01. Portfolio Volatility

August 1, 2021

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
close_price_df = pd.read_csv("./sample_stock_close_price.csv", index_col=0,␣
↪parse_dates=True)
close_price_df.columns = ['Samsung Electronics', 'SK Hynix', 'KAKAO', 'NAVER',␣
↪'LG Chemistry', 'Samsung SDI', 'HyunDai', 'Celltrion', 'KODEX Inverse']
close_price_df.tail(3)
```

[2]:

date	Samsung Electronics	SK Hynix	KAKAO	NAVER	LG Chemistry	Samsung SDI	HyunDai	Celltrion	KODEX Inverse
2021-07-28	79200.0	114000.0	148000.0	442000.0	835000.0	759000.0	222500.0	261500.0	3765.0
2021-07-29	79000.0	114000.0	148500.0	439500.0	835000.0	765000.0	222000.0	261500.0	3755.0
2021-07-30	78500.0	112500.0	147000.0	433500.0	842000.0	741000.0	218000.0	253500.0	3805.0

[3]:
```
close_price_df['cash'] = 1
close_price_df.tail(3)
```

[3]:

date	Samsung Electronics	SK Hynix	KAKAO	NAVER	LG Chemistry	Samsung SDI	HyunDai	Celltrion	KODEX Inverse	cash
2021-07-28	79200.0	114000.0	148000.0	442000.0	835000.0	759000.0	222500.0	261500.0	3765.0	1
2021-07-29	79000.0	114000.0	148500.0	439500.0	835000.0	765000.0	222000.0	261500.0	3755.0	1
2021-07-30	78500.0	112500.0	147000.0	433500.0	842000.0	741000.0	218000.0	253500.0	3805.0	1

[4]:
```
close_price_df.pct_change().std()
```

[4]:
```
Samsung Electronics    0.017170
SK Hynix               0.024069
KAKAO                  0.023309
NAVER                  0.022415
LG Chemistry           0.024015
Samsung SDI            0.024331
HyunDai                0.021317
Celltrion              0.030325
KODEX Inverse          0.011115
cash                   0.000000
dtype: float64
```

0.1 Portfolio Variance

$$\begin{aligned} \sigma_p^2 &= \sum_{i.j} w_i w_j Cov(r_i, r_j) \\ &= W^T \cdot K \cdot W \end{aligned} \tag{1}$$

[5]:
```
cov_matrix = close_price_df.pct_change().cov()
cov_matrix
```

[5]:

	Samsung Electronics	SK Hynix	KAKAO	NAVER
Samsung Electronics	0.000295	0.000210	0.000060	0.000082
SK Hynix	0.000210	0.000579	0.000100	0.000088
KAKAO	0.000060	0.000100	0.000543	0.000167
NAVER	0.000082	0.000088	0.000167	0.000502
LG Chemistry	0.000133	0.000170	0.000103	0.000081
Samsung SDI	0.000151	0.000196	0.000117	0.000096
HyunDai	0.000113	0.000131	0.000065	0.000069
Celltrion	0.000069	0.000111	0.000114	0.000090
KODEX Inverse	-0.000140	-0.000153	-0.000076	-0.000084
cash	0.000000	0.000000	0.000000	0.000000

	LG Chemistry	Samsung SDI	HyunDai	Celltrion
Samsung Electronics	0.000133	0.000151	0.000113	0.000069
SK Hynix	0.000170	0.000196	0.000131	0.000111
KAKAO	0.000103	0.000117	0.000065	0.000114
NAVER	0.000081	0.000096	0.000069	0.000090
LG Chemistry	0.000577	0.000290	0.000167	0.000116
Samsung SDI	0.000290	0.000592	0.000126	0.000116
HyunDai	0.000167	0.000126	0.000454	0.000082
Celltrion	0.000116	0.000116	0.000082	0.000920
KODEX Inverse	-0.000157	-0.000140	-0.000127	-0.000091
cash	0.000000	0.000000	0.000000	0.000000

	KODEX Inverse	cash
Samsung Electronics	-0.000140	0.0

```
SK Hynix                -0.000153   0.0
KAKAO                   -0.000076   0.0
NAVER                   -0.000084   0.0
LG Chemistry            -0.000157   0.0
Samsung SDI             -0.000140   0.0
HyunDai                 -0.000127   0.0
Celltrion               -0.000091   0.0
KODEX Inverse            0.000124   0.0
cash                     0.000000   0.0
```

0.1.1 Portfolio 1 Volatility

[6]:
```
portfolio_1_weights = np.array([0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0.2])
portfolio_1_weights, portfolio_1_weights.sum()
```

[6]:
```
(array([0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0. , 0.2]), 1.0)
```

[7]:
```
portfolio_1_variance = portfolio_1_weights.dot(cov_matrix).
 ↪dot(portfolio_1_weights)
portfolio_1_variance
```

[7]:
```
0.00011289727684812392
```

[8]:
```
portfolio_1_std = np.sqrt(portfolio_1_variance)
portfolio_1_std
```

[8]:
```
0.010625313023536009
```

0.1.2 Portfolio 2 Volatility

[9]:
```
portfolio_2_weights = np.array([0.1, 0.1, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0])
portfolio_2_weights, portfolio_2_weights.sum()
```

[9]:
```
(array([0.1, 0.1, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0. , 0. ]), 1.0)
```

[10]:
```
portfolio_2_variance = portfolio_2_weights.dot(cov_matrix).
 ↪dot(portfolio_2_weights)
portfolio_2_variance
```

[10]:
```
0.00018539262138360 97
```

[11]:
```
portfolio_2_std = np.sqrt(portfolio_2_variance)
portfolio_2_std
```

[11]:
```
0.013615895908224684
```

0.1.3 Portfolio Volatility Compare

[12]:
```
print('portfolio 1 Volatility : ', round(portfolio_1_std, 4))
print('portfolio 2 Volatility : ', round(portfolio_2_std, 4))
print('portfolio Volatility delta : ', round(portfolio_2_std - portfolio_1_std,␣
↪6))
```

```
portfolio 1 Volatility :  0.0106
portfolio 2 Volatility :  0.0136
portfolio Volatility delta :  0.002991
```

0.1.4 Trying to Get Portfolio Volatility Delta using Portfolio Weights Delta

[13]:
```
portfolio_weights_delata = portfolio_2_weights - portfolio_1_weights
portfolio_weights_delata
```

[13]:
```
array([ 0. ,  0. ,  0.2,  0. ,  0. ,  0. ,  0. ,  0. ,  0. , -0.2])
```

[14]:
```
portfolio_variance_delta = portfolio_weights_delata.dot(cov_matrix).
↪dot(portfolio_weights_delata)
```

[15]:
```
np.sqrt(portfolio_variance_delta)
```

[15]:
```
0.0046617005562492346
```

-> It's not an appropriate methodology. In my opinion, it seems impossible to obtain with just the delta of portfolio weights.