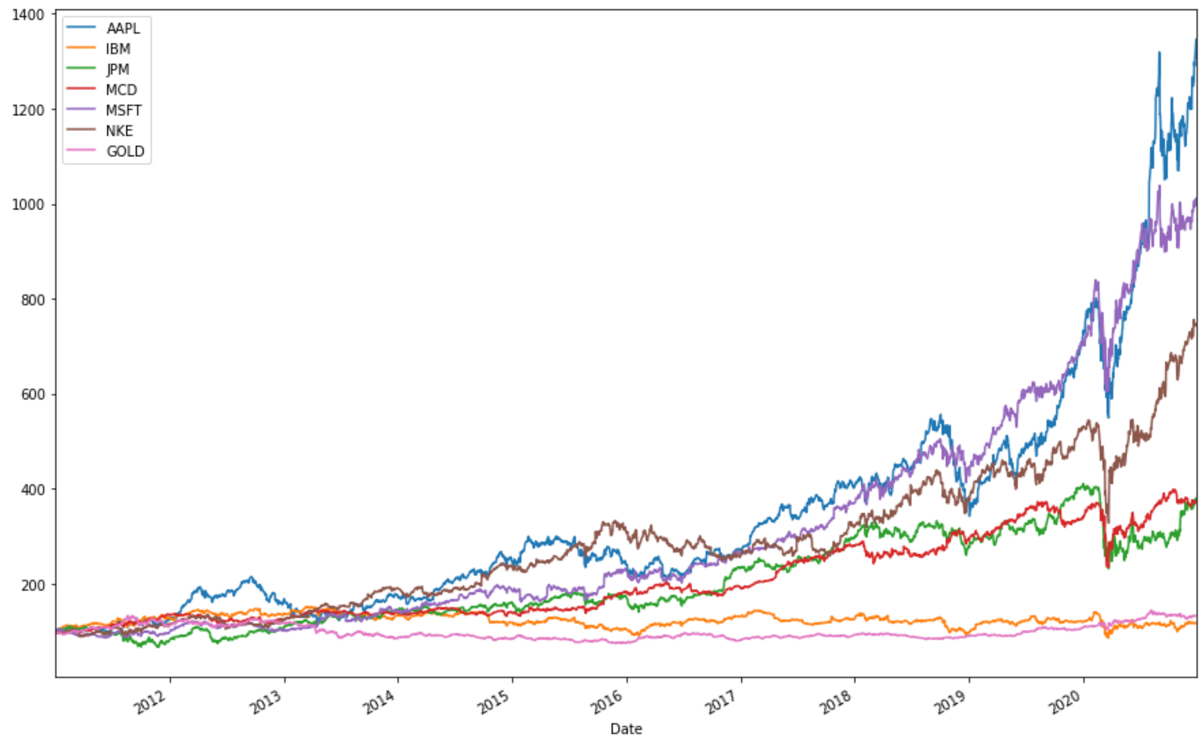
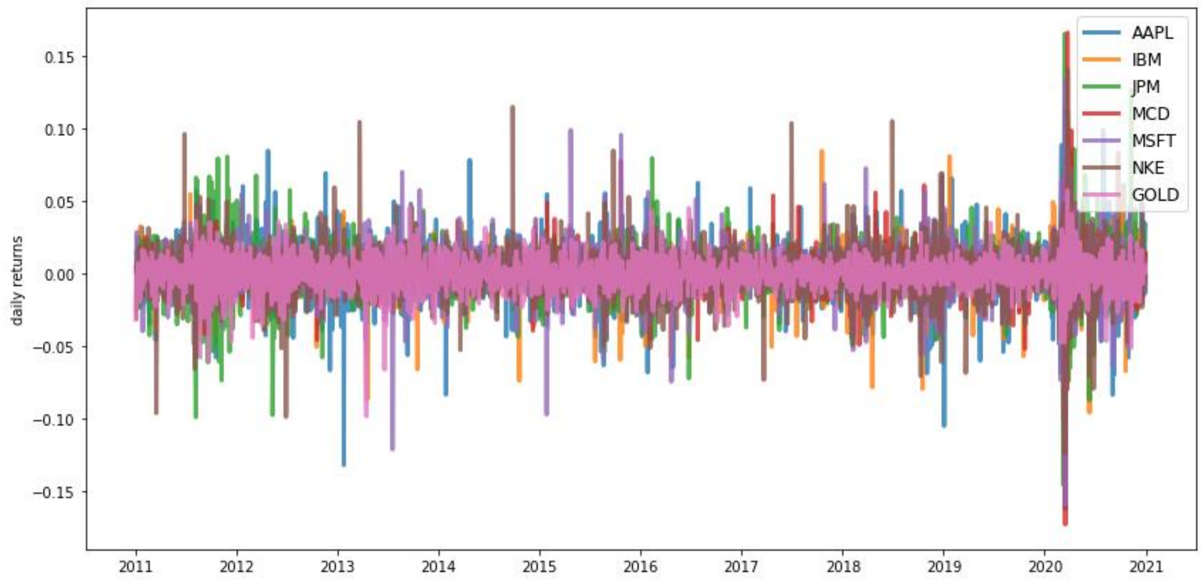
Monte Carlo optimization

Plot prices



Plot returns



Check metrics

Means:

AAPL 0.257426

IBM 0.016836

JPM 0.134468

MCD 0.132283

MSFT 0.230606

NKE 0.200816

GOLD 0.028618

Covariance:

	AAPL	IBM	JPM	MCD	MSFT	NKE	GOLD
AAPL	0.081177	0.027697	0.033939	0.021452	0.040876	0.029370	0.002107
IBM	0.027697	0.052320	0.036392	0.020267	0.030918	0.025687	0.000710
JPM	0.033939	0.036392	0.082852	0.026582	0.037617	0.035785	-0.003798
MCD	0.021452	0.020267	0.026582	0.038067	0.023809	0.024215	0.001155
MSFT	0.040876	0.030918	0.037617	0.023809	0.066118	0.030917	0.000554
NKE	0.029370	0.025687	0.035785	0.024215	0.030917	0.067818	-0.002378
GOLD	0.002107	0.000710	-0.003798	0.001155	0.000554	-0.002378	0.026868

Correlation:

	AAPL	IBM	JPM	MCD	MSFT	NKE	GOLD
AAPL	1.000000	0.424988	0.413843	0.385909	0.557949	0.395835	0.045110
IBM	0.424988	1.000000	0.552738	0.454138	0.525671	0.431237	0.018928
JPM	0.413843	0.552738	1.000000	0.473327	0.508242	0.477400	-0.080507
MCD	0.385909	0.454138	0.473327	1.000000	0.474567	0.476576	0.036122
MSFT	0.557949	0.525671	0.508242	0.474567	1.000000	0.461703	0.013139
NKE	0.395835	0.431237	0.477400	0.476576	0.461703	1.000000	-0.055711
GOLD	0.045110	0.018928	-0.080507	0.036122	0.013139	-0.055711	1.000000

Generate random weights:

```
array([0.22454855, 0.20413161, 0.04620425, 0.04290099, 0.13930751,
    0.19058471, 0.15232238])
```

calculate expected portfolio return from annualized return values

0.14788645952274218

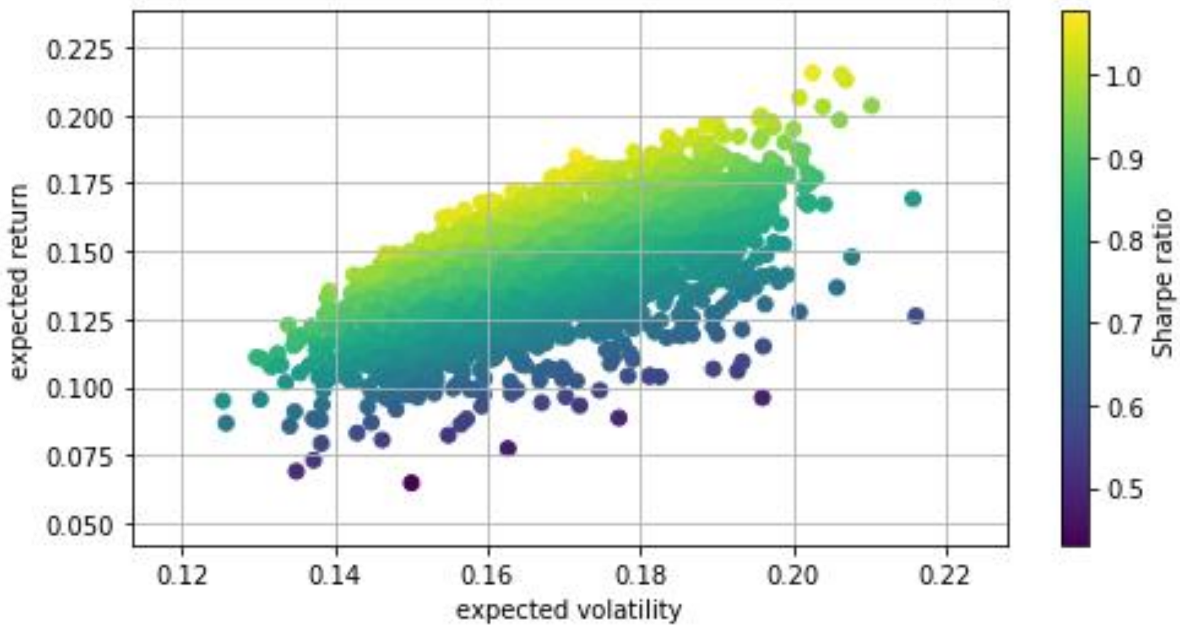
calculate expected portfolio variance

0.027993858069324654

calculate expected portfolio standard deviation

0.16731365177212723

Plot the results from the simulation



noa * [1. / noa,] # initial guess of the weights - equal distribution

[0.14285714285714285,

 0.14285714285714285,

 0.14285714285714285,

 0.14285714285714285,

 0.14285714285714285,

 0.14285714285714285,

 0.14285714285714285]

optimal portfolio composition results

array([0.274, 0.   , 0.   , 0.107, 0.253, 0.208, 0.158])

calculate statistics

array([0.189, 0.174, 1.086])

define a function that minimizes the variance

absolute minimum variance portfolio results:

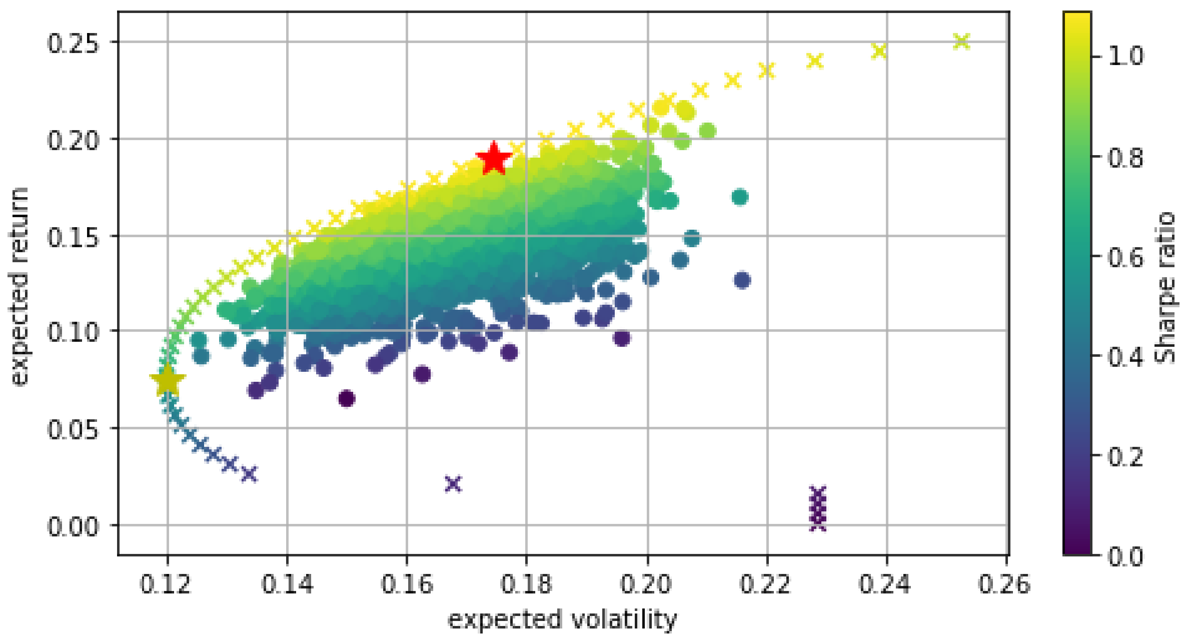
array([0.014, 0.108, 0.033, 0.211, 0.02 , 0.082, 0.531])

calculate statistics

array([0.074, 0.12 , 0.619])

Derive Efficient frontier

Plot optimization results – тук всички редове след този коментар преди празния ред се изпълняват наведнъж, за да се начертае всичко на една графика



Capital market line

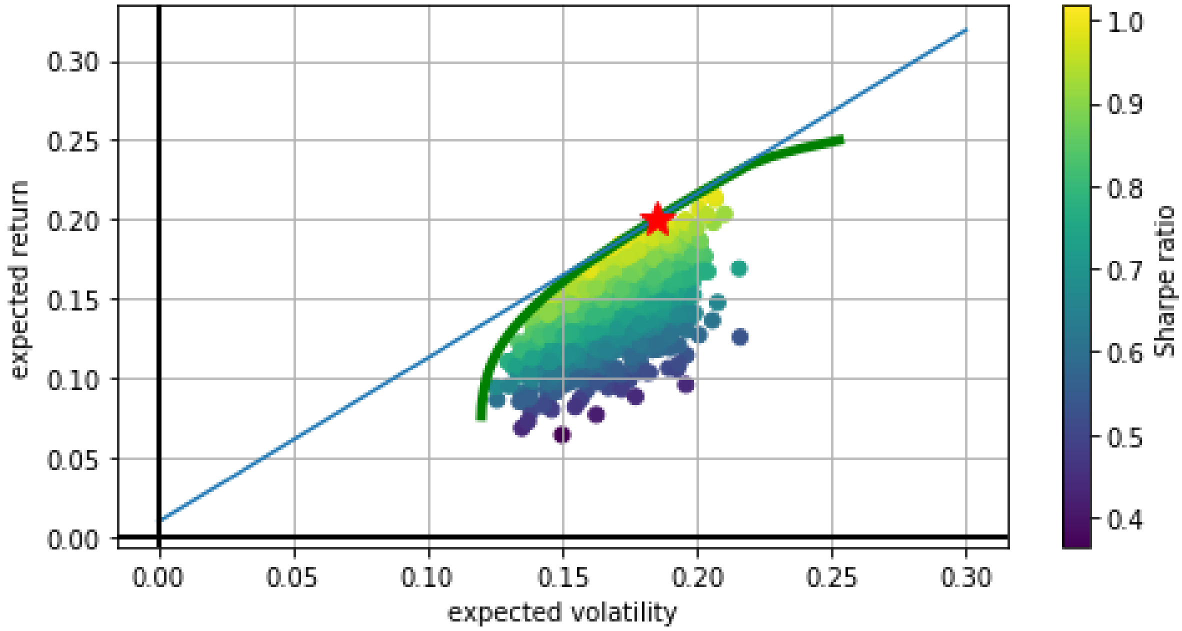
results from the numerical optimization

array([0.01   , 1.03062471, 0.18519155])

equations' results

array([0., -0., 0.])

plot optimal portfolio – пак всички редове преди празния наведнъж



optimal portfolio weights

optimal portfolio weights print

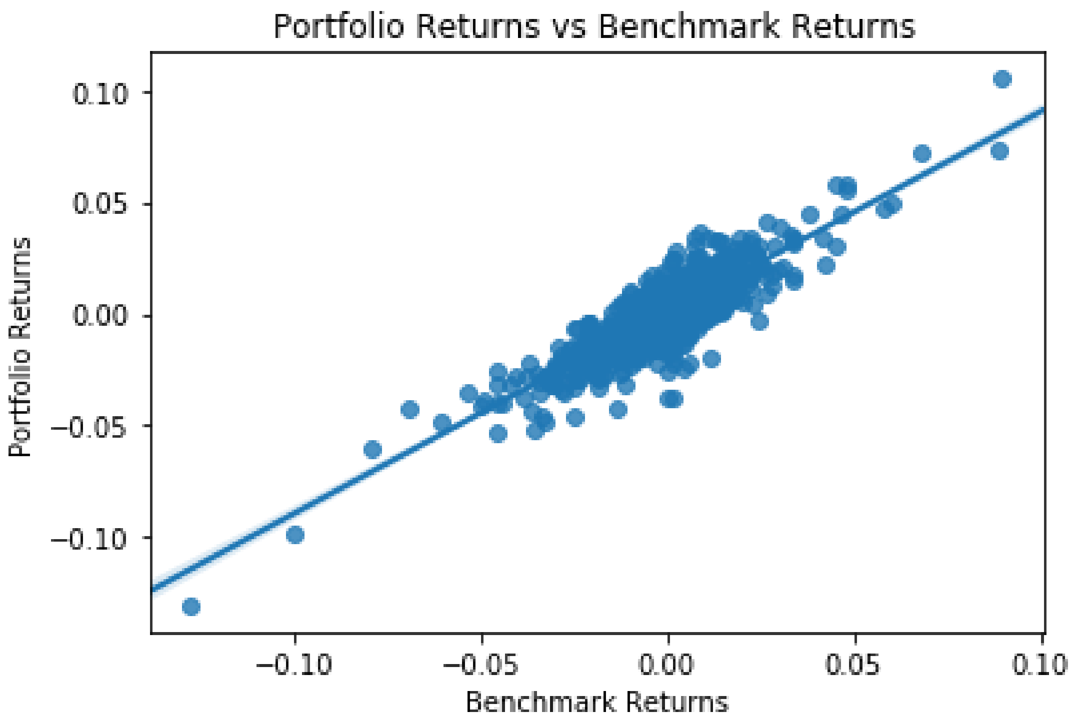
array([0.302, 0.   , 0.   , 0.09 , 0.277, 0.22 , 0.111])

Check statistics

array([0.20088233, 0.18521005, 1.08461893])

Alpha and Beta

plot optimal portfolio returns vs benchmark returns – всички редове до празния се изпълняват наведнъж



print beta

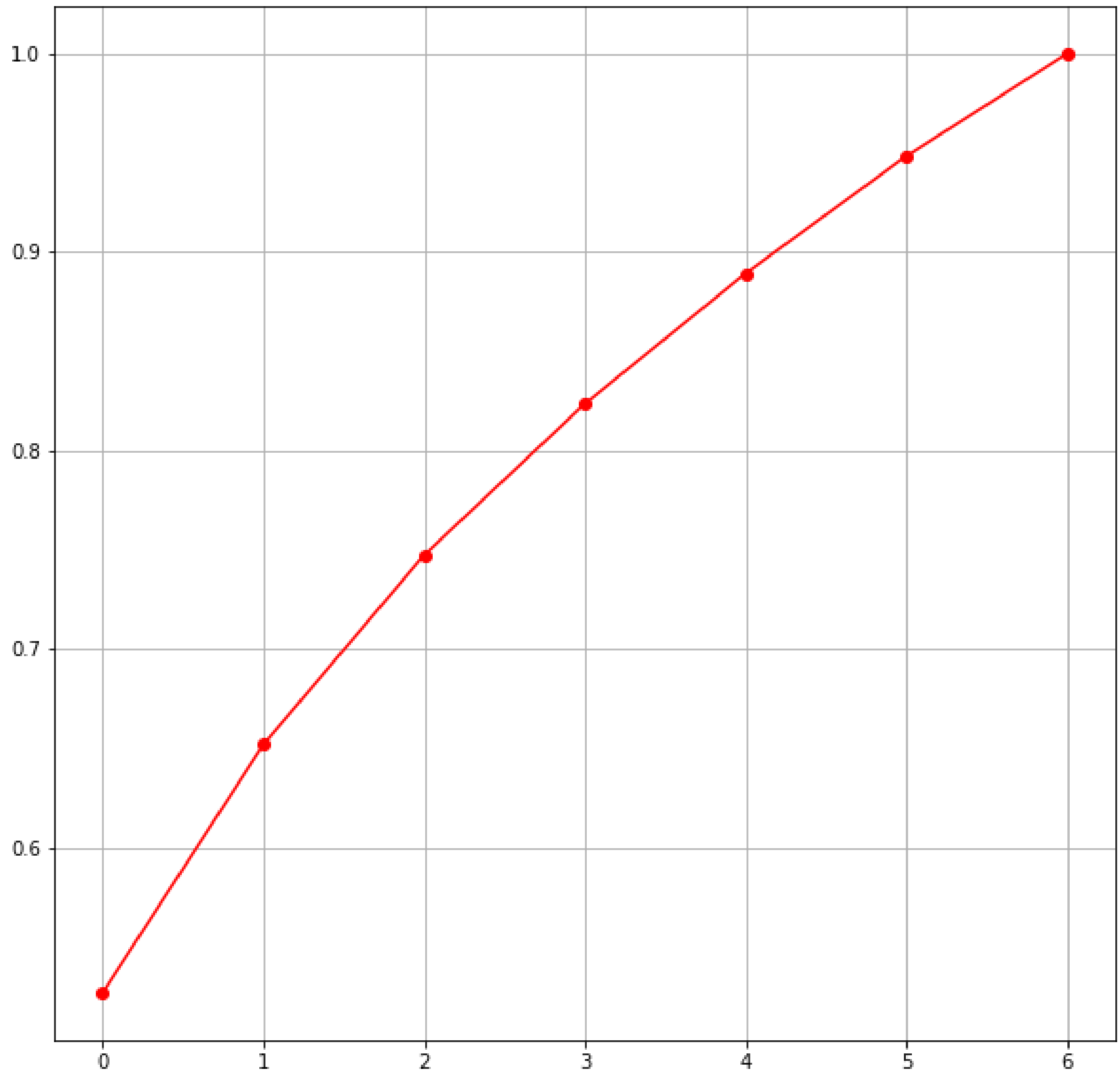
0.9016

print alpha

0.00041

############ PCA part ##############

find optimal number of components



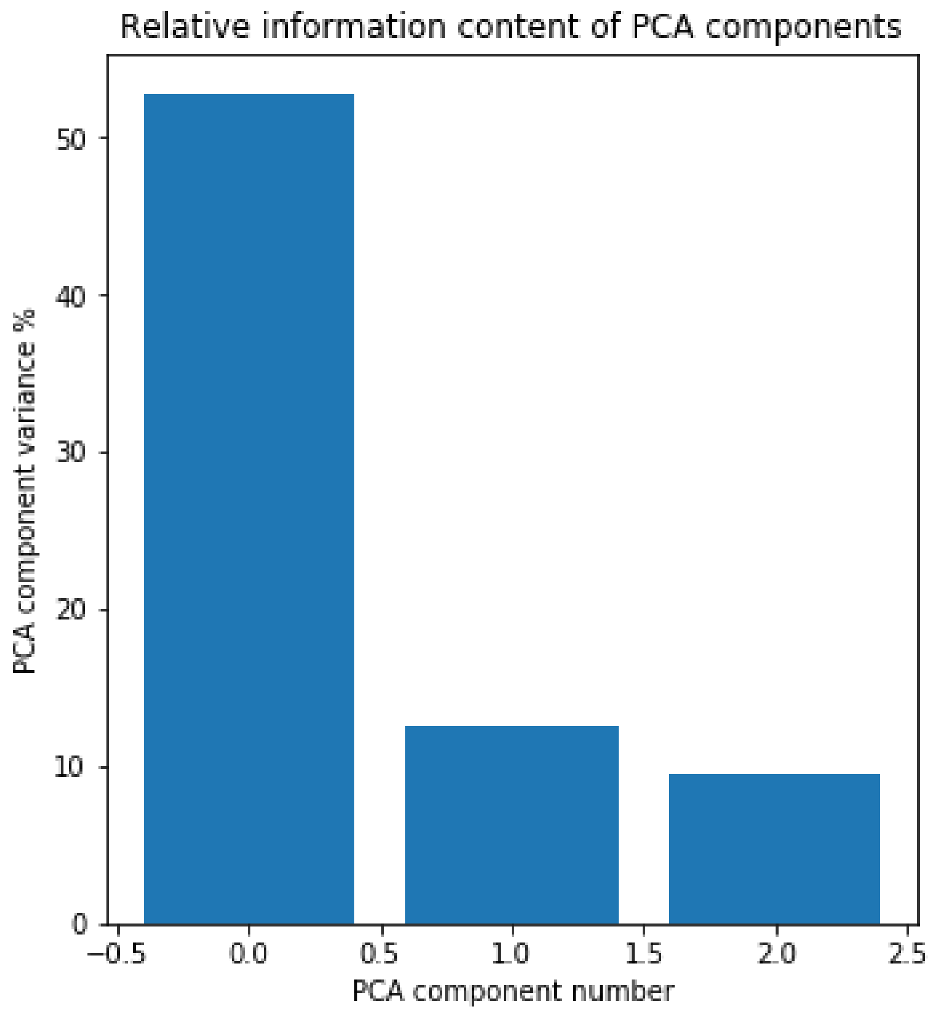
PCA with 3 components

explained variance ratio (first three components): 0.7473709517414355

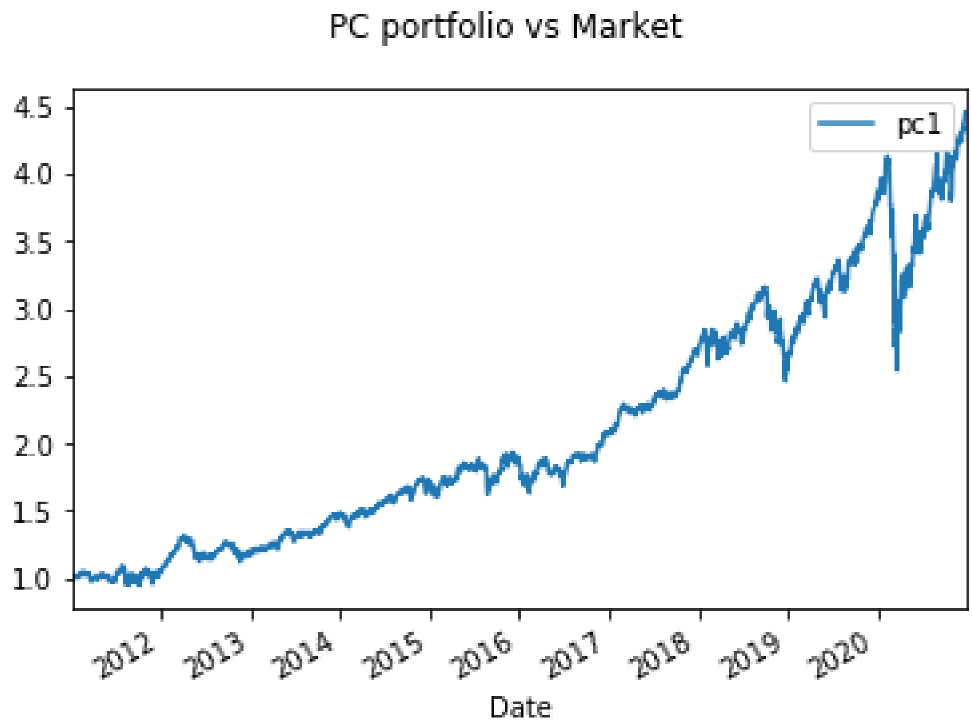
explained variance by each component

array([0.52689465, 0.12502717, 0.09544913])

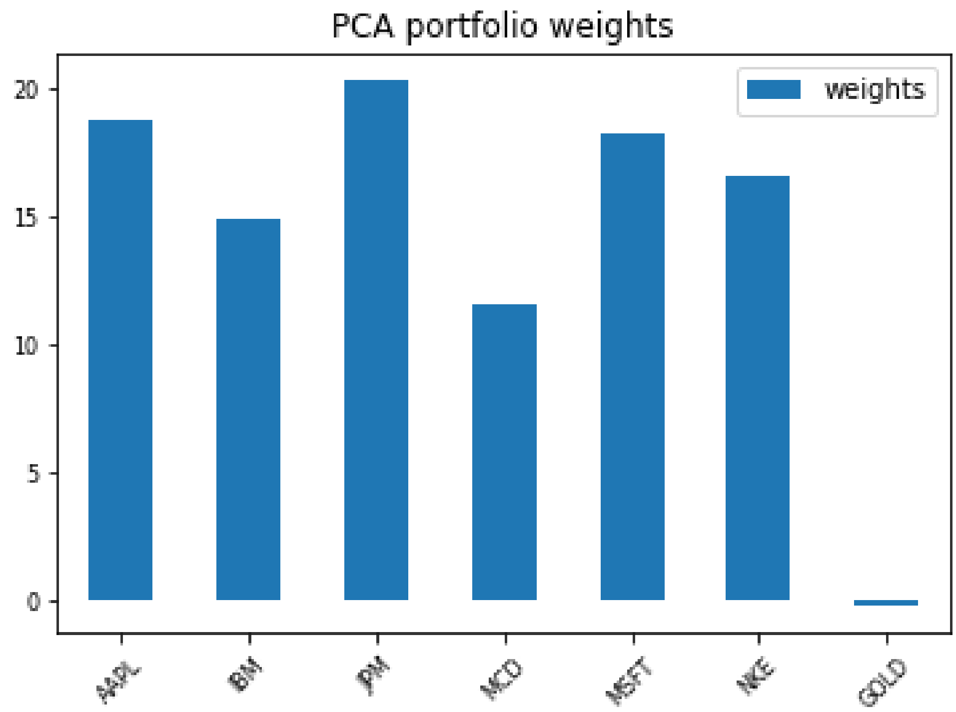
plot information content



#plot PC portfolio vs market



portfolio weights



weights

AAPL 18.741815

IBM 14.896949

JPM 20.321120

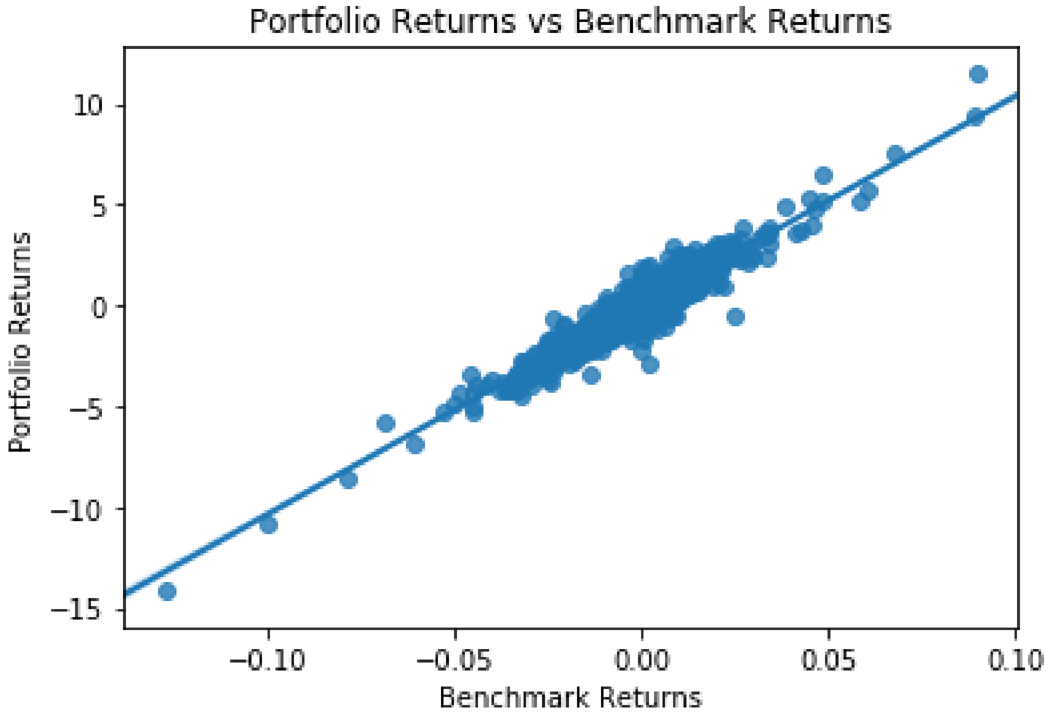
MCD   11.500847

MSFT 18.191867

NKE   16.572417

GOLD -0.225014 = 0

plot optimal portfolio returns vs benchmark returns



The portfolio beta is 103.6358

The portfolio alpha is 0.02231