

Section 3

Date	
Property	
Weeks	2주차

2. The only free lunch in Finance

포트폴리오 설계



A와 B 두 개의 asset으로 구성된 포트폴리오가 있다고 가정.

A : return 4% , risk 10%

B : return 6% , risk 14%

A와 B의 비중을 50:50으로 한다고 가정, 포트폴리오의 예상 return & risk 는?

→ 절반인 return 5%, risk 12% 가 아니다!

왜냐하면 두 asset이 완벽히 correlated 하지 않기 때문에!

수익(return)은 5%가 맞아도, 변동성(volatility;risk) 는 두 자산의 변동성보다 작을 것

수익을 식으로 나타내면

$$R(w_A, w_B) = w_A \times R_a + w_B \times R_b$$

변동성을 식으로 나타내면

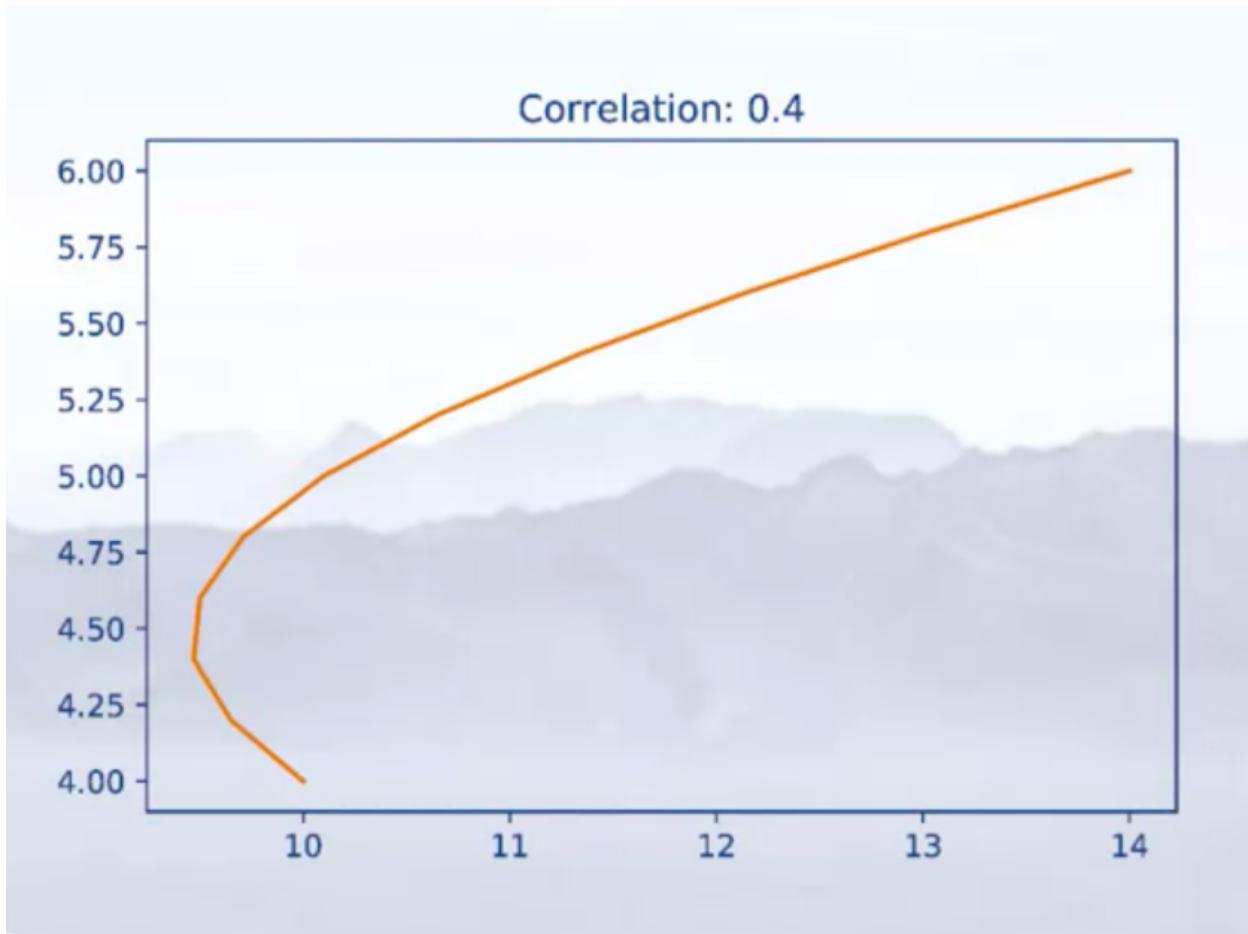
$$\sigma^2(w_a, w_b) = \sigma_A^2 w_A^2 + \sigma_B^2 w_B^2 + 2w_A w_B \sigma_A \sigma_B \rho_{A,B}$$

(a의 분산 × a의 가중치제곱)+(b의 분산×b의 가중치제곱)+(2×a의 가중치×b의 가중치×a의 변동성×b의 변동성×a,b의 상관관계)

상관관계 식

	Portfolio W(A)	W(B)	Return	Vol
1	0.0	1.0	6.0	14.00
2	0.1	0.9	5.8	13.03
3	0.2	0.8	5.6	12.14
4	0.3	0.7	5.4	11.34
5	0.4	0.6	5.2	10.65
6	0.5	0.5	5.0	10.10
7	0.6	0.4	4.8	9.71
8	0.7	0.3	4.6	9.50
9	0.8	0.2	4.4	9.47
10	.9	0.1	4.2	9.65
11	1.0	0.0	4.0	10.00

a,b의 risk은 12%가 아닌 10.10% 다!

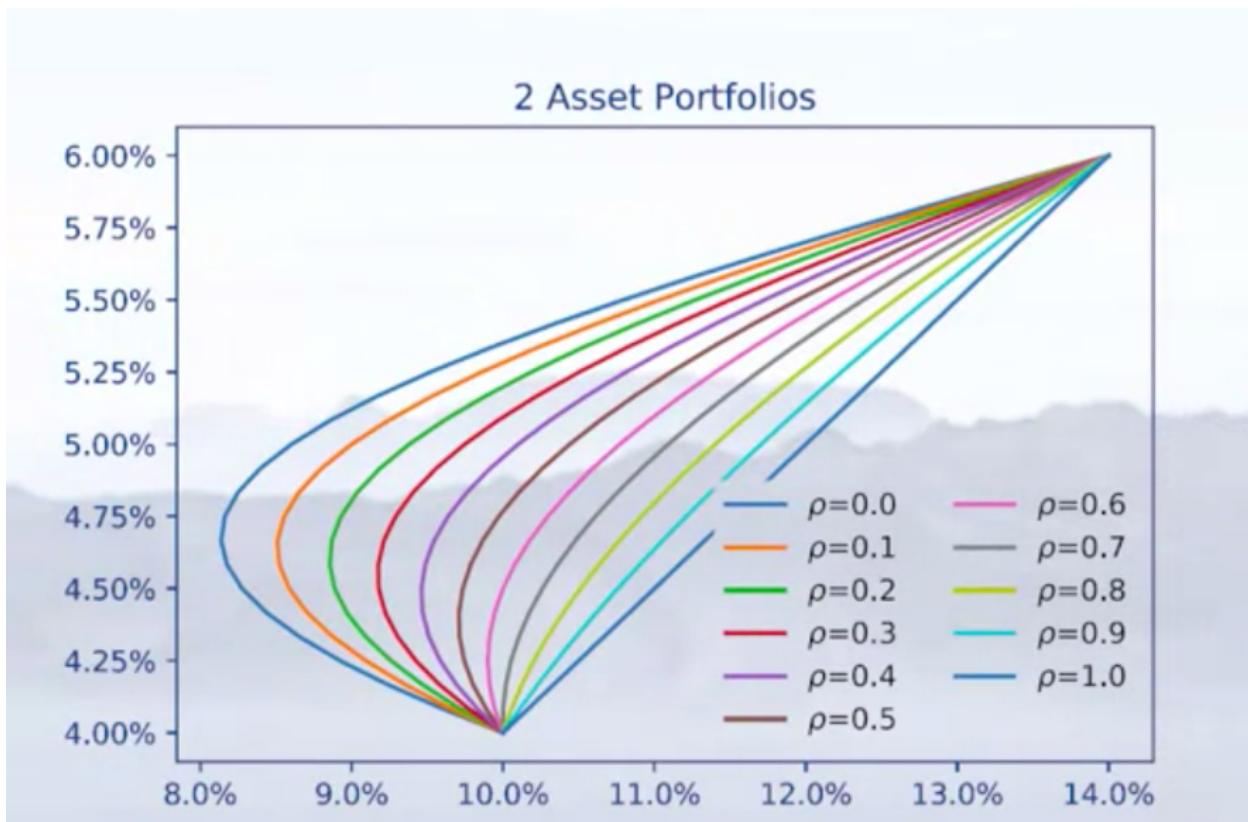


상관계수가 0.4인 A와 B 사이의 가능한 모든 포트폴리오 : A-B 커브

각 변동성이 10%, 14% 인 asset의 포트폴리오 구성에서 최소값인 10%보다 작은 변동성을 가질 수 있다! → 매직!

→ 가장 작은 변동성을 가질 수 있도록, 두 asset의 비율을 잘 조정해서 섞어야

- 모든 correlation 을 그래프로 표현하면 다음과 같다.
- 완벽히 상관이 있으면 (perfectly correlated) $\rho=1.0 \rightarrow$ 직선
- 점점 상관도가 떨어질수록 커브가 생김
→ 상관도가 떨어질수록, 두 asset의 최소 변동성보다 상대적으로 작은 변동성을 가진 포트폴리오 조합이 있다!
(decorrelated assets일수록 lower volatility인 포트폴리오 구성 가능)



3. Lab Session-Efficient frontier-Part 1

30개 분야의 수익률을 가져온다.

```
ind = pd.read_csv("data/ind30_m_vw_rets.csv", header = 0, index_col = 0, parse_dates=True)/100
```

	Food	Beer	Smoke	Games	Books	Hshld	Clhs	Hlth	Chems	Txtls	...	Telcm	Servs	BusEq
192607	0.0056	-0.0519	0.0129	0.0293	0.1097	-0.0048	0.0808	0.0177	0.0814	0.0039	...	0.0083	0.0922	0.0206
192608	0.0259	0.2703	0.0650	0.0055	0.1001	-0.0358	-0.0251	0.0425	0.0550	0.0814	...	0.0217	0.0202	0.0439
192609	0.0116	0.0402	0.0126	0.0658	-0.0099	0.0073	-0.0051	0.0069	0.0533	0.0231	...	0.0241	0.0225	0.0019
192610	-0.0306	-0.0331	0.0106	-0.0476	0.0947	-0.0468	0.0012	-0.0057	-0.0476	0.0100	...	-0.0011	-0.0200	-0.0109
192611	0.0635	0.0729	0.0455	0.0166	-0.0580	-0.0054	0.0187	0.0542	0.0520	0.0311	...	0.0163	0.0377	0.0364

5 rows × 30 columns

- 데이터 전처리 작업

1. 시계열 데이터로 처리하기 위해 integer 형태의 index를 datetime으로 type을 바꿔줌.

→ string datetime에 Str 바꾸는게 있다. from datetime import strptime

```
ind.index = pd.to_datetime(ind.index, format = "%Y%m").to_period('M')
```

2. 컬럼명 내에 공백 제거 :

str 함수로 컬럼명을 문자열 형태로 인식하게 한다.

strip 함수로 문자열의 양 끝에 존재하는 공백과 \n을 제거한다.

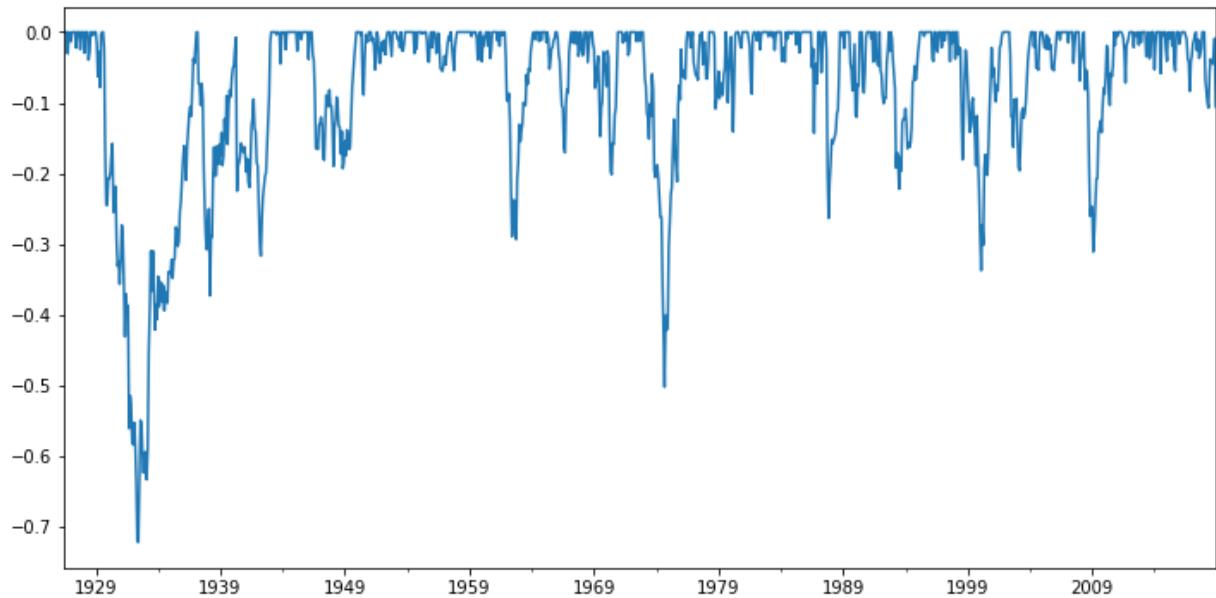
```
ind.columns = ind.columns.str.strip()
```

→ Columns로 해서 Series로 인식했는지 확인하는 것이 중요.

→ Pandas DataFrame Series 구분. / 임시 변환 vs 결과 변환 / copy로 원본 백업

```
erk.drawdown(ind["Food"]["Drawdown"].plot.line(figsize = (12, 6))
```

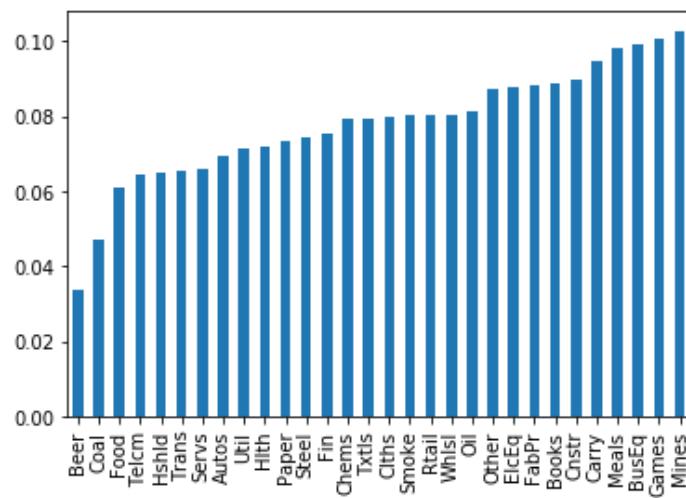
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dcce50d748>
```



기존에 설정해둔 함수를 이용해 drawdown을 확인한다.

```
erk.var_gaussian(ind, modified = True).sort_values().plot.bar()
```

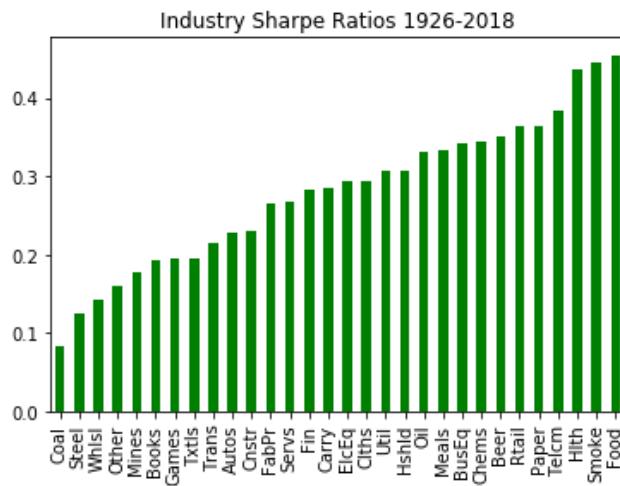
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dcce5f9f48>
```



var_gaussian으로 VAR 확인

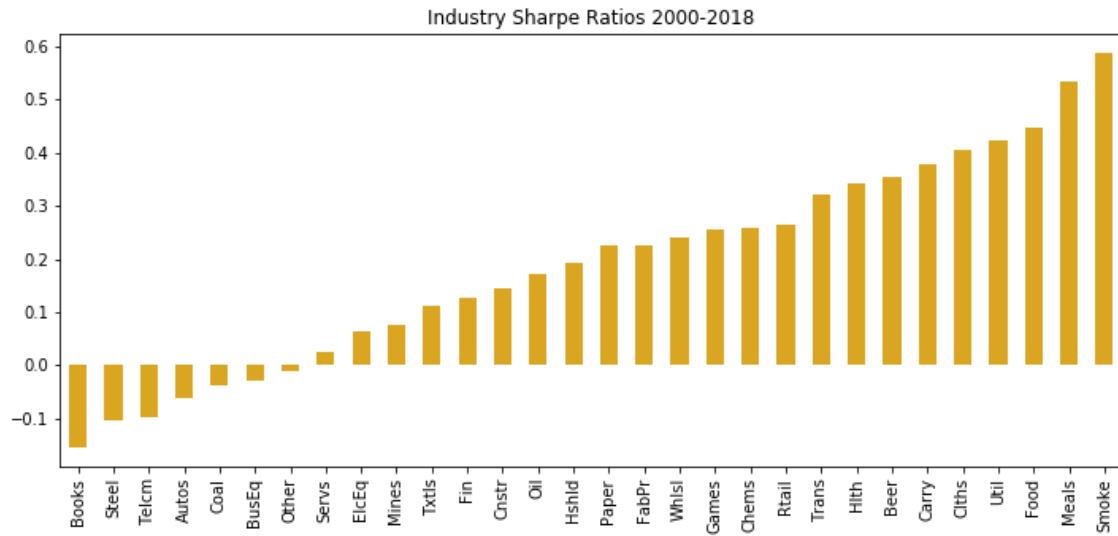
```
erk.sharpe_ratio(ind, 0.03, 12).sort_values().plot.bar(title = "Industry Sharpe Ratios 1926-2018",
color = "green")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dcce6ea448>
```



```
erk.sharpe_ratio(ind["2000":], 0.03, 12).sort_values().plot.bar(title = "Industry Sharpe Ratios 2000-2018",
color = "goldenrod", figsize=(12,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dcceaa53c8>
```



Sharpe Ratio 확인 Books, Steel 등은 무위험 수익률보다 낮을 수익률을 보임을 알 수 있다.

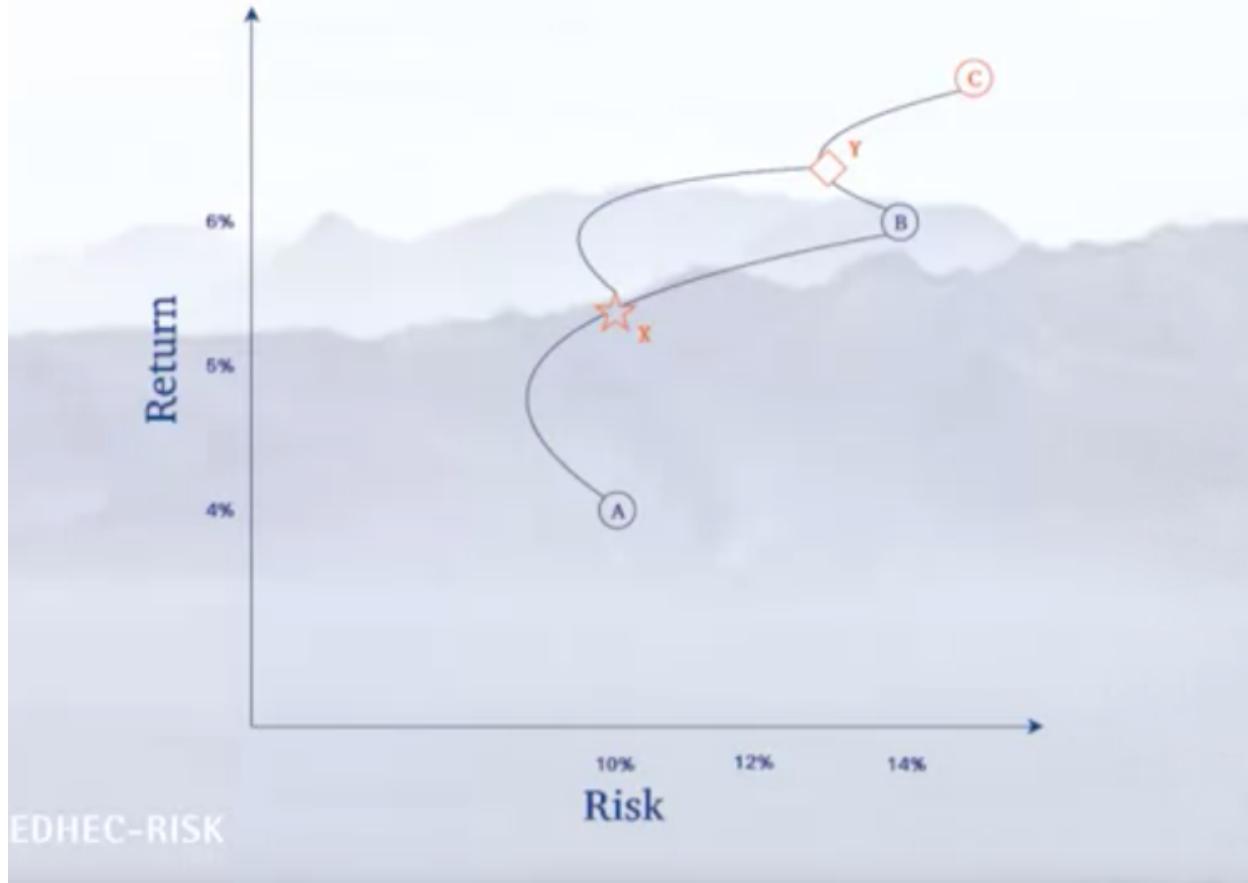


Efficient Frontier를 확인하기 위한 사전 작업으로 Annualized Return과 Covariance를 확인한다.

4. Markowitz Optimization and the Efficient Frontier

두 자산에 자본(capital)을 할당할 때 , 두 자산 변동성의 최소값보다 더 작은 최소값 조합이 가능한 걸 확인함.

LET'S INTRODUCE A THIRD ASSET C

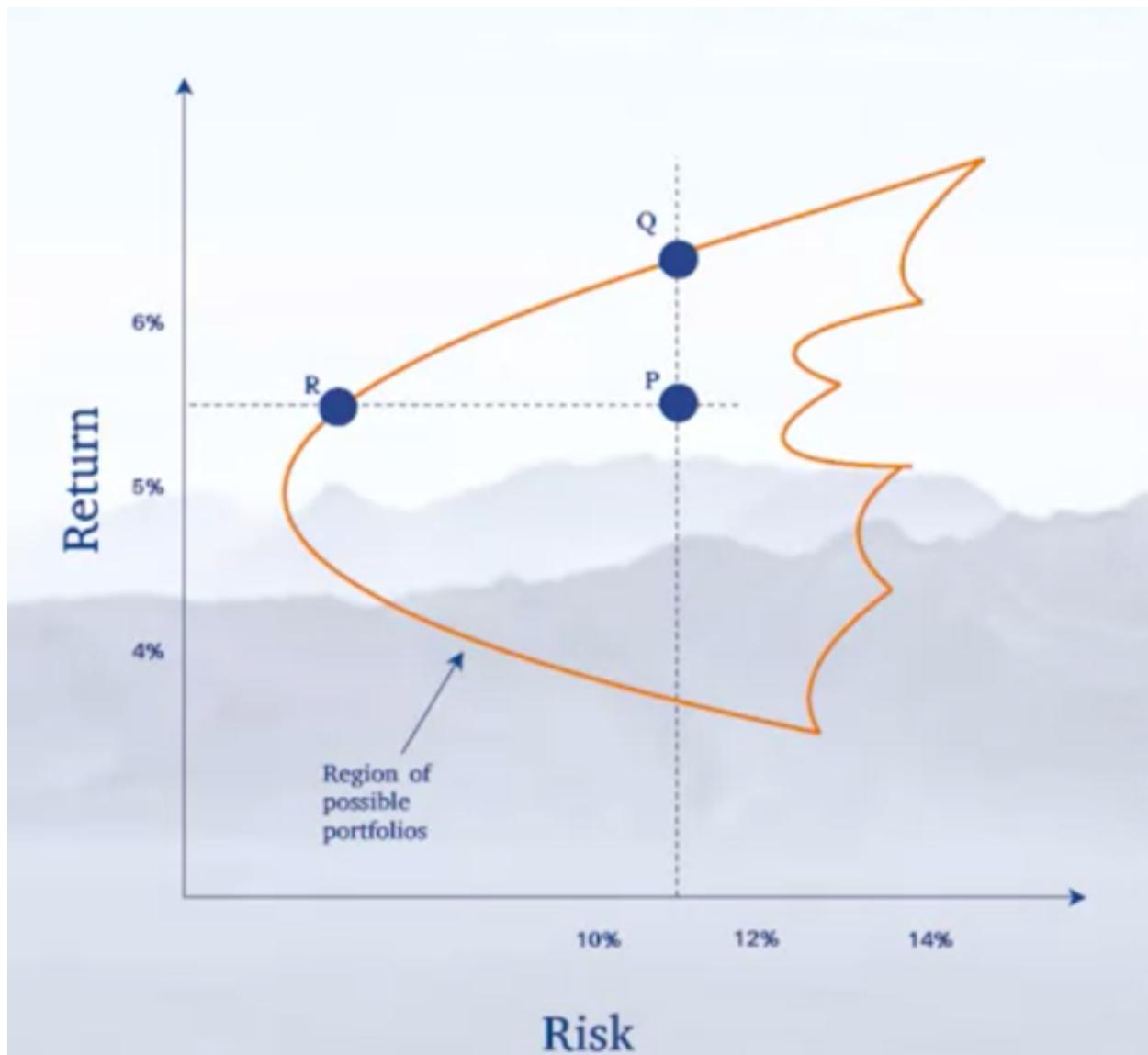


A와 B 사이의 가능한 모든 포트폴리오 : A-B 커브에 새로운 asset C 추가!

점 X : A와 B 사이의 한 점, A와 B의 가중치를 뜻함

마찬가지로, 점 Y : B와 C 사이의 한 점, B와 C의 가중치를 뜻함

점 X 와 점 Y으로 만들 수 있는 포트폴리오 조합인 새로운 곡선

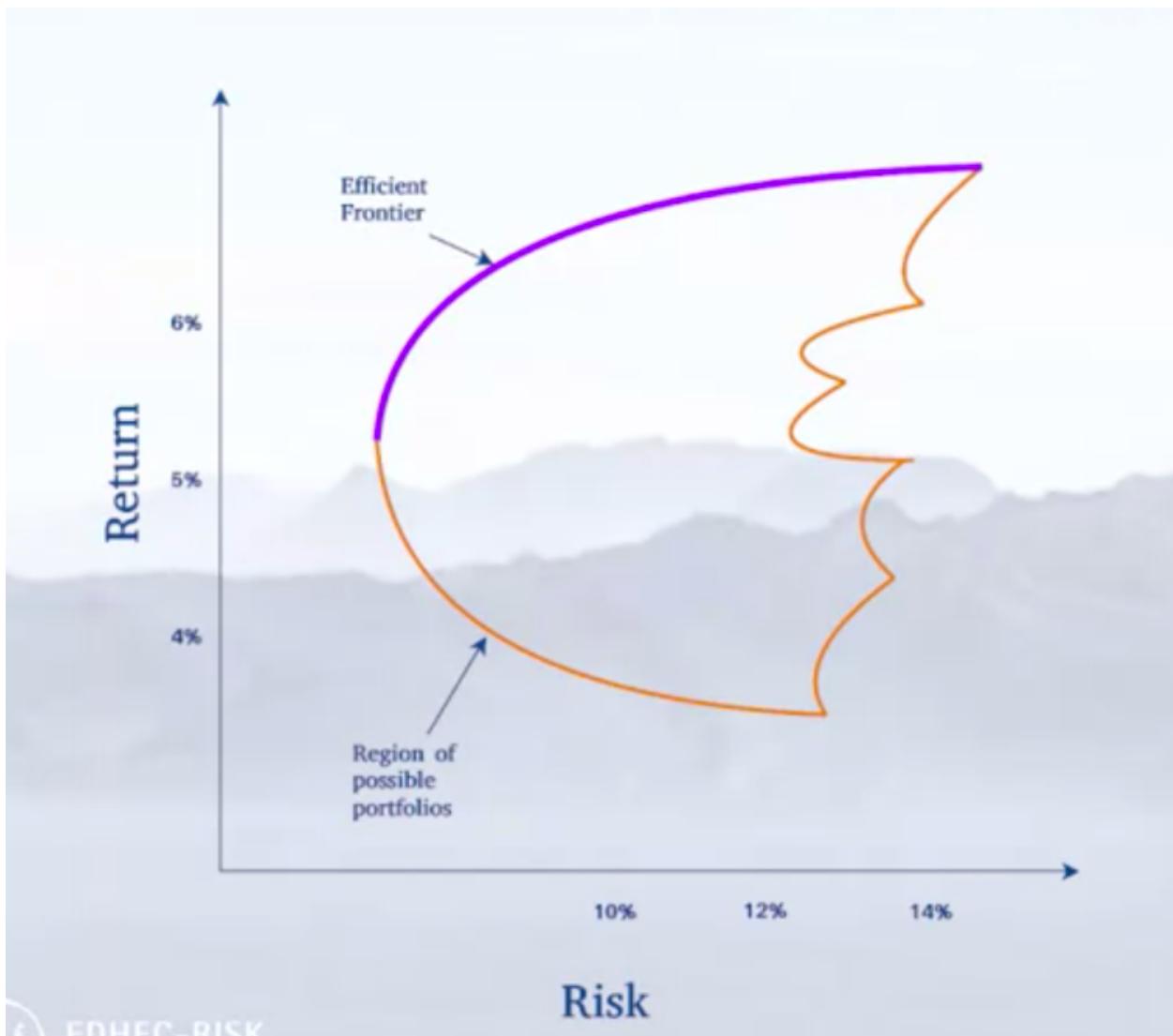


asset이 추가되면, 포트폴리오는 결국 곡선이 아닌 '면적'이 됨

당연히 면적 안에 있는 점 P보다는, 면적을 구성하는 **선에 위치한** Q나 R이 더 적합

점 R은 점 P와 같은 return에 risk가 더 작음

점 Q는 점 P와 같은 risk에 return이 더 높음



포트폴리오 면적 안에 있는게 아닌 , 포트폴리오 엣지(edge)나 프론티어(frontier)에 있는 점으로 구성해야! (efficient frontier : risk 와 return 이 최적인 곳)

efficient frontier 중 최적의 포트폴리오 찾는 것을 코드로 배울 것!

5. Applying quadprog to draw the efficient Frontier

Efficient Frontier를 그리는 법!

1. 이론을 세운다
2. 해당 정보들을 통해 Efficient Frontier를 최적화 한다.

▼ 개념

수익률

$$R_p = \sum_{i=1}^k w_i R_i$$

아래와 같이 행렬로 표현 가능

$$R_p = w^T R$$

분산

$$\sigma_p^2 = \sum_{i=1}^k \sum_{j=1}^k w_i w_j \sigma_i \sigma_j \rho_{i,j}$$

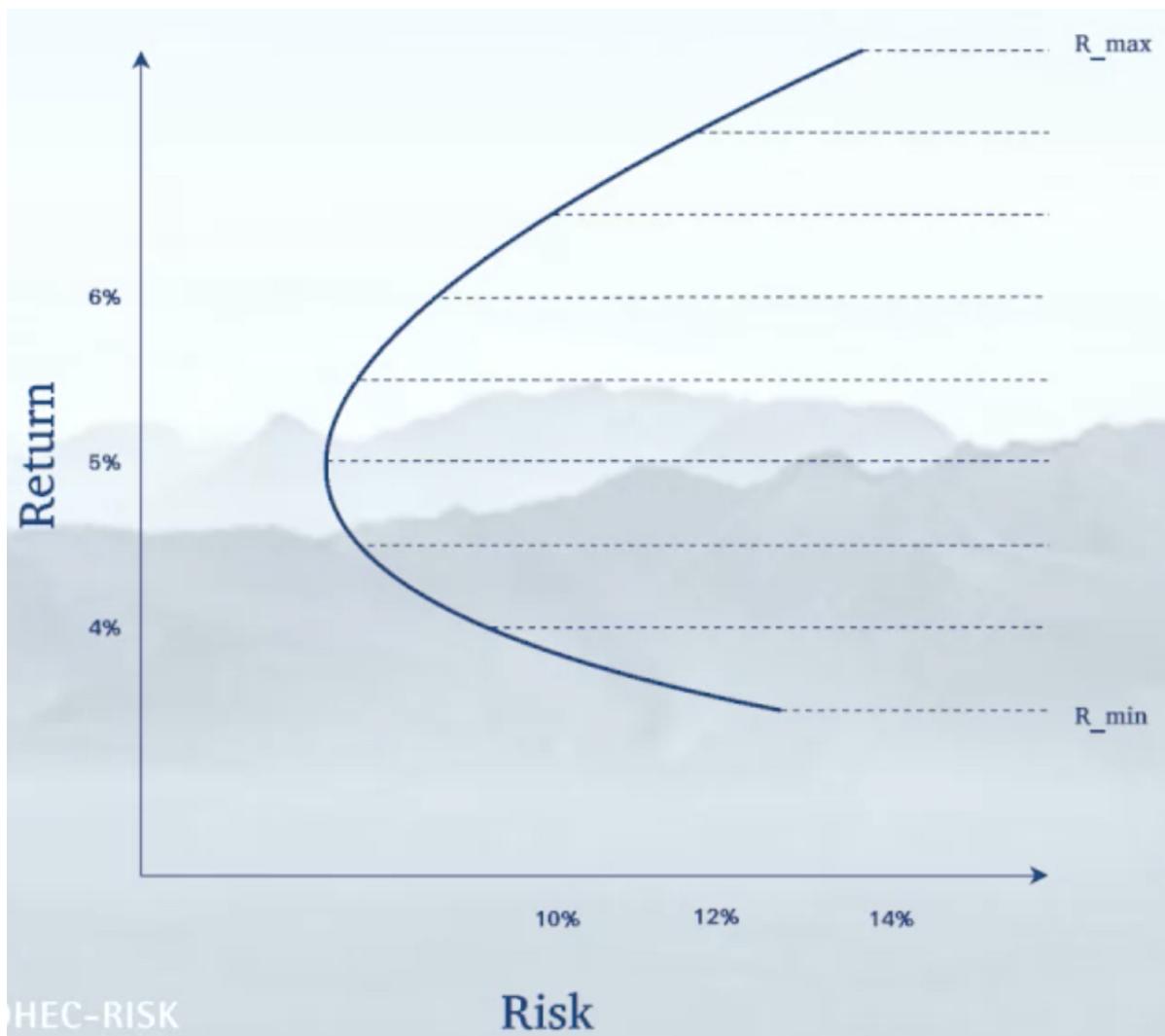
모든 경우의 수(i, j)를 더해서 구함 (가중치 i, 가중치 j, 공분산 i,j)

아래와 같이 행렬로 표현 가능

$$\sigma_p^2 = w^T \sum w$$

diagona element : $\sigma_{ii} = \sigma_i \sigma_i \rho_{ii}$

▼ 실전



`x : Risk(volatility) / y : return`

1. 최소/최대 Return을 구한다.
2. 적절한 비율로 나눈다.
3. 각각의 Return에 맞는 Risk를 구한다
4. 잇는다

표현방법

Quadratic Form

$$\begin{aligned}
 & \text{Minimize : } \frac{1}{2} w_t^T w \\
 & \text{Subject to} \\
 & w_T R = r_0 \\
 & w^T 1 = 1 \\
 & w \geq 0
 \end{aligned}$$

Quadratic Optimizer을 통해 해결...

6. Lab Session-Asset Efficient Frontier-Part 2

```

def portfolio_return(weights, returns):
    """
    Weights -> Returns
    """
    return weights.T @ returns

def portfolio_vol(weights, covmat):
    """
    Weights -> Vol
    """
    return weights.T @ covmat @ weights

```

return 과 volatility 를 matrix 계산을 통해 쉽게 구할수 있다.

2 Asset Frontier

```

n_points = 20
weights = [np.array([w, 1-w]) for w in np.linspace(0, 1, n_points)]

```

n 개의 weights를 생성한다

```
weights  
[array([0., 1.]),  
 array([0.05263158, 0.94736842]),  
 array([0.10526316, 0.89473684]),  
 array([0.15789474, 0.84210526]),  
 array([0.21052632, 0.78947368]),  
 array([0.26315789, 0.73684211]),  
 array([0.31578947, 0.68421053]),  
 array([0.36842105, 0.63157895]),  
 array([0.42105263, 0.57894737]),  
 array([0.47368421, 0.52631579]),  
 array([0.52631579, 0.47368421]),  
 array([0.57894737, 0.42105263]),  
 array([0.63157895, 0.36842105]),  
 array([0.68421053, 0.31578947]),  
 array([0.73684211, 0.26315789]),  
 array([0.78947368, 0.21052632]),  
 array([0.84210526, 0.15789474]),  
 array([0.89473684, 0.10526316]),  
 array([0.94736842, 0.05263158]),  
 array([1., 0.])]
```

```
rets = [erk.portfolio_return(w, er[l]) for w in weights]  
vols = [erk.portfolio_vol(w, cov.loc[l,l]) for w in weights]  
ef = pd.DataFrame({"R": rets, "Vol": vols})  
ef.plot.scatter(x, y)
```

7. Lab Session-Applying Quadprog to Draw the Efficient Frontier

<이전 실습 내용에서 이어짐>

```
ind = erk.get_ind_returns()  
er = erk.annualize_rets(ind["1996":"2000"], 12)  
cov = ind["1996":"2000"].cov()
```

erk.get_ind_returns(): 데이터는 주어진 csv 파일을 읽어와서 시계열 데이터로 바꾸고, column에서 빈칸을 지워 처리하기 쉬운 형태로 만든것

er → 1996년부터 2000년까지 return을 annualize한것.

cov→ 1996년부터 2000년까지 종목들의 covariance를 구한것.

<7장에서는 2 asset코드였다면 N-asset으로 확장시킬것>

```
def plot_ef(n_points, er, cov, style=".-"):
    """
    Plots the N-asset efficient frontier
    """
    weights = ??????
    rets = [portfolio_return(w, er) for w in weights]
    vols = [portfolio_vol(w, cov) for w in weights]
    ef = pd.DataFrame({
        "Returns": rets,
        "Volatility": vols
    })
    return ef.plot.line(x="Volatility", y="Returns", style=style)
...
```

확장할 때 바꿔야 할 부분?

-return이나 volatility에 대한 부분은 동일함.

-weight 설정만 바꾸면 됨.

weight는 target return의 volatility를 최소화 시키는 방향으로 가야함.

```
def plot_ef(n_points, er, cov):
    """
    Plots the multi-asset efficient frontier
    """
    weights = optimal_weights(n_points, er, cov)
    rets = [portfolio_return(w, er) for w in weights]
    vols = [portfolio_vol(w, cov) for w in weights]
    ef = pd.DataFrame({
        "Returns": rets,
        "Volatility": vols
    })
    return ef.plot.line(x="Volatility", y="Returns", style='.-')
```

<Weight를 optimize 시키기 → optimal_weights함수 정의>

```
[2]: from scipy.optimize import minimize
```

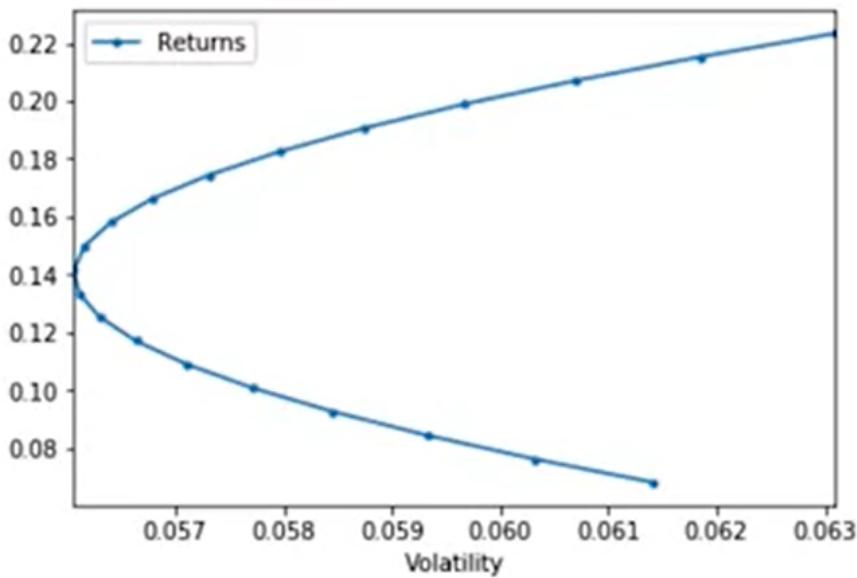
2. weights 세트를 optimal해주는 함수 define

```
def optimal_weights(n_points, er, cov):
    """
    -> list of weights to run the optimizer on to minimize the vol
    """
    target_rs = np.linspace(er.min(), er.max(), n_points)
    weights = [minimize_vol(target_return, er, cov) for target_return in target_rs]
    return weights
```

3. volatility를 minimize하는 함수 정의하기

target return → 어떠한 return을 정했을때, w값들이 어떻게 되는지?

예를 들어서,



target return을 0.15라고 할때, 그때의 volatility가 최소화된 w값들을 찾자는게 우리의 목표
→ argument에 target_return과 er, 그리고 covariance matrix를 넣어줌.

```
import numpy as np

def minimize_vol(target_return, er, cov):
    """
    target_ret -> W
    """
    n=er.shape[0] # er의 행. 즉 asset의 개수를 의미함.
    init_guess=np.repeat(1/n,n) #처음의 시작값들을 1/n으로하는 n개의 배열 생성
    bounds=((0.0 , 1.0),)*n #w 값에 constraint를 걸어줌. 0~1까지 값 가지는 tuple n개 만들어줌
    return_is_target= {
        'type': 'eq', #equal
        'args':(er,),
        'fun' : lambda weights, er: target_return- erk.portfolio_return(weights,er)
    }
    weights_sum_to_1:{ #constraint
        'type': 'eq',
        'fun': lambda weights: np.sum(weights)-1
    }
    results=minimize(erk.prototype_vol, init_guess,
                    args=(cov,), method="SLSQP",
                    options={'disp': False},
                    constraints=(return_is_target, weights_sum_to_1),
                    bounds=bounds)
    return results.x
```

return_is_target 과 weight_sum_to_1은 weight값들이 들어왔을때 constraint를 걸어줌.

'fun' → w들로 계산한 return과 target_return의 차이가 없어야함을 나타내는 function

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>
*scipy.optimize에 minimize가 built in 되어 있기 때문에 사용하였음.

```
[2]: from scipy.optimize import minimize
```

<결과 적용>

```
: l = ["Smoke", "Fin", "Games", "Coal"]
plot_ef(25, er[l], cov.loc[l,l])
```

<matplotlib.axes._subplots.AxesSubplot at 0x1a1bfae1d0>

