BofA Final Report

Ajay Dugar, Meng Tian, and Qiaomin Wang

7/13/2021

Introduction

Perhaps all the newcomers to the financial field, when they start to learn portfolio allocation, they will be taught to optimize the portfolio based on the covariance and correlation matrices of the assets. However, simply allocating a portfolio based on variance and correlation has its drawbacks. Variance and correlation are just pure statistics which in many cases cannot mirror the situation of the real world. Correlation matrices lack the notion of hierarchy, which allows weights to vary freely in unintended ways (Lo´pez de Prado [2016]). This is one of the main reasons why modern portfolio optimization techniques often fail to outperform a basic equal-weighted allocation (DeMiguel et al. [2009]).

Recently, many literatures proposed hierarchical clustering-based allocation. These strategies generally explore the hierarchical structure of the assets in the portfolio first. Then they will apply some risk-based allocation based on the hierarchical structure. After sufficient research, we selected three methods for in-depth study. Firstly, Lo´pez de Prado [2016] introduced the Hierarchical Risk Parity (HRP) approach in 2006. This approach properly addresses three major problems in quadratic optimizers: Instability, concentration, and underperformance. However, HRP also has its own drawbacks which we will discuss in this paper. Raffinot [2017, 2018] proposes two hierarchical clustering-based approaches: The Hierarchical Clustering-Based asset allocation (HCAA) and The Hierarchical Equal Risk Contribution Portfolio (HERC). HCAA optimizes portfolios based on the same idea of HRP just in a different way. HERC is a combination of the previous two approaches. The Hierarchical Equal Risk Contribution Portfolio (HERC) merges and enhances the machine learning approach of HCAA and the Top-Down recursive bisection of HRP (Thomas Raffinot [2018]).

Our project aims to show how reasonable and competitive these three methods are by testing them across our selected data and analyzing their out-sample performances. For this project, Ajay Dugar mainly focused on coding HRP. Qiaomin Wang focused on coding HCAA. And Meng Tian is in charge of HERC coding.

In this paper we will introduce the methodologies of these three approaches in the first section. Then we will describe the setup of our empirical implementation. In the third section the performance of empirical implementation is presented and analyzed. Since HERC is the most robust approach theoretically, we did more testing work on this method.

Portfolio Strategies

Hierarchical Risk Parity (HRP) Portfolio

Markowitz's Critical Line Algorithm (CLA) issue for inequality-constrained portfolio optimization is that small changes in expected returns create very different optimal portfolios (Michaud, 1998). Since return forecasts are rarely accurate, a focus on the covariance matrix has been used for risk-based asset allocation approaches. However, this doesn't solve the instability issues, since quadratic programming methods utilize the inversion of the covariance matrix. When positive-definite (all eigenvalues are positive) matrices are inverted, large errors occur when the matrix has a large condition number (ratio between the absolute values of the maximal and minimal eigenvalues) (Bailey and López de Prado, 2012). When we add assets to a portfolio, the condition number increases, more so for correlated investments. At a certain point, the condition number increases to a point that the numerical errors destabilize the inverse matrix, where a small change to the estimations will result in very different inverse matrices. This is described as “Markowitz’s curse”: The more correlated the assets, the greater the requirement for diversification, but the higher the likelihood of unstable solutions for the optimal portfolio. In this regard, the diversification is offset by the errors introduced by estimation in the inversion of the correlation matrix.

To estimate a covariance matrix of size N which is not singular requires $\frac{N(N+1)}{2}$ observations. For $N = 50$, this would require 5 years of daily returns. Through such a long period, the correlation structures are not invariant, and thus, the risk-optimization methodology is flawed. This shortcoming has been demonstrated empirically (De Miguel et al., 2009), where equally-weighted portfolios have produced superior returns to mean-variance and risk-based optimization portfolios out-of-sample.

HRP has 3 steps:

Step 1: Minimum Spanning Tree (MST) - create a hierarchical tree from the correlation matrix using the Single Linkage Clustering Algorithm (SLCA)
Step 2: Quasi-diagonalization - group similar investments by rearranging the covariance matrix
Step 3: Recursive bisection - Optimally distribute weights of the investments using an inverse variance allocation for uncorrelated assets using the Top-Down allocation algorithm.

Tree-Clustering algorithm (Creation of MST)

Step 1: Calculate $N \times N$ correlation matrix with entries $\rho_{i,j}$
Step 2: Convert the distance matrix from the correlation matrix, where $d_{i,j} = \sqrt{\frac{1}{2}(1 - \rho_{i,j})}$
Step 3: Normalize the distance matrix, where $\tilde{d}_{i,j} = \sqrt{\sum_{n=1}^{N}(d_{n,i} - d_{n,j})^2}$

Quasi-Diagonalization

Step 1: Take the distance matrix and group the

Top-Down allocation algorithm (Recursive Bisection)

Step 1: Set all securities weights to 1, $w_i = 1, i \in 1, \ldots, n$
Step 2: Bisect the portfolio into two sets, s_1 and s_2
Step 3: Calculate the covariance matrix for each set, V_1 and V_2
Step 4: Calculate $W_i = diage(V_i)^{-1} * \frac{1}{tr(diag(V_i)^{-1})}$
Step 5: Allow $V_{adj,i} = W_i' * V_i * W_i$
Step 6: Allow $a_1 = 1 - \frac{V_{adj,1}}{V_{adj,1}+V_{adj,2}}$ and $a_2 = 1 - a_1$
Step 7: Adjust the weightings for each set, $W_{s_i,new} = w_{s_i,old} * a_i$

The Hierarchical Clustering-based Asset Allocation(HCAA)

The Hierarchical Clustering-based Asset Allocation is to place assets into groups, suggested by data. Thus, the entity in a given cluster tends to be similar and entities in different clusters tend to be dissimilar (Raffinot, 2017). The objective is to build a binary tree that shows the relationship of groups of assets, which could be further applied to the selection of clusters. Here are the steps of the HCAA method:

Step 1: Hierarchical clustering. Step 2: Selection of the optimal number of clusters based on Gap index. Step 3: Capital allocation across clusters Step 4: Capital allocation within clusters

For the first step, there are four measurements of dissimilarity between two clusters:

a. Simple Linkage. The distance between two clusters is determined by the minimum distance between any two points in the clusters. This method is sensitive to outliers and may cause a problem called chaining;

b. Complete Linkage. The distance between two clusters is determined by the maximum distance between any two points in the clusters. This method is also sensitive to outliers;

c. Average Linkage. The distance between two clusters is determined by the average of the distance between any two points in the clusters;

d. Ward's method. The distance between two clusters is the increase of squared error that results when two clusters are merged. This method is insensitive to noise and outlier; therefore, it is robust as the Average Linkage.

For the second step, according to Tibshirani et al (2001), the Gap index compares the logarithm of the in-cluster distance with the distance of uniformly distributed data, which is not clustering.

Suppose the data is clustered into k clusters, $C_1, C_2, \ldots C_k$. Let $n_r = |C_r|$ and d stand for the squared Euclidean distance $\sum_j (x_{ij} - x_{rj})^2$. Let $D_r = \sum_{i,i' \in C_r} d_{ii'}$ be the sum of the pairwise distance for all points in the cluster r and set $W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r$. By Tibshirani et al (2001), W is the pooled within-cluster sum of squares around the cluster means. The optimal number of clusters is then the value of k which makes the logarithm of the W smallest compared with the benchmark.

For steps 3 and 4, there are many ways for capital allocation:

a. equally weighted allocation. It is very simple but is highly associated with its hierarchical clustering; In addition, there are several methods are based on the risk-budgeting approach. A risk-budgeting portfolio is defined by Roncalli (2013):

$$RC_{w_i} = b_i RC_w$$

$$b_i > 0$$

$$\sum_{i=1}^{N} b_i = 1$$

$$w_i \geq 0$$

$$\sum_{i=1}^{N} w_i = 1$$

Here we have R_w as the volatility as definted as the risk of the portfolio, with $R_w = \sigma_w = \sqrt{w' \Sigma w}$. By Euler decomposition, $R_w = \sum_{i=1}^{N} w_i \frac{\partial R_w}{\partial w_i}$. Then, the risk contribution is defined as $RC_{w_i} = w_i \frac{(\Sigma w)_i}{\sqrt{w' \Sigma w}}$. b_i is the risk budged of each component in the portfolio. The sum of the risk budget is also 1 as defined by the restrictions. W_i is the weight of each componenet in the portfolio

b. minimum-variance allocation. It is similar to the equally weighted allocation:

$$b_i = w_i$$

c. most diversified portfolio allocation. For this method, the risk budget is related to the product of the weight of the asset and its volatility.

$$b_i = \frac{w_i \sigma_i}{\sum_{i=1}^{N} w_i \sigma_i}$$

d. equal-risk-contribution allocation; The risk budget is equal for all assets.

$$b_i = \frac{1}{N}$$

e. inverse volatility risk budget allocation:

$$b_i = \frac{\sigma_i^{-2}}{\sum_{i=1}^{N} \sigma_i^{-2}}$$

These methods contributed to the capital allocation in this framework. They can be used both with and without clustering. By solving this portfolio optimization problem, we can measure their performance and obtain the best way for HCAA.

Hierarchical Equal Risk Contribution(HERC) Portfolio

We just introduce two popular hierarchical clustering-based allocation strategies. However, even though HRP performs better than other traditional allocation strategies, it has its disadvantages. In the first step of HRP, single linkage (SL) has been used to measure the distance between clusters. SL is sensitive to outliers and can result in a problem called chaining, whereby clusters end up being deep and wide. This effect prevents the algorithm from forming any dense clusters. Thus, large weights will be allocated to few assets and consequentially it builds an unbalanced portfolio. The method of generating a hierarchical tree also has drawbacks. In HRP, the hierarchical tree clustering algorithm identifies and segregates the assets into two clusters and repeats this step until each asset becomes an individual cluster. Aditya Vyas (2020) pointed out that this approach has two problems: 1) when dealing with a very large dataset it makes the algorithm computationally slow, and 2) we will face the problem of overfitting. This leads to a situation that even a little inaccuracy in data can result in a huge estimation mistake. Aditya Vyas also mentioned that during recursive bisection, the division does not follow the structure of the dendrogram. Instead, the bisection is based on the number of assets. Finally, there is a very common problem that HRP uses the covariance matrix to estimate risk. There are many studies already pointed out that variance cannot represent the true risk of an asset or portfolio. Also, covariance cannot illustrate the underlying relationship between two assets.

Hence, here we are going to introduce our third, very innovative strategy called the Hierarchical Equal Risk Contribution Portfolio (HERC). This strategy was coming up by Thomas Raffinot in his research study (2018). HERC is a combination of the previous two strategies. It merges and enhances the machine learning approach of HCAA and the Top-Down recursive bisection of HRP. It consists of four main steps:

Step 1: Hierarchical clustering

Step 2: Selection of the optimal number of clusters

Step 3: Top-Down recursive division

Step 4: Naive Risk Parity within clusters

In HERC, Ward linkage is used by default when clustering. Step 2 properly fixes the problem of the hierarchical tree clustering algorithm. Also, HERC modified the recursive bisection to make it base on the dendrogram and follow an Equal Risk Contribution allocation. For the risk metric, Thomas Raffinot tested three metrics: variance (Var), conditional value at risk (CVaR), and Conditional Drawdown at Risk (CDaR). Thomas Raffinot studied the performance of HERC by applying HERC across two disparate datasets (multi-assets and individual stocks). The result of the study can be summarized as:

“The ‘Hierarchical 1/N’ is difficult to beat, but Hierarchical Equal Risk Contribution portfolios based on downside risk measures achieve statistically better risk-adjusted performances, especially those based on the Conditional Drawdown at Risk.”

In the future, we are going to combine these three frameworks together and begin modeling with the data list provided. Accordingly, additional literature will be referred to during the process. At last, we are going to give a promising model that has the best performance.

Data

We will use the list of assets provided by our project host to test our method. As our host said the datasets we’re selected to have the following features:

- be good proxies for most representative asset and sub-asset classes
- to be widely available
- to be as liquid as possible
- to have daily granularity
- to encompass periods with as many market regimes as possibles
- time series have “nicer” statistical properties compared to time series of, say, individual stocks or bonds

Using this criteria, we have found the following indexes (BCOMTR, S5HLTH, LBUSTRUU, S5INDU, RU20INTR, S5INFT, S5COND, S5MATR, S5CONS, S5TELS, S5ENRS, S5UTIL, S5FINL, SPXT). The data chosen spans from 1/1/1990 to 7/16/2021 and consists of daily returns of each of the indexes. The training data includes the data between 1/1/1990 and 1/3/2008, and the testing data includes data between 1/4/2008 and 7/16/2021.

Table 2: Daily data sets

Name	Description	Name	Description
BCOMTR	Bloomberg Commodity Index Total Return	RU20VATR	iShares Russell 2000 Value ETF
HFRIFWI	HFRI Fund Weighted Composite Index	RUMCINTR	iShares Russell Mid-Cap ETF
LBUSTRUU	Bloomberg Barclays US Aggregate Bond Index	RUMRINTR	iShares Micro-Cap ETF
LG30TRUU	Bloomberg Barclays Global High Yield Total Return Index Value Unhedge	RUTPINTR	iShares Russell Top 200 ETF
LMBITR	Bloomberg Barclays Municipal Bond Index Total Return Index Value Unhedged USD	S5COND	S&P 500 Consumer Discretionary Index
NDDUE15X	Amundi MSCI Europe Ex UK Ucits ETF Dr	S5CONS	S&P 500 Consumer Staples Index
NDDUJN	MSCI Japan Index	S5ENRS	S&P 500 Energy Index
NDDUNA	iShares MSCI North America UCITS ETF	S5FINL	S&P 500 Financials Sector GICS Level 1 Index
NDDUPXJ	MSCI Pacific ex Japan UCITS ETF	S5HLTH	S&P 500 Health Care Index
NDDUUK	iShares MSCI UK ETF	S5INDU	S&P 500 Industrials Index
NDDUWXUS	MSCI World ex USA total net return	S5INFT	S&P 500 Information Technology Index
NDUEEGF	SPDR MSCI Emerging Markets UCITS ETF	S5MATR	S&P 500 Materials Index
RU10GRTR	iShares Russell 1000 Growth ETF	S5RLST	S&P 500 Real Estate Index
RU10VATR	iShares Russell 1000 Value ETF	S5TELS	S&P 500 Communication Services Index
RU20GRTR	iShares Russell 2000 Growth ETF	S5UTIL	S&P 500 Utilities Index
RU20INTR	Russell 2000 Total Return	SPXT	Proshares S&P 500 EX Technology ETF

Description of Indexes

Portfolio Metrics

Given the time series of daily out-of-sample returns generated by each strategy in each dataset, several comparison criteria are computed:

1. The adjusted Sharpe ratio (ASR) explicitly adjusts for skewness and kurtosis by incorporating a penalty factor for negative skewness and excess kurtosis:

$$ASR = SR(1 + \frac{\mu_3}{6}SR - \frac{(\mu_4 - 3)}{24}SR^2)$$

Where μ_3 and μ_4 are skewness and kurtosis of the return distributions and SR denotes the traditional Sharpe ratio $SR = \frac{\mu - r_f}{\sigma}$ where r_f is the risk-free rate

2. The certainty-equivalent return (CEQ) is the risk free rate of return that the investor accepted instead of undertaking the strategy.

$$CEQ = (\mu - r_f) - \frac{\gamma}{2}\sigma^2$$

Where γ is the risk aversion. Results are reported for $\gamma = 1$, but other values are also calculated as a robustness check. More precisely, the employed definition of CEQ captures the level of expected utility of a mean-variance investor which is approximately equal to the certainty-equivalent return for an investor with quadratic utility (DeMiguel et al., 2009). It is the most important number to consider for building profitable portfolios (Levy, 2016).

3. The max drawdown (MDD) is an indicator of downside risk over a specified time. The MDD offers investors a worst case scenario.

4. The average turnover per rebalancing (TO):

$$TO = \frac{1}{F} \sum_{i=1}^{F} |w_{i,j} - w_{i,j-1}|$$

5. The sum of squared portfolio weights (SSPW) shows the underlying level of diversification in the portfolio

$$SSPW = \frac{1}{F} \sum_{j=2}^{F} \sum_{i}^{N} w_{i,j}^2$$

Where $SSPW$ ranges from 0 to 1, where 1 represents the most concentrated portfolio.

Empirical Results

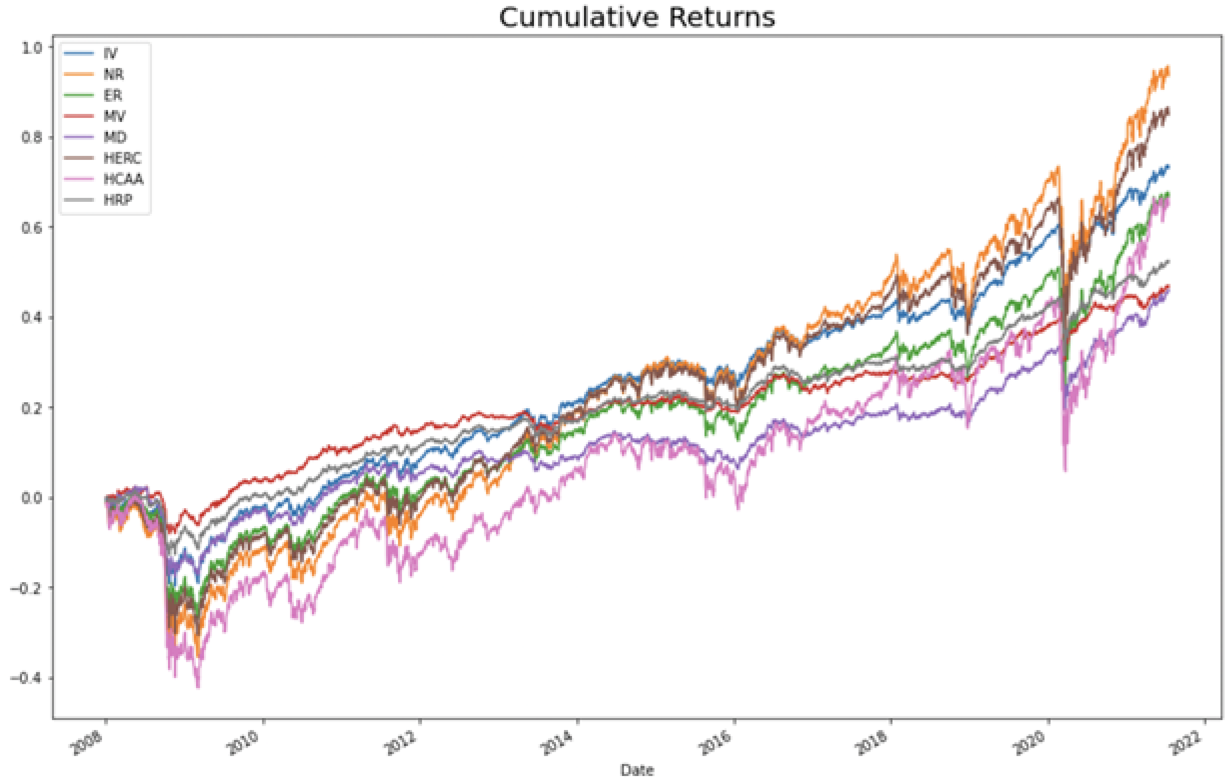
HRP Results

For Hierarchical Risk Parity, we find that the HRP produces superior returns initially, but as the hierarchy of the correlations decay, whether from fundamental changes or decaying of the correlations, the returns of the method tend to decrease. Comparative to the returns of the HERC and HCAA returns, the HRP lags behind both of them.

There are a number of reasons that hinder the returns of HRP, which have been referenced in the literature. First, is that the bisection method creates two equally sized groups, which may result in multiple assets of high correlations being split between the two groups at any given step. Additionally, the the weighing aspect of the top-down approach causes even the inverse volatility (IV) and minimum variance (MV) portfolios to outperform over longer time periods, without a recalculation of the weights. This gives the following results. We trained our method on a data set from 1990 to 2008 and tested it from 2008 to 2021.

	statistics
cum_return	0.113443
ari_mean_return	0.000128
geo_mean_return	0.031333
daily_min_return	-7.557313
drawdown	-0.085185
vol	0.610315
sharpe_ratio	0.000210
skewness	-1.213007
kurtosis	18.639478
modified_VaR	-59.185022
C_VaR	-92.721068

HRP Results



Cumulative Results of All Methods

HCAA Results

For the HCAA, we tested ‘Hierarchical 1/N’ allocation and recursive bisection. Raffinot (2017) applied ‘Hierarchical 1/N’ allocation in his paper. However, this approach didn’t take risk into account which can be problematic in some cases. We want to test HCAA which considers risk as a reference of weighting. Thus, recursive bisection has been tested. We trained our method on a data set from 1990 to 2008 and tested it from 2008 to 2021. Following is the performance table of the two approaches.

	Hierarchical 1/N	Risk-based
cum_return	0.113561	0.122801
ari_mean_return	0.000149	0.000191
geo_mean_return	0.031411	0.037239
daily_min_return	-15.519507	-20.5228
drawdown	-0.200884	-0.25318
vol	1.691491	2.295645
sharpe_ratio	0.000088	0.000083
skewness	-0.823563	-0.70304
kurtosis	11.537765	11.00969
modified_VaR	-172.387646	-231.199
C_VaR	-277.119183	-376.815

HCAA Results

We can see that these two approaches generate closed cumulative returns. Risk-based has a slightly better return. However “Hierarchical 1/N” performs way more better on volatility which means it’s more stable than risk-based approach. Thus “Hierarchical 1/N” even has a higher sharpe ratio.

HERC Results

	equally weighting				variance			
	ward	single	average	complete	ward	single	average	complete
cum_return	0.127187	0.119243	0.110461	0.119243	0.083065	0.099771	0.07576	0.099771
ari_mean_return	0.00022	0.000194	0.000186	0.000194	0.000154	0.00016	0.000128	0.00016
geo_mean_return	0.046144	0.043614	0.040614	0.043614	0.029436	0.036623	0.025826	0.036623
daily_min_return	-18.429907	-12.058874	-13.568505	-12.058874	-13.924844	-8.475977	-11.593216	-8.475977
drawdown	-0.22152	-0.161571	-0.192814	-0.161571	-0.192724	-0.128917	-0.149273	-0.128917
vol	2.091185	1.558404	1.711267	1.558404	2.137915	1.27851	1.753193	1.27851
sharpe_ratio	0.000105	0.000124	0.000109	0.000124	0.000072	0.000125	0.000073	0.000125
skewness	-0.564471	-0.494574	-0.505935	-0.494574	-0.371757	-0.271381	-0.202592	-0.271381
kurtosis	8.190823	5.713909	5.984659	5.713909	3.872849	3.26893	2.863624	3.26893
modified_VaR	-21.264004	-16.109242	-17.705596	-16.109242	-22.333522	-13.127137	-18.025372	-13.127137
C_VaR	-31.510268	-23.200736	-25.468221	-23.200736	-31.868802	-18.617422	-25.680191	-18.617422

	standard_deviation				expected_shortfall			
	ward	single	average	complete	ward	single	average	complete
cum_return	0.083563	0.077124	0.077124	0.098551	0.200916	0.138296	0.141483	0.141483
ari_mean_return	0.000156	0.000132	0.000132	0.000162	0.000288	0.000119	0.000128	0.000128
geo_mean_return	0.029671	0.026526	0.026526	0.03614	0.068286	0.029209	0.031593	0.031593
daily_min_return	-14.182091	-11.803946	-11.803946	-9.44043	-15.136015	-5.566465	-5.903763	-5.903763
drawdown	-0.196091	-0.156859	-0.156859	-0.147752	-0.154232	-0.066089	-0.07025	-0.07025
vol	2.157156	1.787198	1.787198	1.492625	1.359324	0.486899	0.469984	0.469984
sharpe_ratio	0.000072	0.000074	0.000074	0.000109	0.000212	0.000244	0.000273	0.000273
skewness	-0.38435	-0.234981	-0.234981	-0.343229	-1.03936	-1.068582	-1.313825	-1.313825
kurtosis	3.981156	3.003199	3.003199	3.746313	23.613303	10.052296	13.832412	13.832412
modified_VaR	-22.550754	-18.442174	-18.442174	-15.457378	-11.954389	-5.122095	-4.870593	-4.870593
C_VaR	-32.188829	-26.279916	-26.279916	-22.003935	-20.456563	-7.073654	-6.848542	-6.848542

	conditional_drawdown_risk			
	ward	single	average	complete
cum_return	0.193966	0.138523	0.140323	0.140323
ari_mean_return	0.000269	0.00012	0.000125	0.000125
geo_mean_return	0.064602	0.029381	0.030732	0.030732
daily_min_return	-12.760811	-5.300534	-5.653084	-5.653084
drawdown	-0.137563	-0.063928	-0.066755	-0.066755
vol	1.169767	0.505401	0.477051	0.477051
sharpe_ratio	0.00023	0.000237	0.000262	0.000262
skewness	-1.089452	-0.891458	-1.161464	-1.161464
kurtosis	23.660633	8.313536	11.3643	11.3643
modified_VaR	-10.336712	-5.294615	-4.993221	-4.993221
C_VaR	-17.525058	-7.264905	-6.934877	-6.934877

We tested the model using the first 20% data for training, and 80% data for testing. As the results shown above, we find that:

1. Under the risk metric, the ward’s linkage gives a higher cumulative return compared with other linkages. The arithmetic mean return tends to be higher as well. However, the variance and volatility of ward’s are also

higher, which means it is more risky than other strategies. Below is all linkage's performance under conditional drawdown at risk (CDaR) risk metric:

conditional_drawdown_risk				
	ward	single	average	complete
cum_return	0.193966	0.138523	0.140323	0.140323
ari_mean_return	0.000269	0.00012	0.000125	0.000125
geo_mean_return	0.064602	0.029381	0.030732	0.030732
daily_min_return	-12.760811	-5.300534	-5.653084	-5.653084
drawdown	-0.137563	-0.063928	-0.066755	-0.066755
vol	1.169767	0.505401	0.477051	0.477051
sharpe_ratio	0.00023	0.000237	0.000262	0.000262
skewness	-1.089452	-0.891458	-1.161464	-1.161464
kurtosis	23.660633	8.313536	11.3643	11.3643
modified_VaR	-10.336712	-5.294615	-4.993221	-4.993221
C_VaR	-17.525058	-7.264905	-6.934877	-6.934877

2. Under the same linkage, the expected shortfall risk metric has the best performance in terms of cumulative return and Sharpe ratio. While the risk metric of conditional drawdown at risk has a cumulative return of 19.40%, which is very close to the expected shortfall metric. However, the Sharpe ratio of it is significantly smaller than the expected shortfall. Other risk metrics have no significant difference between each other, according to its mean return, Sharpe ratio and VaR. Below is all risk metrics combined with ward's linkage:

	ward				
	equally weighting	variance	standard_deviation	expected_shortfall	conditional_drawdown_risk
cum_return	0.127187	0.083065	0.083563	0.200916	0.193966
ari_mean_return	0.00022	0.000154	0.000156	0.000288	0.000269
geo_mean_return	0.046144	0.029436	0.029671	0.068286	0.064602
daily_min_return	-18.429907	-13.924844	-14.182091	-15.136015	-12.760811
drawdown	-0.22152	-0.192724	-0.196091	-0.154232	-0.137563
vol	2.091185	2.137915	2.157156	1.359324	1.169767
sharpe_ratio	0.000105	0.000072	0.000072	0.000212	0.00023
skewness	-0.564471	-0.371757	-0.38435	-1.03936	-1.089452
kurtosis	8.190823	3.872849	3.981156	23.613303	23.660633
modified_VaR	-21.264004	-22.333522	-22.550754	-11.954389	-10.336712
C_VaR	-31.510268	-31.868802	-32.188829	-20.456563	-17.525058

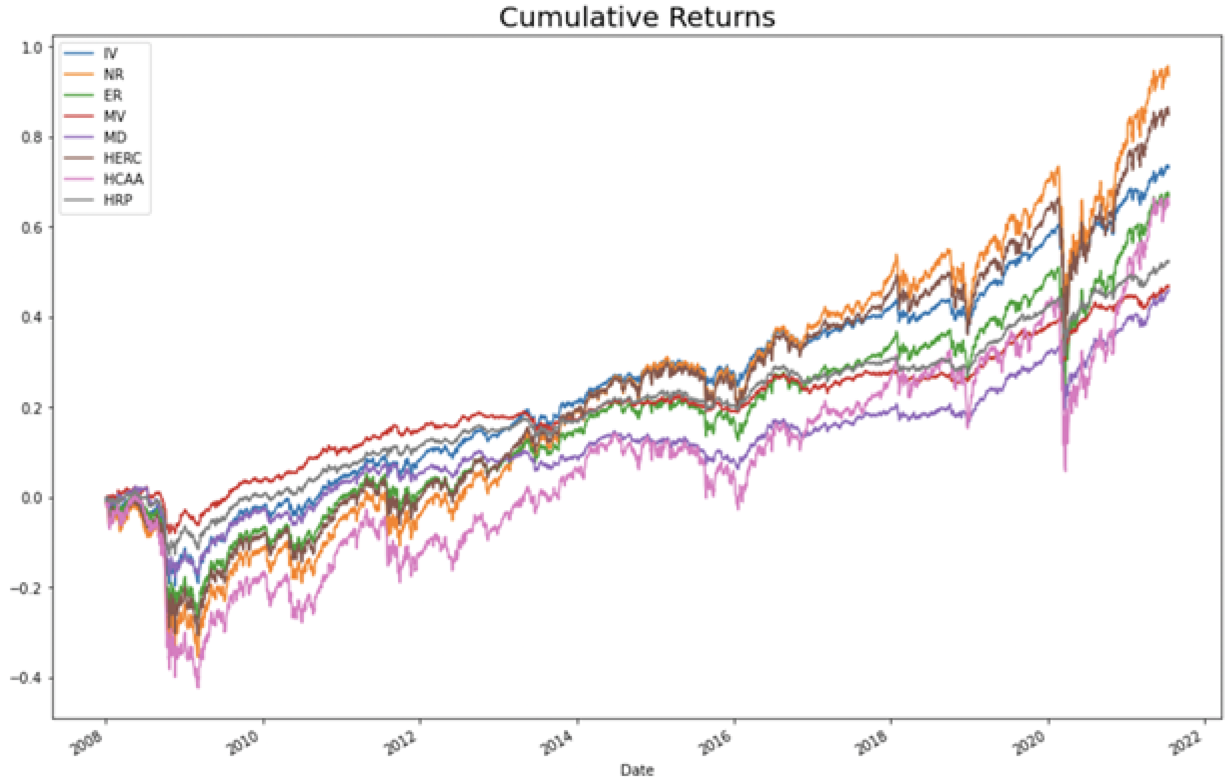
3. Compared with the Buy & Hold (B&H) strategy, we find that the cumulative return of B&H is higher than the HERC with best performance. However, the Sharpe ratio of the HERC is higher than the B&H, and its volatility and VaR is significantly lower than B&H. Thus, the HERC is a competitive strategy. And with more historical data included for training, the weights and clusters could be more reasonable for evaluation.

	expected_shortfall&ward	original
cum_return	0.200916	0.319823
ari_mean_return	0.000288	0.000534
geo_mean_return	0.068286	0.116931
daily_min_return	-15.136015	-32.957905
drawdown	-0.154232	-0.292651
vol	1.359324	2.871143
sharpe_ratio	0.000212	0.000186
skewness	-1.03936	-0.860043
kurtosis	23.613303	22.308889
modified_VaR	-11.954389	-25.035602
C_VaR	-20.456563	-43.671779

We tested the model using 80% data for training, and 20% data for testing. Below are the results, we find that the cumulative return of HERC gets closer to B&H because of a better weight-combination derived from more price data. Meanwhile, the volatility and VaR is still significantly smaller than B&H.

	expected_shortfall&ward	original
cum_return	0.2391	0.3088
ari_mean_return	0.000283	0.000508
geo_mean_return	0.066085	0.106245
daily_min_return	-15.136015	-32.957905
drawdown	-0.154232	-0.292651
vol	1.519115	3.208154
sharpe_ratio	0.000186	0.000158
skewness	-1.087365	-0.898116
kurtosis	22.236072	21.021817
modified_VaR	-13.80094	-28.835766
C_VaR	-23.622895	-50.360081

In conclusion, the HERC has a better chance to make a profit in terms of Sharpe ratio. Although the mean return is lower than the benchmark, it is less risky in terms of volatility and VaR. With more data for training, the HERC gives more reasonable weights combinations based on historical price information.



Cumulative Results of All Methods

Conclusion

From the out-of-sample results, as well as the numerical risk measures calculated, HERC outperforms HCAA and vastly outperforms HRP. However, given the long-term investment horizon some of the hierarchical aspects of these models, the correlation structure may decay over time. Further research is necessary to determine the ideal time period over which these hierarchies are maintained, and whether this differs depending on market conditions, or whether these are a more fundamental aspect of asset types and sectors.

References

- Bailey, D. and M. López de Prado (2012). The sharpe ratio efficient frontier. Journal of Risk, 15(2).
- Michaud,R. (1998). Efficient asset management.
- Raffinot, T. (2017). Hierarchical clustering-based asset allocation.The Journal of Portfolio Management 44(2), pp. 89–99.
- Raffinot, T. (2018). The hierarchical equal risk contribution portfolio. SSRN e-Print.
- Roncalli, T. (2013). Introduction to Risk Parity and Budgeting. Boca Raton, FL: Chapman & Hall.
- Tibshirani et al. (2001). Estimating the number of clusters in a data set via the gap statistic. J.R.Statist.Soc.B.

Appendix

Script of HRP

```r
returns = read.csv("return.csv")
returns[,1] = as.Date(returns[,1])
returns2 = data.frame(lapply(returns[,2:15], function(x) as.numeric(as.character(x))))
returns3 = data.frame(cbind(returns[,1],returns2))
returns3[is.na(returns3)] = 0
returns = returns3
names(returns)[1] = "Date"
rm(returns2)
rm(returns3)
```

```r
train_start = "1990-01-03"
train_end = "2007-07-02"
test_start = "2007-07-03"
test_end = "2021-07-16"

train_data = returns[which(returns$Date==train_start):which(returns$Date==train_end),2:15]
test_data = returns[which(returns$Date==test_start):which(returns$Date==test_end),2:15]

covMat <- cov(train_data)
corMat <- cor(train_data)
```

```r
clustOrder <- hclust(dist(corMat), method = 'single')$order
getIVP <- function(covMat) {
  invDiag <- 1/diag(as.matrix(covMat))
  weights <- invDiag/sum(invDiag)
  return(weights)
}
getClusterVar <- function(covMat, cItems) {
  covMatSlice <- covMat[cItems, cItems]
  weights <- getIVP(covMatSlice)
  cVar <- t(weights) %*% as.matrix(covMatSlice) %*% weights
  return(cVar)
}
getRecBipart <- function(covMat, sortIx) {
  w <- rep(1,ncol(covMat))
  w <- recurFun(w, covMat, sortIx)
  return(w)
}
recurFun <- function(w, covMat, sortIx) {
  subIdx <- 1:trunc(length(sortIx)/2)
  cItems0 <- sortIx[subIdx]
  cItems1 <- sortIx[-subIdx]
  cVar0 <- getClusterVar(covMat, cItems0)
  cVar1 <- getClusterVar(covMat, cItems1)
  alpha <- 1 - cVar0/(cVar0 + cVar1)

  # scoping mechanics using w as a free parameter
  w[cItems0] <- w[cItems0] * alpha
  w[cItems1] <- w[cItems1] * (1-alpha)

  if(length(cItems0) > 1) {
    w <- recurFun(w, covMat, cItems0)
  }
  if(length(cItems1) > 1) {
    w <- recurFun(w, covMat, cItems1)
  }
  return(w)
}
out <- getRecBipart(covMat, clustOrder)
out
```

Script of HCAA

```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


get_ipython().run_line_magic('matplotlib', 'inline')

import pandas as pd
import numpy as np
import fastcluster
from scipy.cluster import hierarchy
import scipy.cluster.hierarchy as spc
import matplotlib.pyplot as plt


# In[2]:


#Setup
df = pd.read_csv('/Users/qiaominwang/Downloads/return.csv')
df = df.set_index('Date')
df = df.set_index(pd.to_datetime(df.index))
df = df.astype('float')
df = df.dropna()
df_train = df.loc['1990-01-03':'2008-01-03',:]
V = df_train.cov()
correl_mat = df_train.corr()


# In[11]:


plt.pcolormesh(correl_mat)
plt.colorbar()
plt.show()


```

```python

#Hierarchical 1/N
pdist = spc.distance.pdist(correl_mat)
dim = len(correl_mat)
linkage = spc.linkage(pdist, method = 'ward')
idx = spc.fcluster(linkage, 0.5 * pdist.max(), 'distance')
clusters = {i: [] for i in range(min(idx),
                                  max(idx) + 1)}

for i, v in enumerate(idx):
    clusters[v].append(i)



# In[14]:


dn = hierarchy.dendrogram(linkage)


# In[208]:


dic = {}
clu = {}
for i in range(len(linkage)-1):
    dic[i+14] = [int(linkage[i][0]), int(linkage[i][1])]
    clu[i+14] = [int(linkage[i][0]), int(linkage[i][1])]

for item in clu:
    i = 0
    while i < len(clu[item]):
        if clu[item][i] > len(V)-1:
            c = clu[item].pop(i)
            clu[item] = clu[item] + dic[c]
            i -= 1
        i += 1

```

```python

for i in clu:
    clu[i].sort()


# In[209]:


w = {int(linkage[len(linkage)-1][1]):0.5, int(linkage[len(linkage)-1][0]):0.5}
i = int(linkage[len(linkage)-1][0]-1)
while i >= 0 :
    w[i] = 0
    i -= 1

i = int(linkage[len(linkage)-1][1])
while i >= len(V) :
    for j in dic[i] :
        w[j] = w[i]/2
    i -= 1


# In[214]:


weights = {}
for item in clusters:
    if len(clusters[item]) > 1 :
        key_list = list(clu.keys())
        val_list = list(clu.values())
        clusters[item].sort()
        position = val_list.index(clusters[item])
        c = key_list[position]
        for i in clusters[item]:
            weights[i] = w[c]/len(clusters[item])
    else :
        weights[clusters[item][0]] = w[clusters[item][0]]


```

```python
weights = pd.DataFrame(weights.items())
weights = weights.set_index(0)


# In[181]:


# Risk-based allocation
dist = 1 - correl_mat
dim = len(dist)
tri_a, tri_b = np.triu_indices(dim, k=1)
X = []

for i in range(len(tri_a)) :
    X.append(dist.iloc[tri_a[i], tri_b[i]])

Z = fastcluster.linkage(X, method='ward')
permutation = hierarchy.leaves_list(
    hierarchy.optimal_leaf_ordering(Z, X))
#ordered_corr = correl_mat[permutation, :][:, permutation]

nb_clusters = 4
clustering_inds = fcluster(Z, 0.5*np.array(X).max(), criterion='distance')
clusters = {i: [] for i in range(min(clustering_inds),
                                 max(clustering_inds) + 1)}
for i, v in enumerate(clustering_inds):
    clusters[v].append(i)

plt.figure(figsize=(8, 8))
plt.pcolormesh(correl_mat)
for cluster_id, cluster in clusters.items():
    xmin, xmax = min(cluster), max(cluster)
    ymin, ymax = min(cluster), max(cluster)

    plt.axvline(x=xmin,
                ymin=ymin / dim, ymax=(ymax + 1) / dim,
                color='r')
    plt.axvline(x=xmax + 1,
```

```python
    ymin, ymax = min(cluster), max(cluster)

    plt.axvline(x=xmin,
                ymin=ymin / dim, ymax=(ymax + 1) / dim,
                color='r')
    plt.axvline(x=xmax + 1,
                ymin=ymin / dim, ymax=(ymax + 1) / dim,
                color='r')
    plt.axhline(y=ymin,
                xmin=xmin / dim, xmax=(xmax + 1) / dim,
                color='r')
    plt.axhline(y=ymax + 1,
                xmin=xmin / dim, xmax=(xmax + 1) / dim,
                color='r')
plt.show()


# In[182]:


for id_cluster, cluster in clusters.items():
    print(id_cluster - 1, ':', cluster)


# In[183]:


def seriation(Z, dim, cur_index):
    if cur_index < dim:
        return [cur_index]
    else:
        left = int(Z[cur_index - dim, 0])
        right = int(Z[cur_index - dim, 1])
        return seriation(Z, dim, left) + seriation(Z, dim, right)

def intersection(lst1, lst2):
    return list(set(lst1) & set(lst2))

```

```python

# In[184]:


def compute_allocation(covar, clusters):
    nb_clusters = len(clusters)
    assets_weights = np.array([1.] * len(covar))
    clusters_weights = np.array([1.] * nb_clusters)
    clusters_var = np.array([0.] * nb_clusters)

    for id_cluster, cluster in clusters.items():
        cluster_covar = covar.iloc[cluster,cluster]
        inv_diag = 1 / np.diag(cluster_covar)
        assets_weights[cluster] = inv_diag / np.sum(inv_diag)

    for id_cluster, cluster in clusters.items():
        weights = assets_weights[cluster]
        clusters_var[id_cluster - 1] = np.dot(
            weights, np.dot(covar.iloc[cluster,cluster], weights))

    for merge in range(nb_clusters - 1):
        print('id merge:', merge)
        left = int(Z[dim - 2 - merge, 0])
        right = int(Z[dim - 2 - merge, 1])
        left_cluster = seriation(Z, dim, left)
        right_cluster = seriation(Z, dim, right)

        print(len(left_cluster),
              len(right_cluster))

        ids_left_cluster = []
        ids_right_cluster = []
        for id_cluster, cluster in clusters.items():
            if sorted(intersection(left_cluster, cluster)) == sorted(cluster):
                ids_left_cluster.append(id_cluster)
            if sorted(intersection(right_cluster, cluster)) == sorted(cluster):
                ids_right_cluster.append(id_cluster)

```

```python
        ids_left_cluster = np.array(ids_left_cluster) - 1
        ids_right_cluster = np.array(ids_right_cluster) - 1
        print(ids_left_cluster)
        print(ids_right_cluster)
        print()

        ids_left_cluster = ids_left_cluster.astype('int')
        ids_right_cluster = ids_right_cluster.astype('int')
        alpha = 0
        left_cluster_var = np.sum(clusters_var[ids_left_cluster])
        right_cluster_var = np.sum(clusters_var[ids_right_cluster])
        alpha = left_cluster_var / (left_cluster_var + right_cluster_var)

        clusters_weights[ids_left_cluster] = clusters_weights[
            ids_left_cluster] * alpha
        clusters_weights[ids_right_cluster] = clusters_weights[
            ids_right_cluster] * (1 - alpha)

    for id_cluster, cluster in clusters.items():
        assets_weights[cluster] = assets_weights[cluster] * clusters_weights[
            id_cluster - 1]

    return assets_weights


# In[185]:


weights = compute_allocation(V, clusters)


```

Script of HERC

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 15 17:32:39 2021

@author: Meng Tian
"""
# from hcaa import HierarchicalClusteringAssetAllocation
# from testHCAA import TestHCAA


import os
import numpy as np
import pandas as pd
from portfoliolab.clustering import HierarchicalEqualRiskContribution
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from scipy.stats import kurtosis, skew,norm

project_path = os.path.dirname(__file__)
data_path = project_path + '\Price.csv'
data = pd.read_csv(data_path, parse_dates=True, index_col="Date")

line = int(len(data)*0.2)
train_data = data[:line]
test_data = data[line+1:]
```

```
# In[1]:

# reading in our data
stock_prices = train_data.sort_values(by='Date')
stock_prices.resample('M').last().plot(figsize=(17,7))
# plt.ylabel('Price', size=15)
# plt.xlabel('Dates', size=15)
# plt.title('Asset Prices Overview', size=15)
# plt.show()
```

```
# In[2]:

herc = HierarchicalEqualRiskContribution()
herc.allocate(asset_names=stock_prices.columns,
              asset_prices=stock_prices,
              risk_measure="expected_shortfall",
              linkage="ward")
# plotting our optimal portfolio
herc_weights = herc.weights
y_pos = np.arange(len(herc_weights.columns))
plt.figure(figsize=(25,7))
plt.bar(list(herc_weights.columns), herc_weights.values[0])
plt.xticks(y_pos, rotation=45, size=10)
plt.xlabel('Assets', size=20)
plt.ylabel('Asset Weights', size=20)
plt.title('HERC Portfolio Weights', size=20)
plt.show()

##### In[3]

plt.figure(figsize=(17,7))
herc.plot_clusters(assets=stock_prices.columns)
plt.title('HERC Dendrogram', size=18)
plt.xticks(rotation=45)
plt.show()
```

```python
# In[4]:

print("Optimal Number of Clusters: " + str(herc.optimal_num_clusters))
```

```python
# In[6]:
#function that compute performance
def compute_performance_table(data_return, principal):
    year_b_day = 250

    #cumulative  return
    cum_return = year_b_day * np.cumprod((data_return + 1)).iloc[-1, 0] / len(data_return)

    #daily mean return
    ari_mean_return = np.mean(data_return)[0]

    #geometric mean return
    geo_mean_return = year_b_day * (np.power(np.cumprod(data_return + 1).iloc[-1 , 0], 1/len(data_return)) - 1)

    #daily min return
    daily_min_return = (year_b_day * np.min(data_return))[0]

    #max 10 days drawdown
    all_cum_return =  np.cumprod((data_return + 1))
    roll_max = all_cum_return.rolling(window = 10).max()
    drawdown = float(np.min(all_cum_return/roll_max - 1)[0])

    #volatility
    vol = (year_b_day * np.std(data_return))[0]

    #sharpe ratio
    sharpe_ratio = ari_mean_return / vol

    #skewness Kurtosis
    skewness = skew(data_return)[0]
    kurtosis_stats = kurtosis(data_return)[0]
```

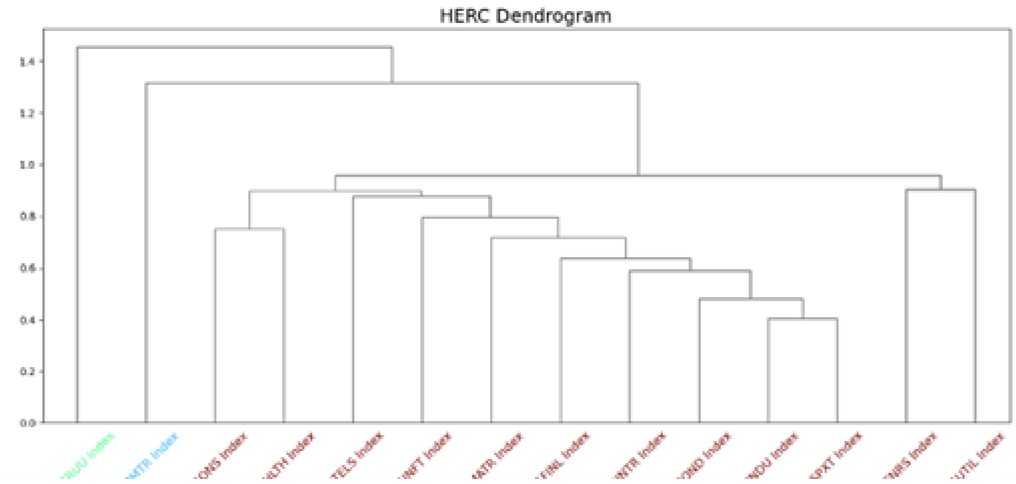
```python
    #modified VaR, CVaR with 95% confidence level
    z = norm.ppf(0.05)
    t = z + 1/6*(z**2 - 1)*skewness + 1/24*(z**3 - 3*z)*kurtosis_stats - 1/36*(2*z**3 - 5*z)*skewness**2
    modified_VaR = principal * (np.mean(data_return) + t*np.std(data_return)) * np.sqrt(year_b_day)
    C_VaR = principal * np.mean(data_return[data_return <= np.quantile(data_return, 0.05)])[0] * np.sqrt(year_b_day)

    result = pd.DataFrame(data = [cum_return, ari_mean_return, geo_mean_return, daily_min_return,
                          drawdown, vol, sharpe_ratio, skewness, kurtosis_stats,
                          modified_VaR[0], C_VaR],
                         index = ['cum_return', 'ari_mean_return', 'geo_mean_return', 'daily_min_return',
                          'drawdown', 'vol', 'sharpe_ratio', 'skewness', 'kurtosis',
                          'modified_VaR', 'C_VaR'],
                          columns = ['statistics'])
    return result

## In[5]:

etf_ret = test_data.pct_change()[1:]
p_return=etf_ret.dot(herc_weights.T)

print(compute_performance_table(p_return, principal=100))

ori_weights = herc_weights
for i in herc_weights.columns:
    ori_weights[i].loc[0]=1/11
```

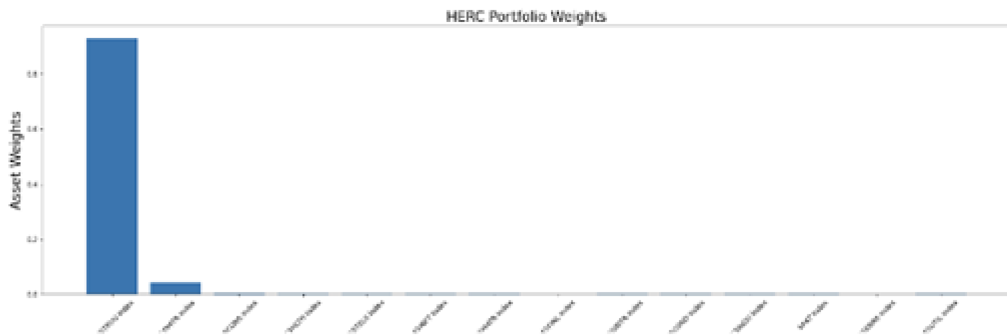
```python
# In[8]:

data_return = etf_ret.dot(ori_weights.T)
ori_cum_return = 250 * np.cumprod((data_return + 1)).iloc[-1, 0] / len(data_return)

print(compute_performance_table(data_return, principal=100))
```

HERC Graphics

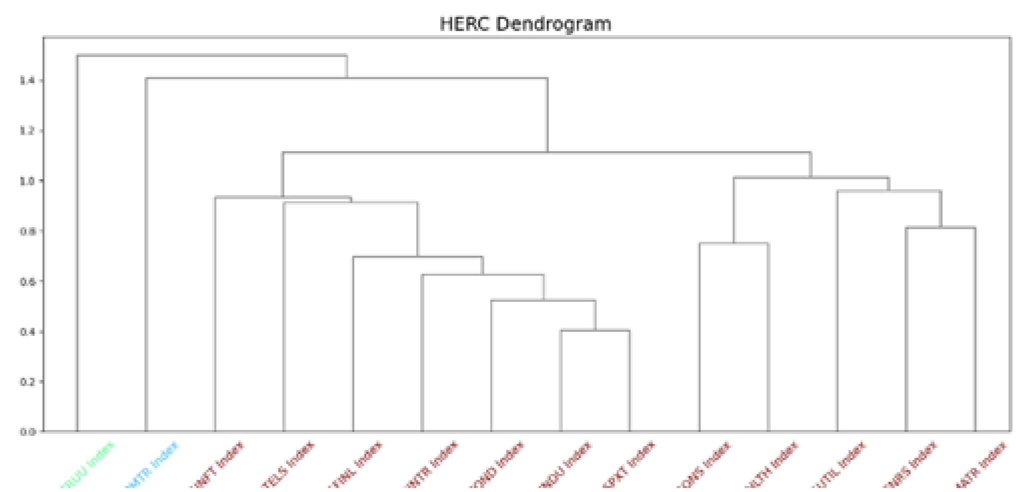
Conditional Drawdown at Risk

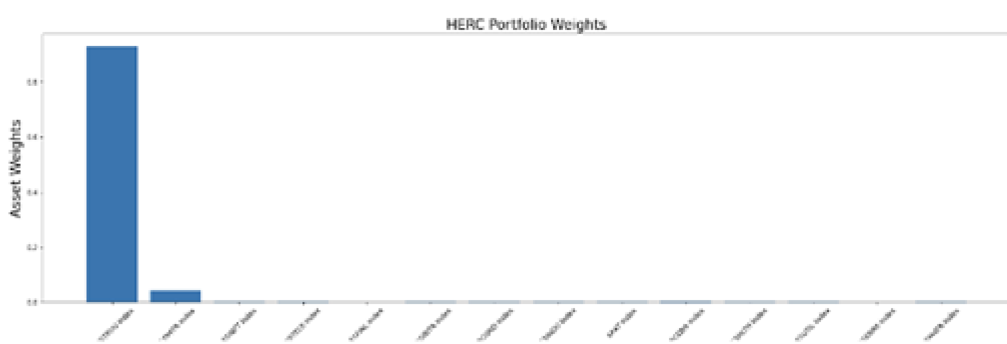
Average



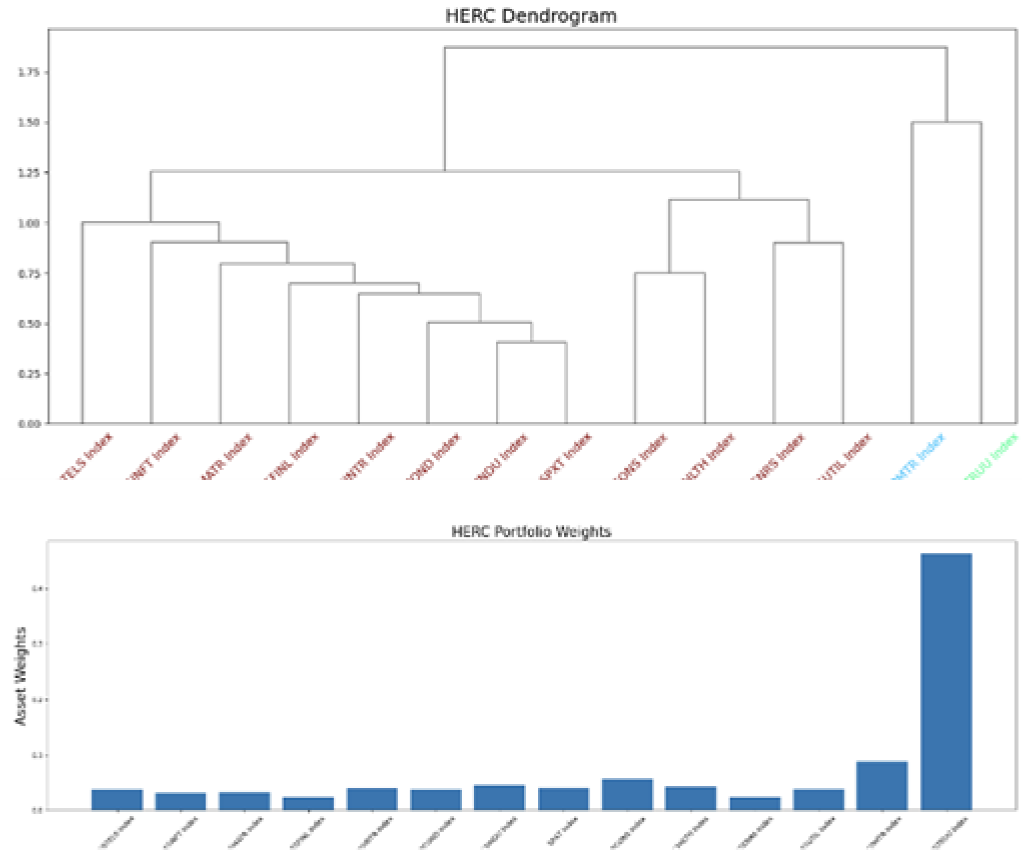


Complete

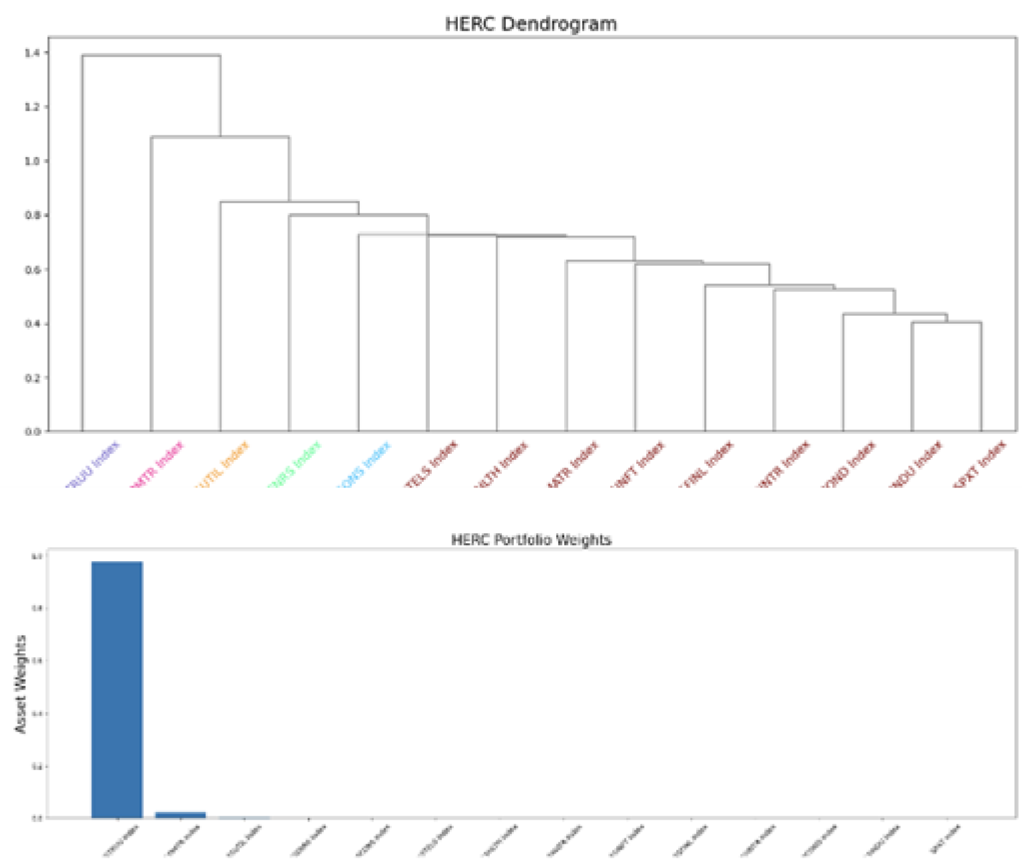




Ward

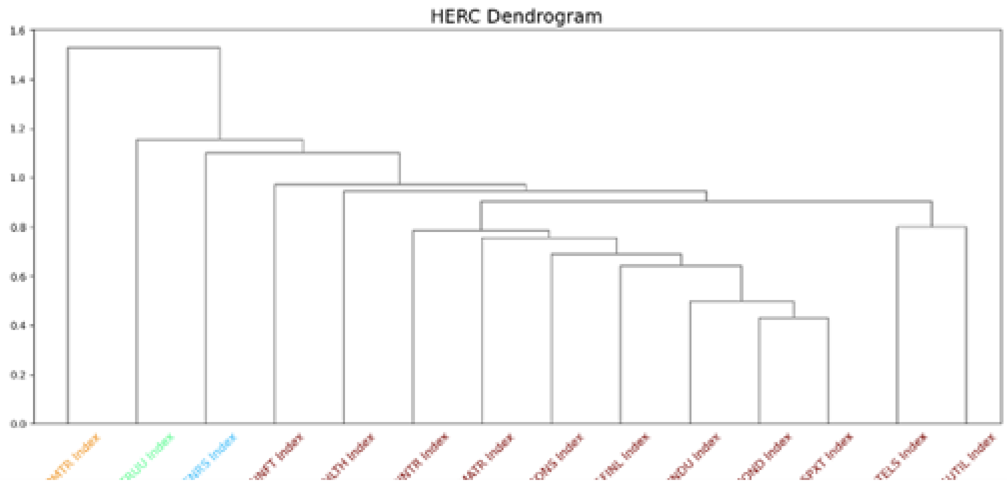


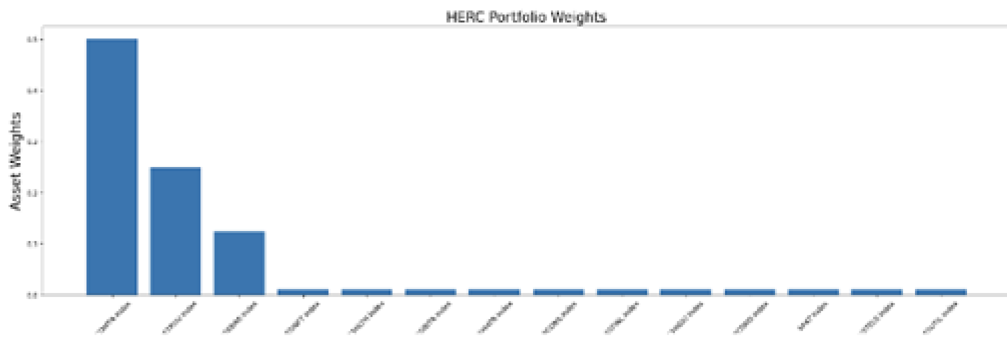
Single



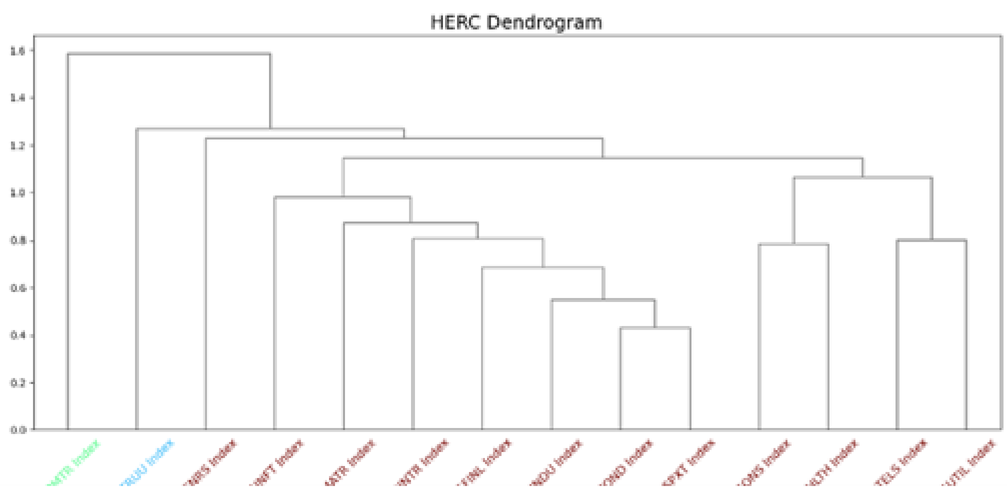
Equally Weighted

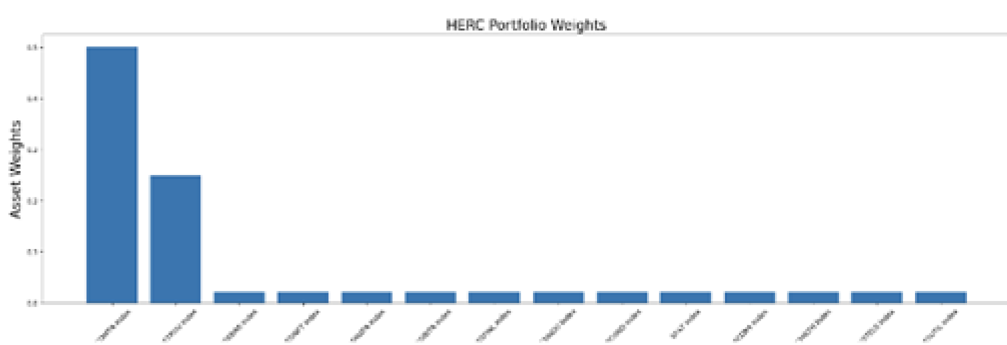
Average



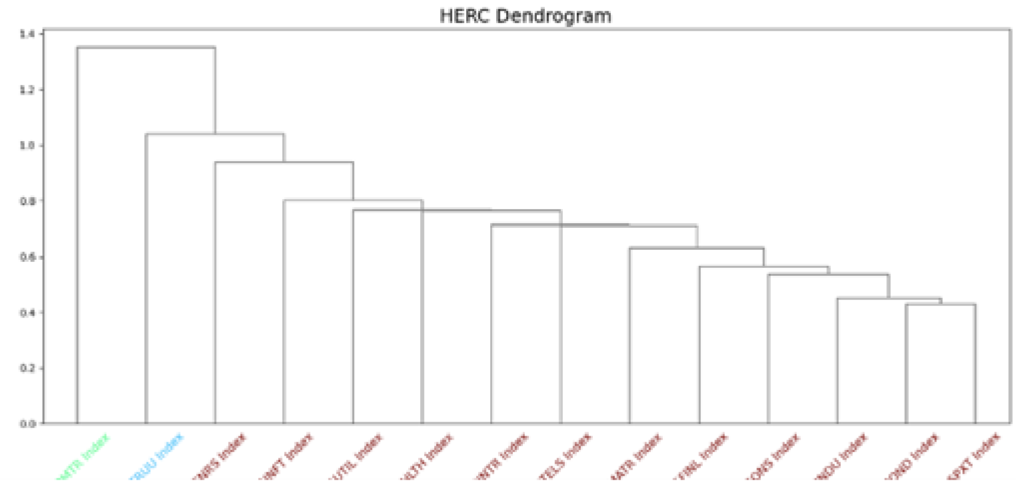


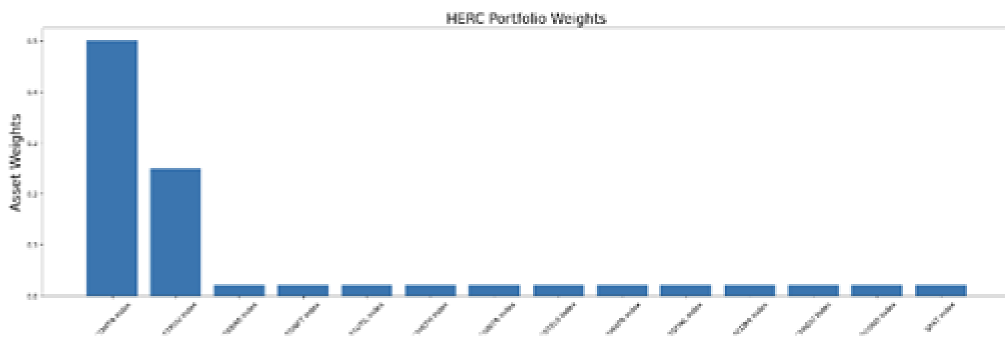
Complete



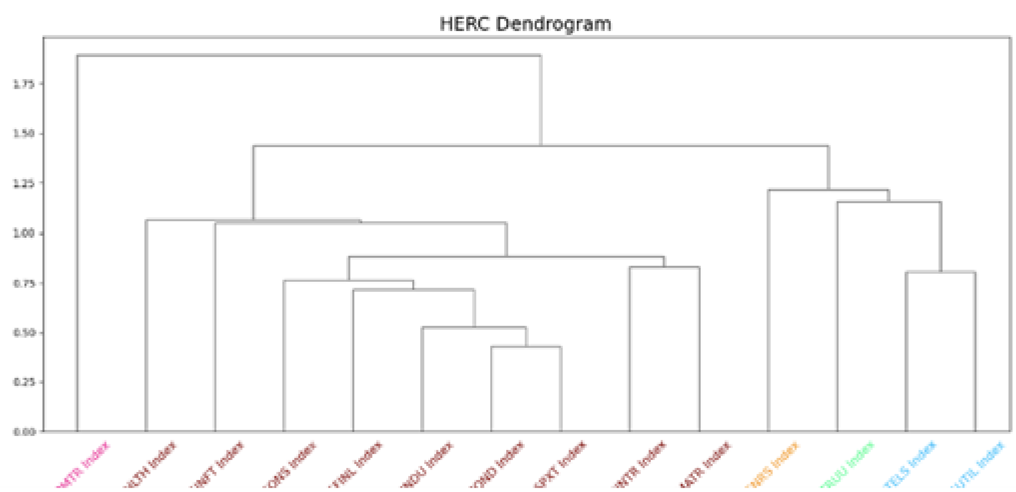


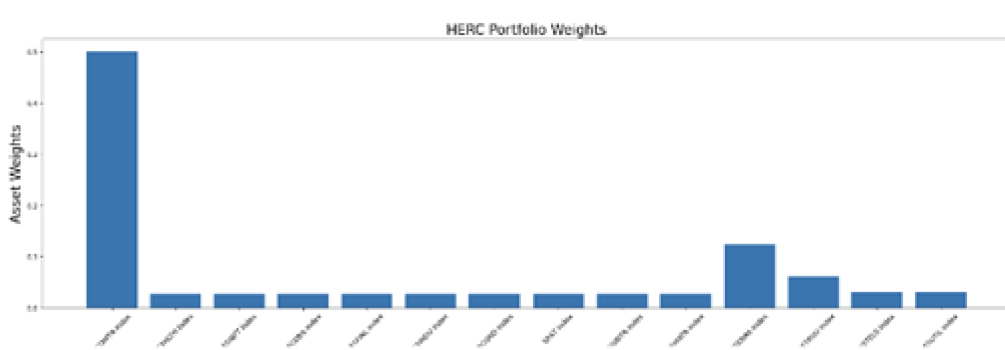
Single





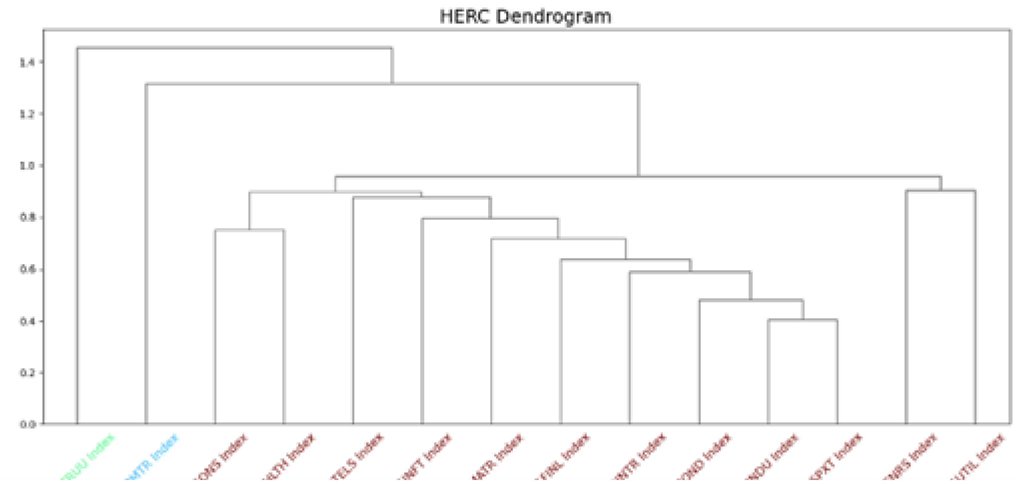
Ward

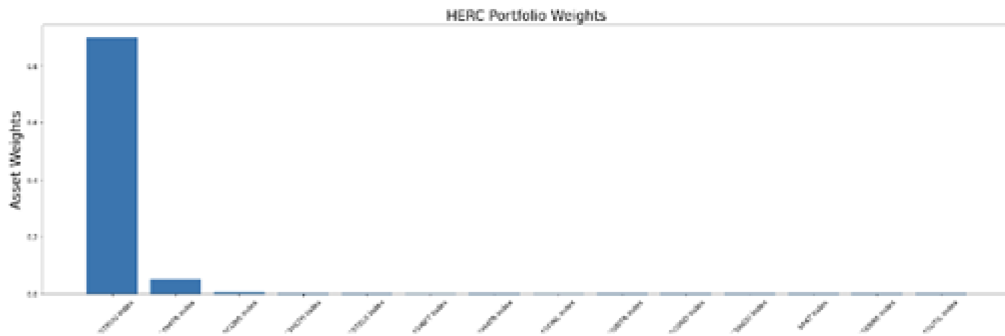




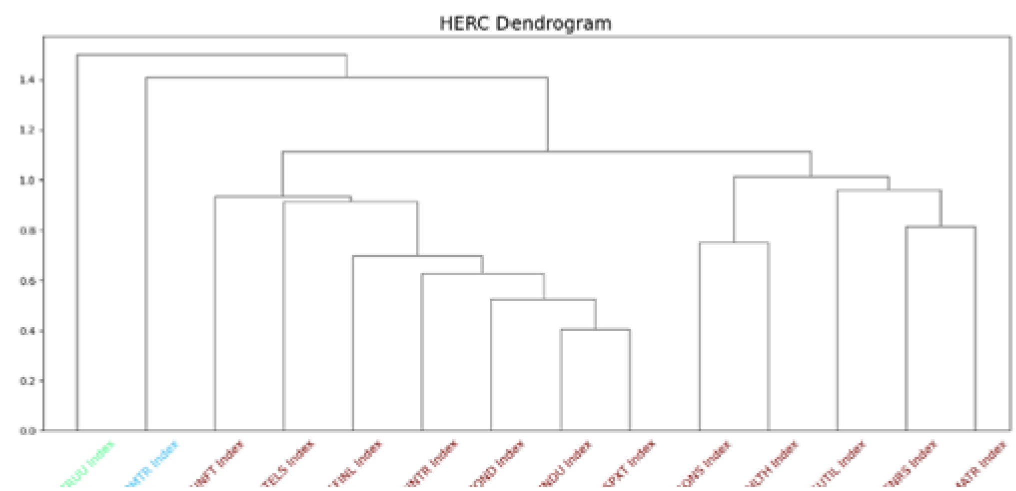
Expected Shortfall

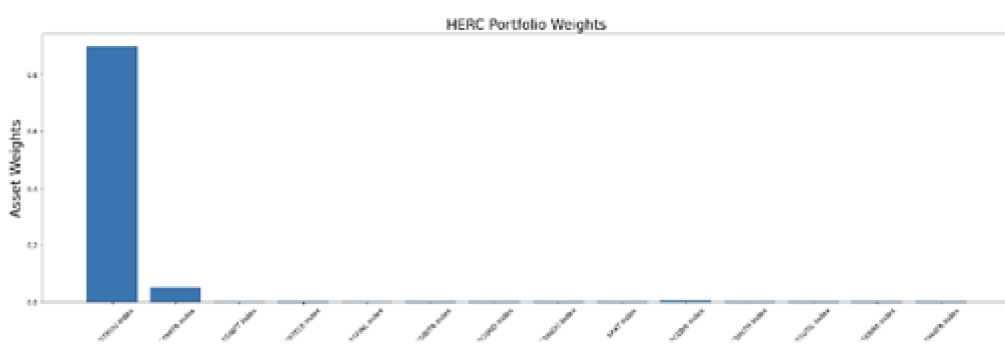
Average



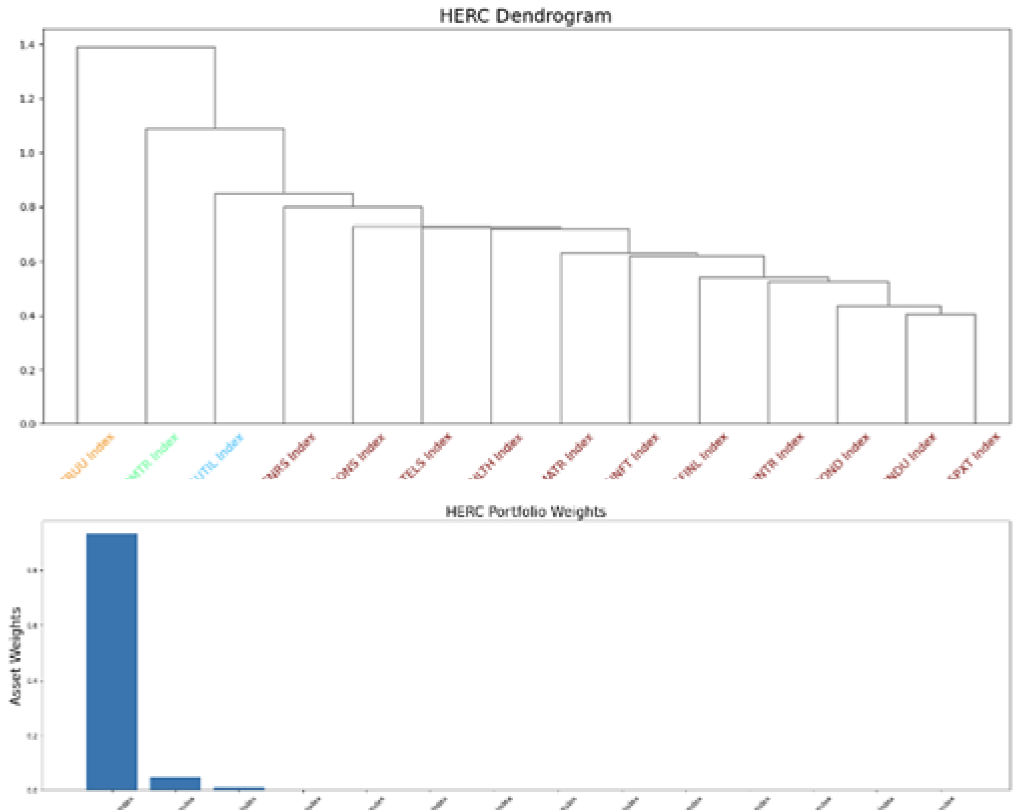


Complete

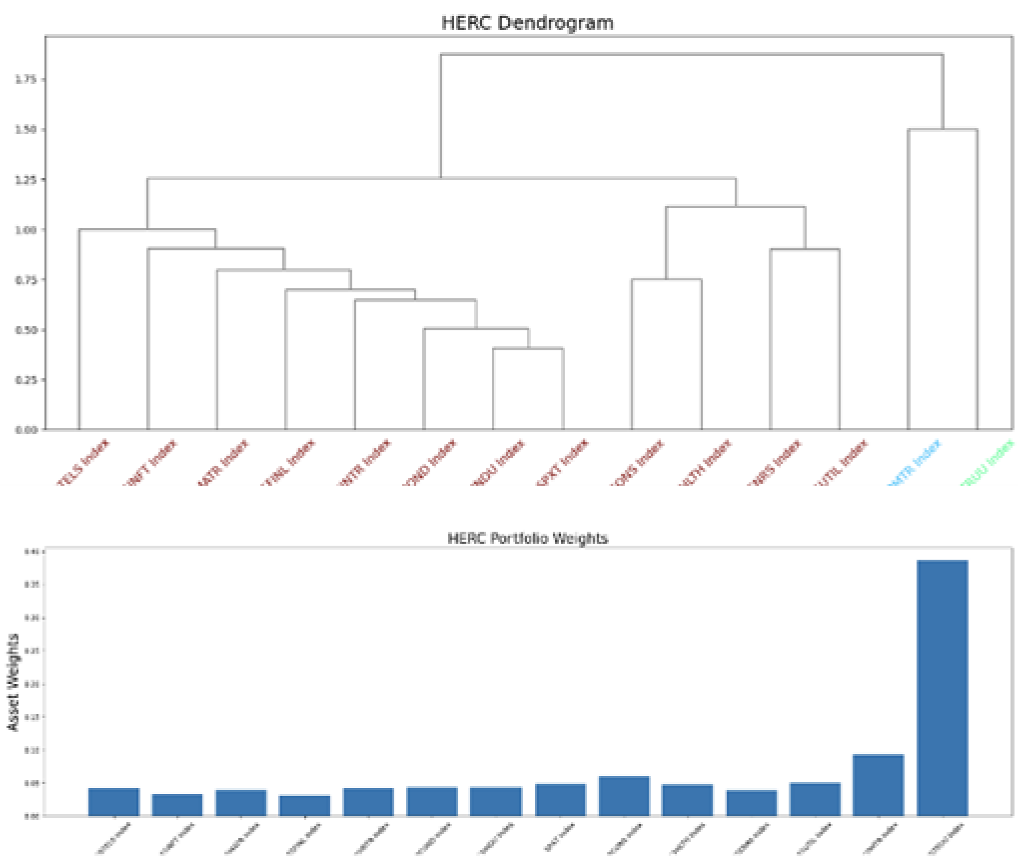




Single

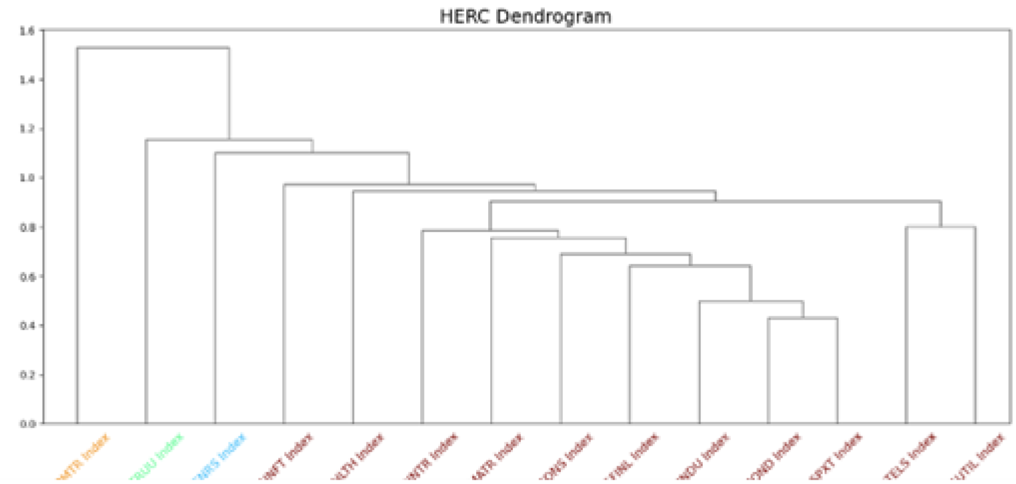


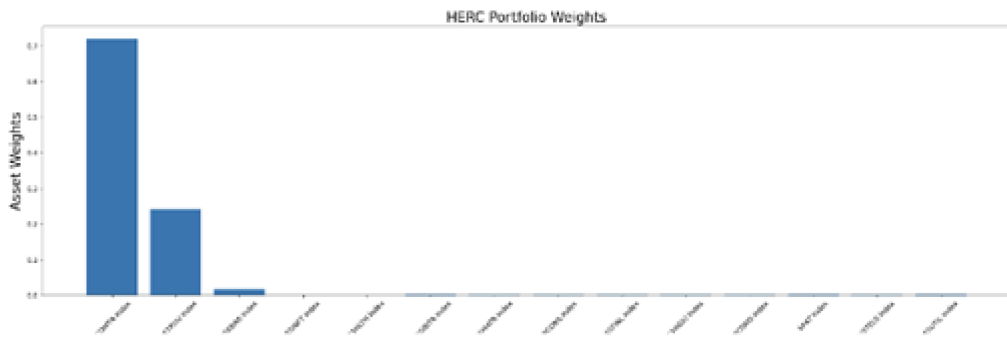
Ward



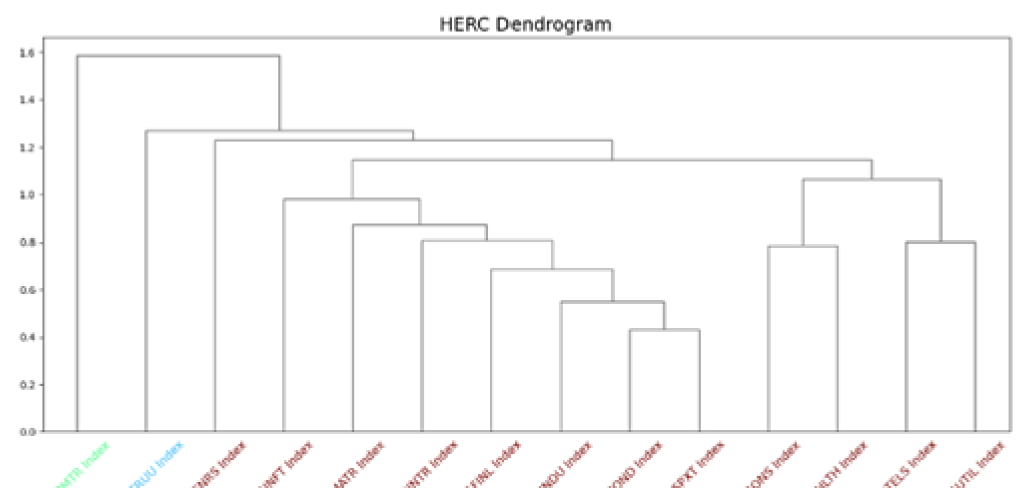
Standard Deviation

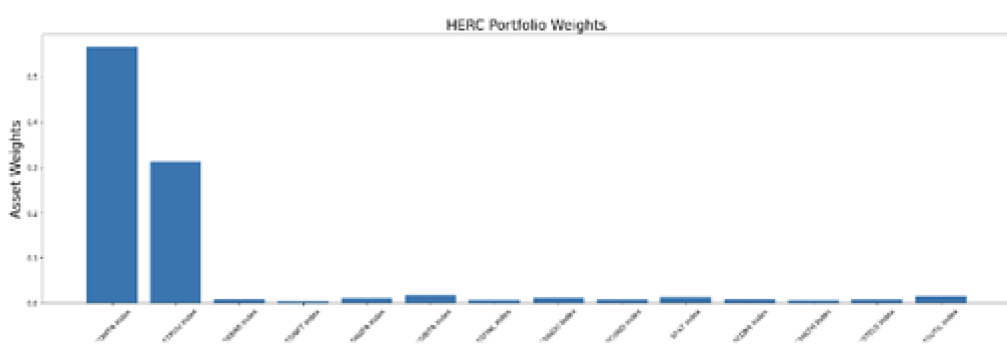
Average



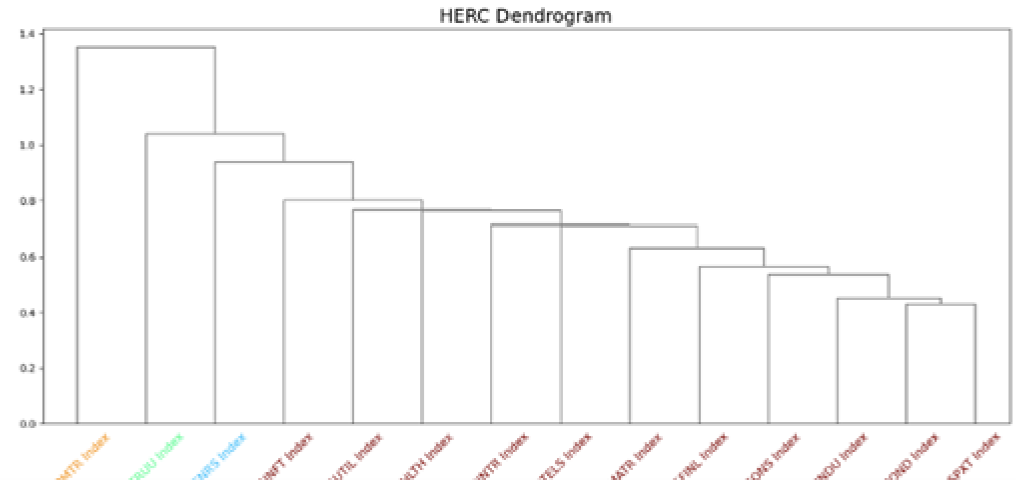


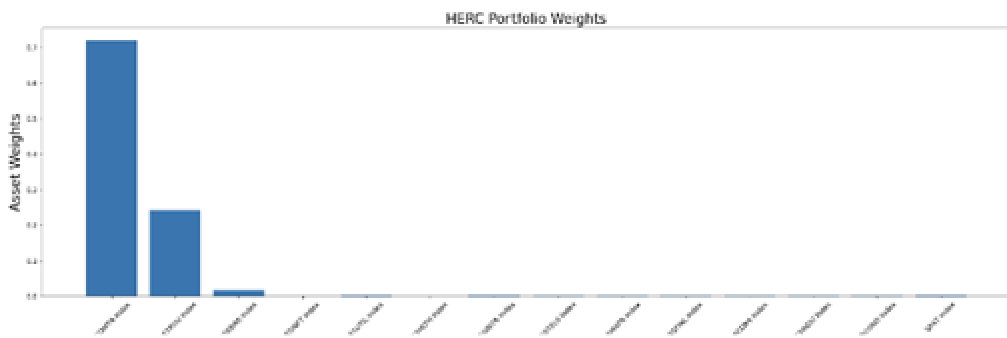
Complete



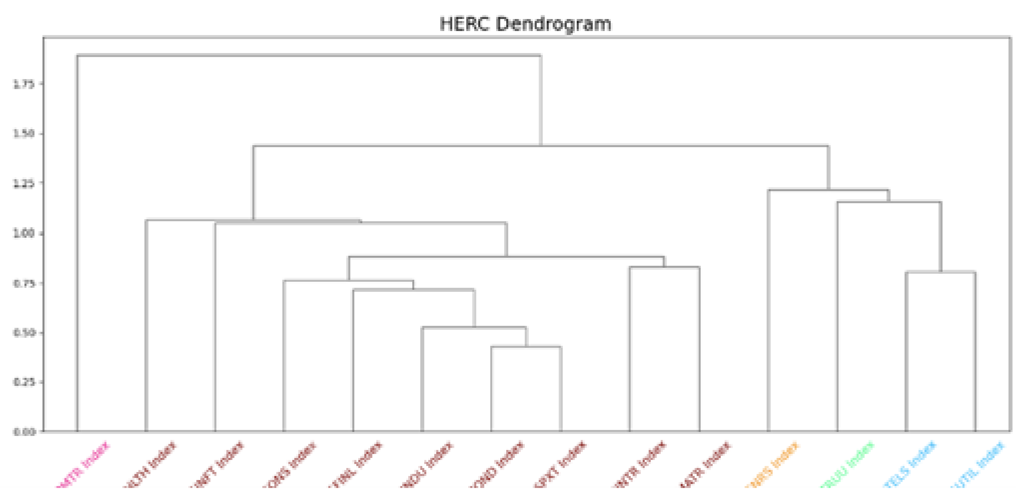


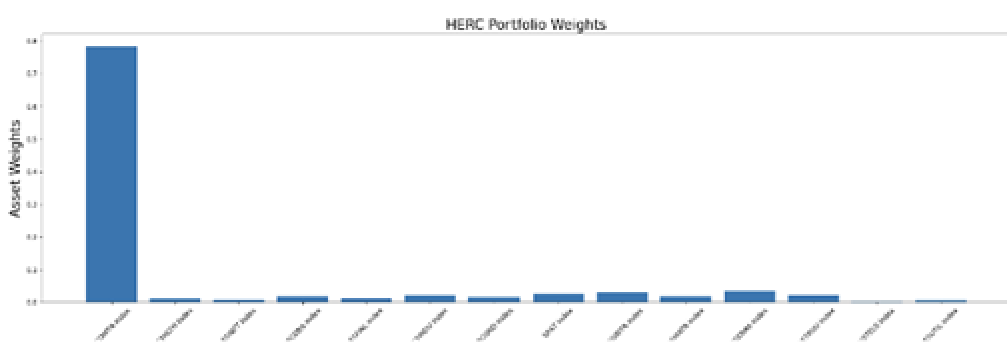
Single





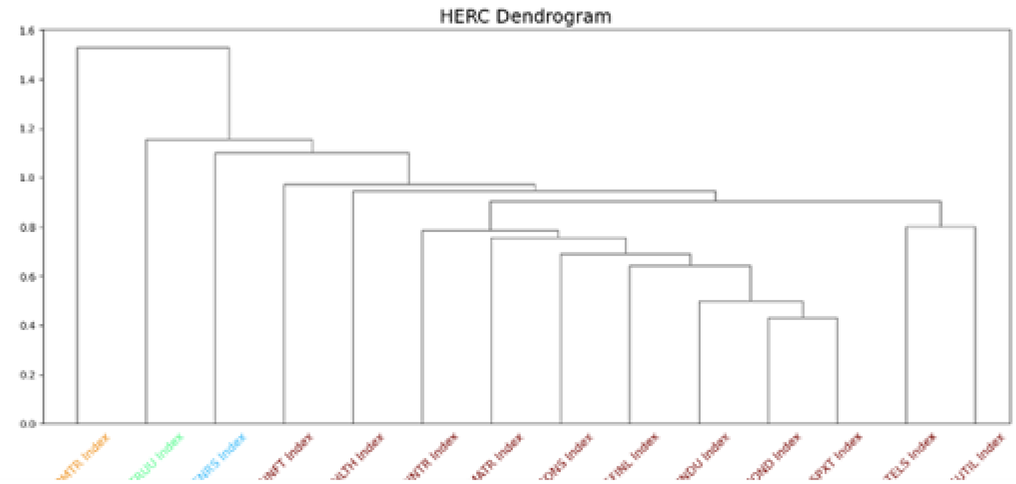
Ward

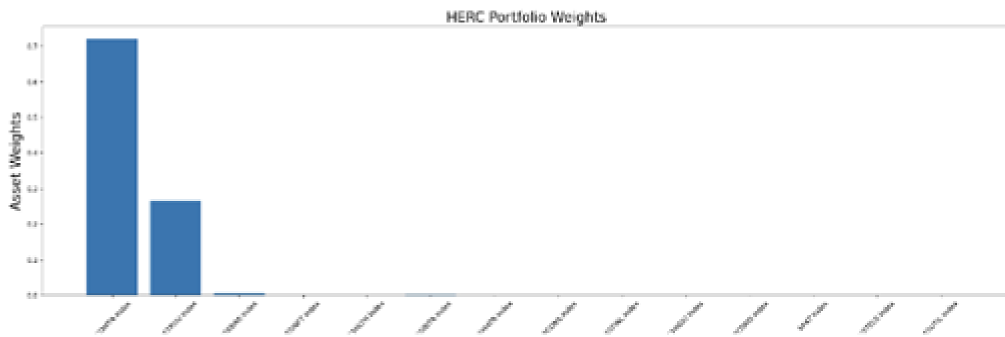




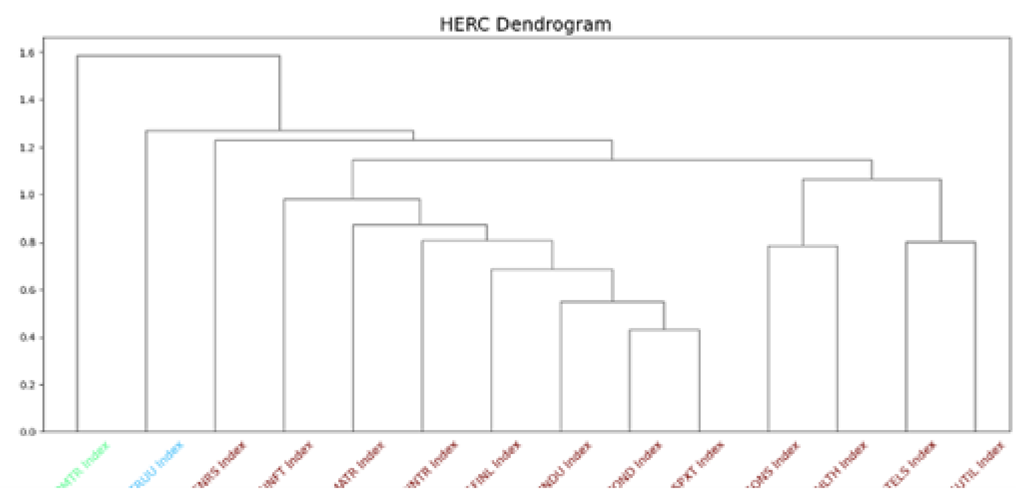
Variance

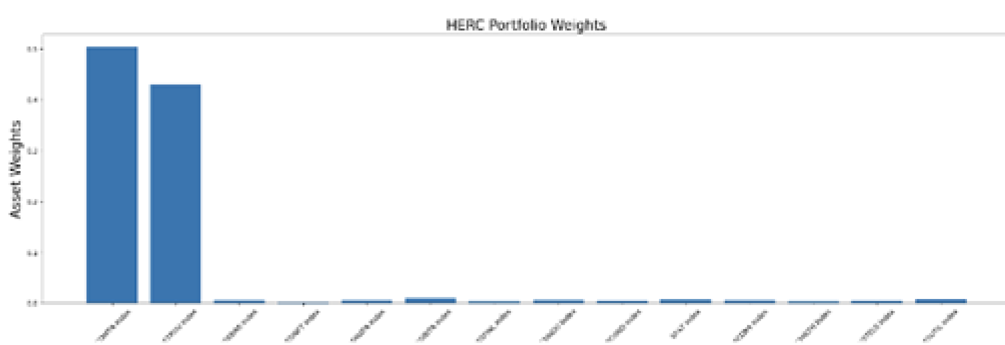
Average



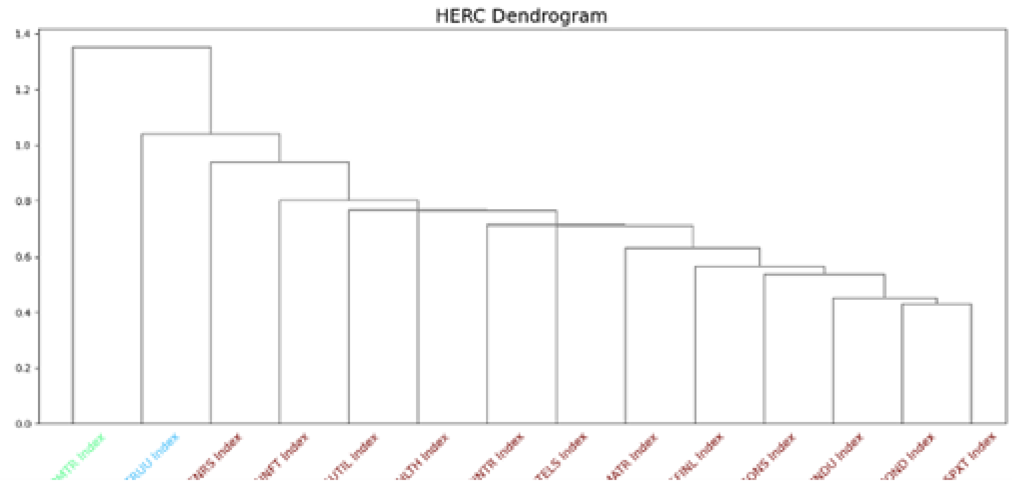


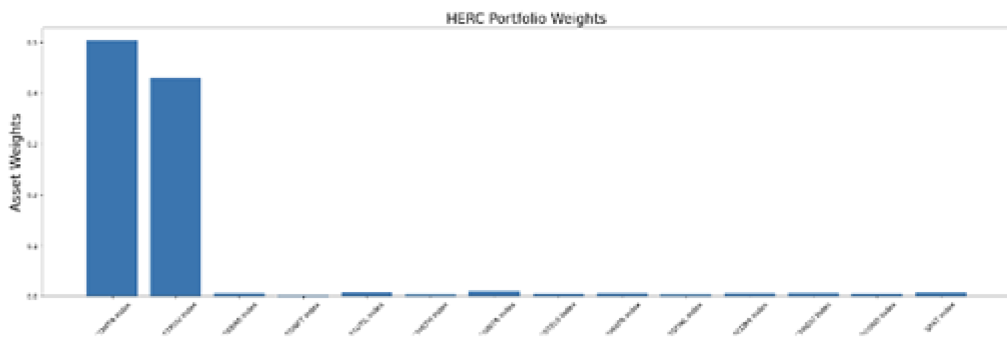
Complete



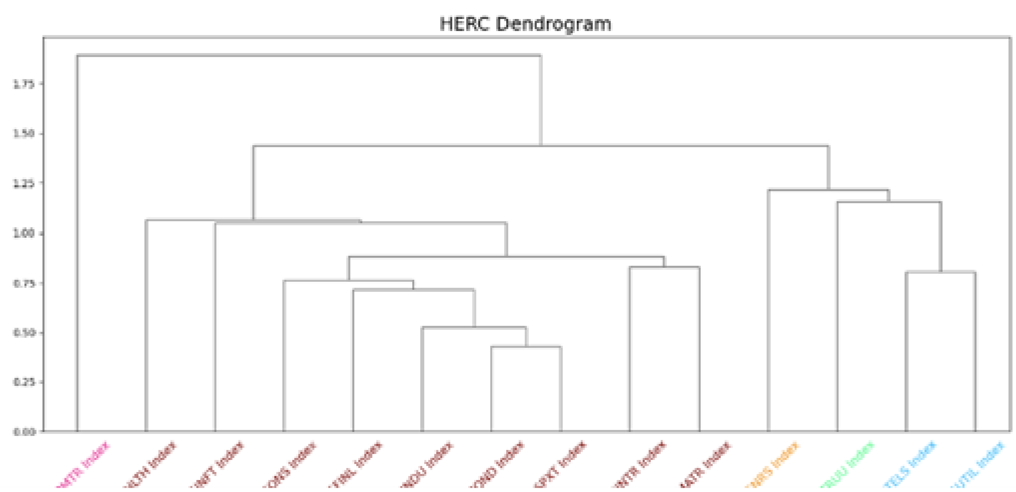


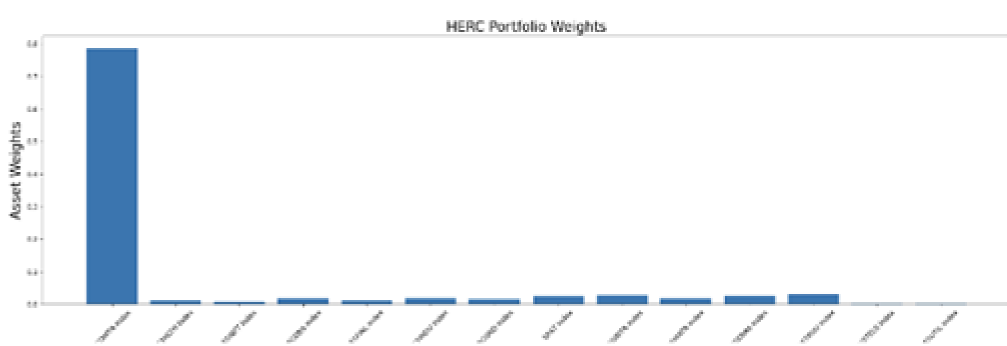
Single





Ward





Script of Interactive App

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from portfoliolab.modern_portfolio_theory import MeanVarianceOptimisation
from portfoliolab.clustering import HierarchicalEqualRiskContribution
import scipy.cluster.hierarchy as spc
from scipy.stats import kurtosis, skew,norm


# In[1]:


import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.express as px


# In[ ]:


df = pd.read_csv('https://raw.githubusercontent.com/qiaominwang/Capstone/main/cum_res.csv')
df = df.set_index('Date')


# In[ ]:


external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)


app.layout = html.Div([
    html.Div([

        html.Div([
```

```
        html.Div([
            dcc.Checklist(
                id = 'selected-methods',
                options=[
                    {'label': 'Inverse Variance', 'value': 'IV'},
                    {'label': 'Naive Risk', 'value': 'NR'},
                    {'label': 'Equal Risk', 'value': 'ER'},
                    {'label': 'Min Volatility', 'value': 'MV'},
                    {'label': 'Max Diversification', 'value': 'MD'},
                    {'label': 'HCAA', 'value': 'HCAA'},
                    {'label': 'HERC', 'value': 'HERC'}
                ],
                value=['HERC'],
                labelStyle={'display': 'inline-block'}
            )
        ],
        style={'width': '48%', 'display': 'inline-block'})
    ]),

    dcc.Graph(id='cum-re-graphic'),

    dcc.Slider(
        id='year--slider',
        min=2000,
        max=2020,
        value=2000,
        marks={str(year): str(year) for year in range(2000,2021,5)},
        step=None
    )
])
```

```
@app.callback(
    Output('cum-re-graphic', 'figure'),
    Input('selected-methods', 'value'),
    Input('year--slider', 'value'))
def update_graph(selected_methods, year_value):
    dff = df.loc[df['Year'] == year_value]

    fig = px.line(dff, x=dff.index,
                     y=selected_methods)

    fig.update_layout(margin={'l': 40, 'b': 40, 't': 10, 'r': 0}, hovermode='closest')

    return fig


if __name__ == '__main__':
    app.run_server(debug=True)
```