금융공학 및 위험관리
투자전략 수립 프로젝트

2021. 10. 25

Team 1

이상엽, 이종현, 허홍

Contents

Risk-Parity portfolio

위험의 분산을 목적으로 하는 투자 전략

기존 mean-variance 포트폴리오의 가장 큰 단점인 expected return을 넣지 않음

또한 블랙 리터만 모형에서 요구하는 주관적인 전망이 반영되지 않아 상대적으로 간단

본 프로젝트에서 RP 포트폴리오 기반의 펀드를 구성하고 이 성능을 검증하고자 함

Risk-Parity portfolio

$$w_i = \frac{1/\sigma_i}{\sum_{n=1}^{N} 1/\sigma_i}$$

RP 포트폴리오 연구 동향

1. Naïve RP 모형[1]: 변동성의 역수를 포트폴리오 가중치로 사용
2. Equal risk contribution RP 모형[2]: 종목별 위험의 기여도를 동일하게 분배
3. Hierarchical RP 모형[3]: 유사한 종목 cluster 형성 후, cluster 내 위험 재분배
4. ML/DL 기반 공분산 행렬 예측[4]: 기존 공분산 행렬을 예측 모델의 결과로 대체

1 Qian, E. E. (2005). On the financial interpretation of risk contribution: Risk budgets do add up. *Available at SSRN 684221*.
2 Chaves, D., Hsu, J., Li, F., & Shakernia, O. (2011). Risk parity portfolio vs. other asset allocation heuristic portfolios. *The Journal of Investing*, *20*(1), 108-118.
3 De Prado, M. L. (2016). Building diversified portfolios that outperform out of sample. *The Journal of Portfolio Management*, *42*(4), 59-69.
4 김영훈, 최흥식, & 김선웅. (2020). XGBoost 를 활용한 리스크패리티 자산배분 모형에 관한 연구. *지능정보연구*, *26*(1), 135-149.

Risk-Parity portfolio

구현 모델: Equal risk contribution RP 모형

선정 이유: RP 모형의 핵심을 잘 표현하며 구현이 상대적으로 간단하다고 판단

$$\boldsymbol{\sigma_P(W) = \sqrt{W^T \Sigma W}}$$

$$w_i \frac{\partial \sigma_P}{\partial w_i} = w_j \frac{\partial \sigma_P}{\partial w_j} = \frac{\sigma_P}{n}, \forall i, j \qquad s.\ t.\ \sum_{i=1}^{n} w_i = 1, \forall w_i \geq 0$$

Risk-Parity portfolio

구현 모델: Equal risk contribution RP 모형

$$\sigma_P(\mathbf{W}) = \sqrt{\mathbf{W}^T \boldsymbol{\Sigma} \mathbf{W}}$$

$$w_i \frac{\partial \sigma_P}{\partial w_i} = w_j \frac{\partial \sigma_P}{\partial w_j} = \frac{\sigma_P}{n}, \forall i, j \qquad s.\,t. \sum_{i=1}^{n} w_i = 1, \forall w_i \geq 0$$

오일러 정리를 이용해 아래와 같이 정리하면,

$$\sigma_P(w) = \sum_{i=1}^{n} w_i \frac{\partial \sigma_P}{\partial w_i}$$

$$= w_1 \frac{\partial \sigma_P}{\partial w_1} + w_2 \frac{\partial \sigma_P}{\partial w_2} + \cdots + w_n \frac{\partial \sigma_P}{\partial w_n}$$

공분산 행렬을 이용하면,

$$\sigma_P(w) = \sum_{i=1}^{n} w_i \frac{(\boldsymbol{\Sigma}\mathbf{W})_i}{\sqrt{\mathbf{W}^T \boldsymbol{\Sigma} \mathbf{W}}}$$

Risk-Parity portfolio

```
from scipy.optimize import minimize
import matplotlib.pyplot as plt

class RiskParity:
    def __init__(self, df):

        self.df = df
        self.firm_list = self.df.columns.values
        self.cov_mat = df.cov().values
        self.std = df.std()

        self.initial_weights = np.repeat(1 / self.cov_mat.shape[0], self.cov_mat.shape[0])
        self.constraints = ({"type": "eq", "fun": RiskParity.weight_summation_constraint,
                             "type": "ineq", "fun": RiskParity.weight_bound_constraint})
        self.options = {"ftol": 1e-20, "maxiter": 1000}


    def optimize(self):
        self.results = minimize(fun=RiskParity.objective_function,
                                x0=self.initial_weights,
                                method='SLSQP',
                                constraints=self.constraints,
                                options=self.options,
                                args=self.cov_mat)
```

Risk-Parity portfolio

Initial weights. Uniform distribution. N x 1

```
@staticmethod
def objective_function(x, cov_mat):
    variance = x.T @ cov_mat @ x
    sigma = variance ** 0.5
    mrc = 1 / sigma * (cov_mat @ x)
    rc = x * mrc
    a = np.reshape(rc, (len(rc), 1))
    risk_diff = a - a.T
    squared_risk_diff_summation = np.sum(np.square(np.ravel(risk_diff)))

    return squared_risk_diff_summation


@staticmethod
def weight_summation_constraint(x):
    return x.sum() - 1.0


@staticmethod
def weight_bound_constraint(x):
    return x
```

$$\sigma_P(\mathbf{W}) = \sqrt{\mathbf{W}^T \boldsymbol{\Sigma} \mathbf{W}}$$

$\frac{\sigma_P}{n}$ Marginal risk contribution. N x 1

N x 1

Broadcasting 에 의해 N x N

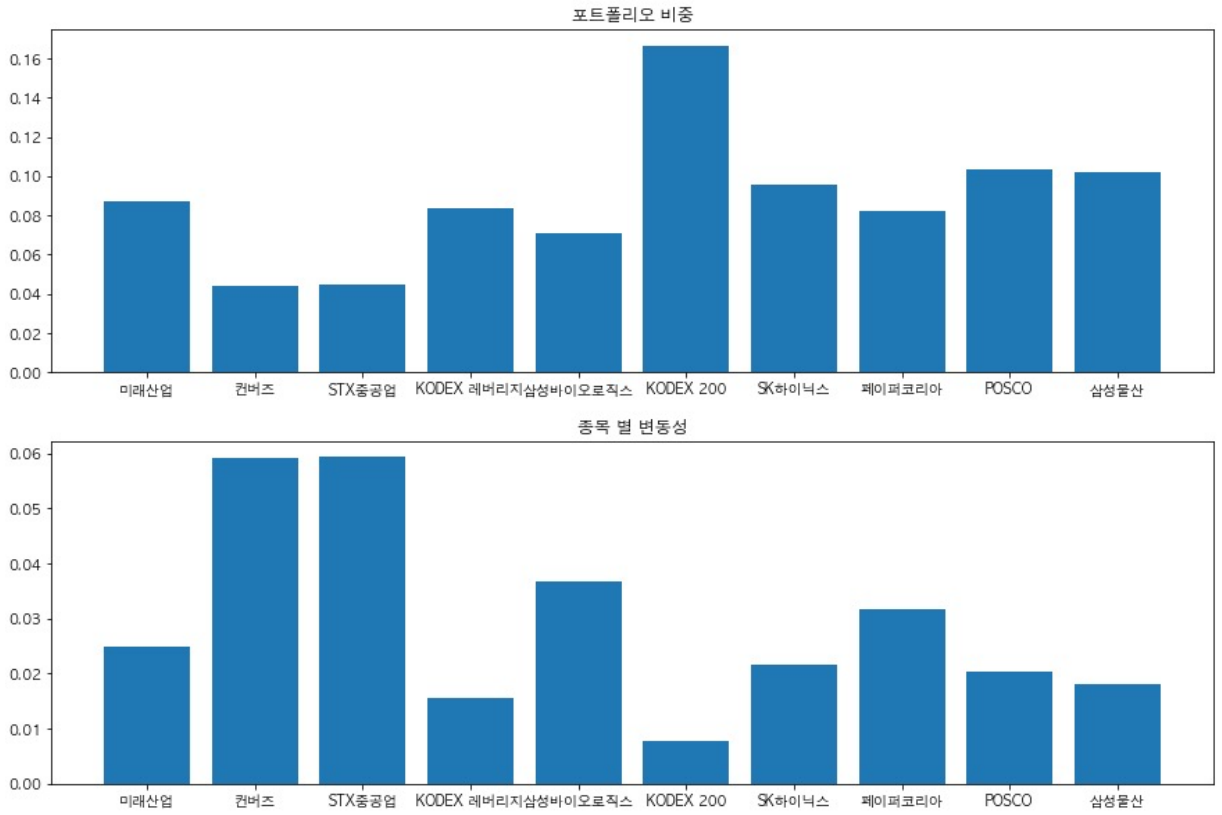
$$J(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} (w_i (V * w)_i - w_j (V * w)_j)^2$$

$$s.\ t.\ \sum_{i=1}^{n} w_i = 1, \forall w_i \geq 0$$

http://henryquant.blogspot.com/2017/08/risk-parity-python-code.html

Risk-Parity portfolio

Name	미래산업	컨버즈	STX중공업	KODEX 레버리지	삼성바이
Name					
미래산업	0.000625	0.000090	0.000123	0.000088	0
컨버즈	0.000090	0.003491	0.000042	0.000104	0
STX중공업	0.000123	0.000042	0.003517	0.000063	0
KODEX 레버리지	0.000088	0.000104	0.000063	0.000244	0
삼성바이오로직스	0.000009	0.000137	0.000063	0.000069	0
KODEX 200	0.000044	0.000052	0.000032	0.000122	0
SK하이닉스	0.000048	0.000027	-0.000008	0.000176	0.000035	0.000088	0.000473	0.000023	0.000058	0.000066
페이퍼코리아	0.000071	0.000138	0.000034	0.000054	0.000013	0.000027	0.000023	0.001007	0.000014	0.000019
POSCO	0.000107	-0.000031	0.000138	0.000134	-0.000047	0.000068	0.000058	0.000014	0.000416	0.000058
삼성물산	0.000050	0.000081	0.000033	0.000143	0.000138	0.000072	0.000066	0.000019	0.000058	0.000328



Risk-Parity portfolio

구현 모델: Hierarchical RP 모형

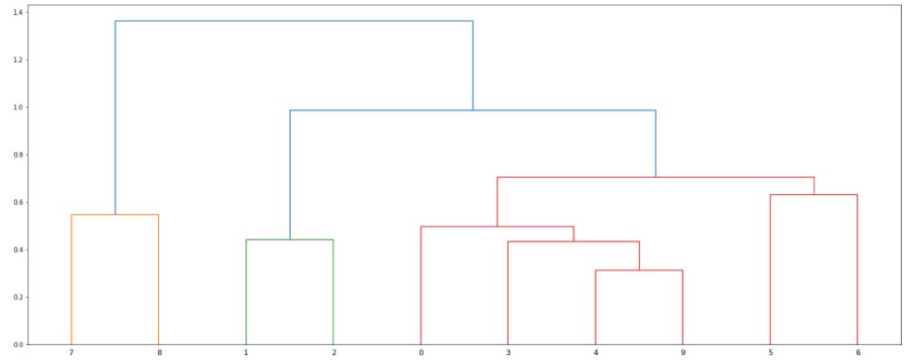
자산 간의 상관관계를 거리로 변형

이 거리가 가까운 종목을 동일 클러스터로. hierarchical clustering 방법을 사용

$$D(i,j) = \sqrt{0.5 * (1-\rho(i,j))}$$

```
In [11]: link = linkage(distance_corr, 'single')
         Z = pd.DataFrame(link)

         fig = plt.figure(figsize=(25, 10))
         dn = dendrogram(Z)
         plt.show()
```

```
/var/folders/nw/x1lqw0rx1mj698bqcgqxhk880000gn/T/ipykernel_2390/2333516210.py:1: ClusterWarning: scipy.cluster: The s
ymmetric non-negative hollow observation matrix looks suspiciously like an uncondensed distance matrix
  link = linkage(distance_corr, 'single')
```



Risk-Parity portfolio

구현 모델: Hierarchical RP 모형

유사한 자산을 Quasi-diagonalization 을 이용하여 가까이 위치하도록 재배치

기존 공분산 행렬에서 공분산이 큰 자산끼리 대각으로 모이도록 함

```
#------------------------------------------------------------------------------
def getQuasiDiag(link):
    # Sort clustered items by distance
    link=link.astype(int)
    sortIx=pd.Series([link[-1,0],link[-1,1]])
    numItems=link[-1,3] # number of original items
    while sortIx.max()>=numItems:
        sortIx.index=range(0,sortIx.shape[0]*2,2) # make space
        df0=sortIx[sortIx>=numItems] #find clusters
        i=df0.index;j=df0.values-numItems
        sortIx[i]=link[j,0] # item 1
        df0=pd.Series(link[j,1],index=i+1)
        sortIx=sortIx.append(df0) # item 2
        sortIx=sortIx.sort_index() # re-sort
        sortIx.index=range(sortIx.shape[0]) # re-index
    return sortIx.tolist()
```

Code snippet 1 – Quasi-diagonalization

Risk-Parity portfolio

구현 모델: Hierarchical RP 모형

상위 클러스터에서 하위 클러스터로 내려가는 반복을 통해 weight 업데이트

Binary tree 라는 특징을 이용하여, 두 child cluster의 risk 계산

앞서 quasi diagonalization 을 적용했기 때문에 대각 행렬에 가깝다고 가정

--> 이 경우, 최적 해는 inverse variance allocation 으로 주어짐

$$V_{adj} = w^T V w$$

$$min \frac{1}{2} w^T \sigma w$$

$$w = \frac{diag[V]^{-1}}{trace(diag[V]^{-1})}$$

$$s.t. e^T w = 1; e = 1^T$$

Risk-Parity portfolio

구현 모델: Hierarchical RP 모형

상위 클러스터에서 하위 클러스터로 내려가는 반복을 통해 weight 업데이트

Binary tree 라는 특징을 이용하여, 두 child cluster의 risk 계산

$$w = \frac{1/\sigma_1}{1/\sigma_1 + 1/\sigma 2} = 1 - \frac{\sigma_1}{\sigma_1 + \sigma_2}$$

$$\alpha_1 = 1 - \frac{V_1}{V_1 + V_2}; \alpha_2 = 1 - \alpha_1$$

$$W_1 = \alpha_1 * W_1$$

$$W_2 = \alpha_2 * W_2$$

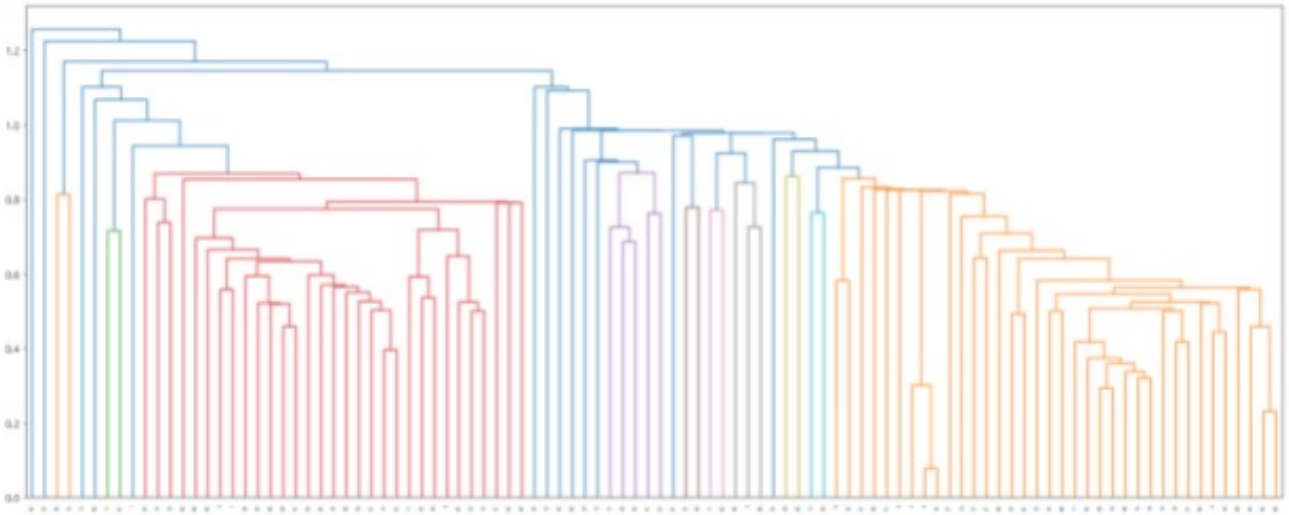
HRP portfolio

```
class HRP:
    def __init__(self, df, num_asset):
        self.df = df
        self.cov_mat = self.df.cov()
        self.num_asset = num_asset


    def optimize(self):
        self.calc_distance()
        self.generate_cluster()
        self.get_quasi_diag()
        self.get_rec_bipart()

    @property
    def weights(self):
        return self.weights
```

HRP portfolio

$$D(i,j) = \sqrt{0.5 * (1 - \rho(i,j))}$$

```
def calc_distance(self):
    corr_mat = df.corr()
    self.distance_mat = np.sqrt(0.5 * (1 - corr_mat))


def generate_cluster(self):
    self.link = linkage(self.distance_mat, 'single')
    self.Z = pd.DataFrame(self.link)
```



HRP portfolio

```
def get_quasi_diag(self):
    self.link = self.link.astype(int)
    self.sort_ix = pd.Series([self.link[-1, 0], self.link[-1, 1]])
    num_items = self.link[-1, 3]

    while self.sort_ix.max() >= num_items:
        self.sort_ix.index = range(0, self.sort_ix.shape[0] * 2, 2)

        df0 = self.sort_ix[self.sort_ix >= num_items]

        i = df0.index
        j = df0.values - num_items

        self.sort_ix[i] = self.link[j, 0]

        df0 = pd.Series(self.link[j, 1], index=i + 1)

        self.sort_ix = self.sort_ix.append(df0)
        self.sort_ix = self.sort_ix.sort_index()

        self.sort_ix.index = range(self.sort_ix.shape[0])
```

```
#------------------------------------------------------------------------------
def getQuasiDiag(link):
   # Sort clustered items by distance
   link=link.astype(int)
   sortIx=pd.Series([link[-1,0],link[-1,1]])
   numItems=link[-1,3] # number of original items
   while sortIx.max()>=numItems:
      sortIx.index=range(0,sortIx.shape[0]*2,2) # make space
      df0=sortIx[sortIx>=numItems] # find clusters
      i=df0.index;j=df0.values-numItems
      sortIx[i]=link[j,0] # item 1
      df0=pd.Series(link[j,1],index=i+1)
      sortIx=sortIx.append(df0) # item 2
      sortIx=sortIx.sort_index() # re-sort
      sortIx.index=range(sortIx.shape[0]) # re-index
   return sortIx.tolist()
```

Code snippet 1 – Quasi-diagonalization

HRP portfolio

```
def get_rec_bipart(self):
    weights = pd.Series(1, index=self.sort_ix)
    c_items = [self.sort_ix]

    while len(c_items) > 0:
        c_items = [i[int(j):int(k)] for i in c_items for j,k in
                   ((0, len(i)/2),(len(i)/2, len(i))) if len(i) > 1]

        for i in range(0, len(c_items), 2):
            c_items0 = c_items[i]
            c_items1 = c_items[i + 1]

            cluster_var0 = self.get_cluster_var(c_items0)
            cluster_var1 = self.get_cluster_var(c_items1)

            alpha = 1 - cluster_var0 / (cluster_var0 + cluster_var1)

            weights[c_items0] *= alpha
            weights[c_items1] *= 1 - alpha

    self.weigths = weights


def get_cluster_var(self, c_items):
    cov_mat_ = self.cov_mat.iloc[c_items, c_items]

    ivp = 1./ np.diag(cov_mat_)
    ivp /= ivp.sum()

    w_ = ivp.reshape(-1,1)
    cluster_var = np.dot(np.dot(w_.T, cov_mat_), w_)[0,0]

    return cluster_var
```

```
#-------------------------------------------------------------------------
def getRecBipart(cov,sortIx):
   # Compute HRP alloc
   w=pd.Series(1,index=sortIx)
   cItems=[sortIx] # initialize all items in one cluster
   while len(cItems)>0:
      cItems=[i[j:k] for i in cItems for j,k in ((0,len(i)/2), \
         (len(i)/2,len(i))) if len(i)>1] # bi-section
      for i in xrange(0,len(cItems),2): # parse in pairs
         cItems0=cItems[i] # cluster 1
         cItems1=cItems[i+1] # cluster 2
         cVar0=getClusterVar(cov,cItems0)
         cVar1=getClusterVar(cov,cItems1)
         alpha=1-cVar0/(cVar0+cVar1)
         w[cItems0]*=alpha # weight 1
         w[cItems1]*=1-alpha # weight 2
   return w
```

Code snippet 2 – Recursive bisection

투자 전략

대상 시장: 한국 KOSPI 상장 기업 전체 (상장 폐지된 종목 포함), 인버스 ETF 제외

대상 기간: 2011.06 ~ 2021.06 (총 120 개월)

분석 도구: Python 3.8

데이터 수집: FinanceDataReader 라이브러리 활용

Out[1]:

	Symbol	Market	Name	Sector	Industry	ListingDate	SettleMonth	Representative	HomePage	Region
1	095570	KOSPI	AJ네트웍스	산업용 기계 및 장비 임대업	렌탈(파렛트, OA장비, 건설장비)	2015-08-21	12월	박대현	http://www.ajnet.co.kr	서울특별시
2	006840	KOSPI	AK홀딩스	기타 금융업	지주사업	1999-08-11	12월	채형석, 이석주(각자 대표이사)	http://www.aekyunggroup.co.kr	서울특별시
6	152100	KOSPI	ARIRANG 200	NaN	NaN	NaT	NaN	NaN	NaN	NaN
7	295820	KOSPI	ARIRANG 200동일가중	NaN	NaN	NaT	NaN	NaN	NaN	NaN
8	253150	KOSPI	ARIRANG 200선물레버리지	NaN	NaN	NaT	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
7321	069260	KOSPI	휴켐스	기타 화학제품 제조업	화합물,화학제품 제조	2002-10-07	12월	신진용	http://www.huchems.com	서울특별시
7325	000540	KOSPI	흥국화재	보험업	손해보험	1974-12-05	12월	권중원	http://www.insurance.co.kr	서울특별시
7326	000547	KOSPI	흥국화재2우B	NaN	NaN	NaT	NaN	NaN	NaN	NaN
7327	000545	KOSPI	흥국화재우	NaN	NaN	NaT	NaN	NaN	NaN	NaN
7328	003280	KOSPI	흥아해운	해상 운송업	외항화물운송업(케미컬탱커)	1976-06-29	12월	이환구	http://www.heung-a.com	서울특별시

투자 전략

포트폴리오 유니버스 구성

- 포트폴리오 구성일 기준, 121 거래일 ~ 1 거래일 전
- 해당 기간 시가 총액 평균치 상위 10 / 30 개 종목 선정. (ETF는 거래대금으로 추정)
- 해당 기간의 수정 종가를 이용하여 종목 간 공분산 행렬 추정

리밸런싱

- 매 21 거래일마다 리밸런싱 실시
- 리밸런싱 시, 종목 선정은 앞선 포트폴리오 유니버스의 기준을 따름

성능 평가

백테스팅을 위한 시장 가정은 단순화

1. 거래 대금에 따른 수수료 없음
2. 개별 종목을 정수 단위로 매수하고 남은 잔금은 그냥 쌓아 놓기
3. 리밸런싱 시 보유 종목 전량 매도 후, 새로운 종목 가중치에 따라 전량 매수

성능 평가

수익률은 log 수익률로 계산하였으며, 3개의 평가 지표를 선정

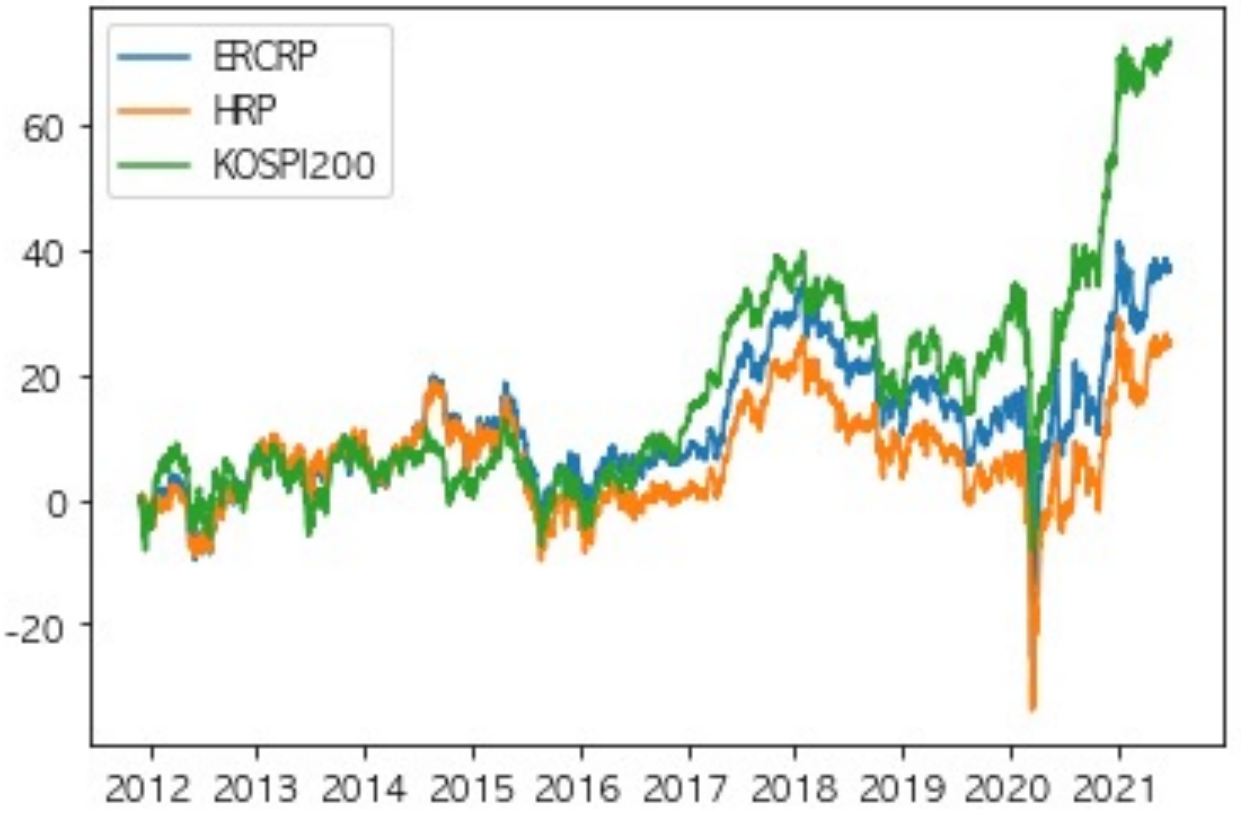
1. 연 환산 수익률
2. 연 환산 위험
3. Sharp ratio (risk free rate 은 가정하지 않음)

성능 평가

평가 기간: 2011-12 ~ 2021-06

선정 종목: 30개

	sharp_ratio	annual_average_return	annual_average_risk
ERCRP	1.67	3.93	2.35
HRP	1.16	2.64	2.29
Benchmark	3.09	7.82	2.53



성능 평가

평가 기간: 2011-12 ~ 2021-06

선정 종목: 10개

	sharp_ratio	annual_average_return	annual_average_risk
ERCRP	1.81	4.87	2.70
HRP	2.22	5.97	2.69
Benchmark	3.09	7.82	2.53

논의

벤치마크 대비 횡보장에서는 상대적으로 유리한 성능

그러나 변동성이 커지는 시장에서는 벤치마크보다 저조한 성능

특히 코로나 이후 상승장에서 성과가 벤치마크에 비해 크게 저조

선정 종목을 줄이면 RP 포트폴리오 전반의 성능이 개선

30 종목에서 ERCRP가 HRP 보다 우수한 성능을 보였으나, 10 종목에서는 HRP가 전반적 우세

논의

추후 발전 계획

ML/DL 기반 공분산 추정 방식을 이용한 RP 포트폴리오 개선 방안 연구

포트폴리오 유니버스 포함 종목 선정 기준의 다변화 실험

Q & A

Project Github

https://github.com/jonghyunlee1993/RP_portfolio_team_01

Thank you!