01. Portfolio Volatility

August 2, 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',␣
↪'KODEX Inverse']
close_price_df.tail(3)
```

[2]:

date	Samsung Electronics	SK Hynix	KAKAO	NAVER	KODEX Inverse
2021-07-28	79200.0	114000.0	148000.0	442000.0	3765.0
2021-07-29	79000.0	114000.0	148500.0	439500.0	3755.0
2021-07-30	78500.0	112500.0	147000.0	433500.0	3805.0

[3]:
```
close_price_df.pct_change().std()
```

[3]:
```
Samsung Electronics    0.017170
SK Hynix               0.024069
KAKAO                  0.023309
NAVER                  0.022415
KODEX Inverse          0.011115
dtype: float64
```

0.1 Portfolio Variance

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

[4]:
```
cov_matrix = close_price_df.pct_change().cov()
cov_matrix
```

[4]:

	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

```
KODEX Inverse                  -0.000140 -0.000153 -0.000076 -0.000084
```

```
                     KODEX Inverse
Samsung Electronics      -0.000140
SK Hynix                 -0.000153
KAKAO                    -0.000076
NAVER                    -0.000084
KODEX Inverse             0.000124
```

0.1.1 Portfolio 1 Volatility

[5]:
```
portfolio_1_weights = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
portfolio_1_weights, portfolio_1_weights.sum()
```

[5]:
```
(array([0.2, 0.2, 0.2, 0.2, 0.2]), 1.0)
```

[6]:
```
portfolio_1_variance = portfolio_1_weights.dot(cov_matrix).
 ↪dot(portfolio_1_weights)
portfolio_1_variance
```

[6]:
```
0.0001019404461898898
```

[7]:
```
portfolio_1_std = np.sqrt(portfolio_1_variance)
portfolio_1_std
```

[7]:
```
0.010096556154941634
```

0.1.2 Portfolio 2 Volatility

[8]:
```
portfolio_2_weights = np.array([0.1, 0.1, 0.4, 0.2, 0.2])
portfolio_2_weights, portfolio_2_weights.sum()
```

[8]:
```
(array([0.1, 0.1, 0.4, 0.2, 0.2]), 1.0)
```

[9]:
```
portfolio_2_variance = portfolio_2_weights.dot(cov_matrix).
 ↪dot(portfolio_2_weights)
portfolio_2_variance
```

[9]:
```
0.0001405378830009183
```

[10]:
```
portfolio_2_std = np.sqrt(portfolio_2_variance)
portfolio_2_std
```

[10]:
```
0.011854867481373139
```

0.1.3 Portfolio Volatility Compare

[11]:
```
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 :  0.0101
portfolio 2 Volatility :  0.0119
portfolio Volatility delta :  0.001758
```

0.1.4 Trying to Get Portfolio Volatility Delta using Portfolio Weights Delta

[12]:
```
portfolio_weights_delta = portfolio_2_weights - portfolio_1_weights
portfolio_weights_delta
```

[12]:
```
array([-0.1, -0.1,  0.2,  0. ,  0. ])
```

[13]:
```
portfolio_variance_delta = portfolio_weights_delta.dot(cov_matrix).
↪dot(portfolio_weights_delta)
```

[14]:
```
np.sqrt(portfolio_variance_delta)
```

[14]:
```
0.005318472294771918
```

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