Hierarchial Risk Parity Method and DCC-GARCH for Tactical Asset Allocation

This report examines some common solutions to the classical optimal portfolio construction problem proposed by Markowitz(1952). In Part II we describe the methodologies, namely mean-variance optimization, hierarchical risk parity, GARCH-DCC and finally, GARCH-DECO. The empirical section tests these frameworks on the universe of 43 assets of our choice and gives detailed comparison between them. We will also construct portfolio with less hierarchy structure to show the performance limitation of HRP. The performance measures we base our conclusions on are Sharpe ratio and annual volatility. Our findings show that, for this particular portfolio, HRP method efficiently decreases out-of-sample variance and indeed provides us a stable and diversified portfolio. However, we will also go through the disadvantages of this method in details.

Contents

1 Introduction **1**

2 Methodology **2**
2.1 Setup . . . 2
2.2 Equal Weighted Portfolio . . . 2
2.3 Risk-based Portfolios . . . 2
2.3.1 Inverse Variance Portfolio . . . 2
2.3.2 Min Variance Portfolio . . . 2
2.3.3 Mean Variance Portfolio . . . 3
2.4 Hierarchical Risk Parity . . . 3
2.4.1 Tree Clustering . . . 3
2.4.1.1 Step 1: compute correlation distance . . . 3
2.4.1.2 Step 2: compute Euclidean distance of the correlation distance . . . 4
2.4.1.3 Step 3: form a cluster . . . 4
2.4.1.4 Step 4: link clusters . . . 4
2.4.1.5 Step 5: update distance matrix . . . 5
2.4.2 Quasi-Diagonalization . . . 5
2.4.3 Recursive Bisection . . . 6
2.5 Covariance Matrix Forecasting . . . 8

3 Data Preparation **8**

4 Results **9**
4.1 Comparison of portfolios . . . 10

5 Conclusion **11**

6 References **11**

1 Introduction

Most of the traditional risk-based asset allocation methods are based on the mean-variance optimization framework proposed by Markowitz (1952). The solution that Markowitz himself proposed is the famous Critical Line Algorithm (CLA), which is a quadratic optimization procedure specifically designed for inequality-constrained portfolio optimization that converges to an exact solution after a known number of iterations. Due to the instability of forecast returns the CLA poses, a number of risk-based methods have arisen, including Minimum Variance Portfolio (MVP) (Clarke, de Silva, & Thorley,), Inverse Volatility Weighted Portfolio (IVWP), Maximum Diversification Portfolio (MDP) (Choueifaty & Coignard,). However, as much as these classical methods do provide practical solutions for industry-wide portfolio optimization problem, they tend to give unstable results. It is known that quadratic programming requires the inversion of a positive-definite covariance matrix, which is prone to large errors when the covariance matrix is numerically ill-conditioned, i.e. it has a high and fast growing condition number, especially as we add more correlated (multi-collinear) investments for diversification. This is the famous dilemma known as Markowitz' curse: the benefits of diversification often are more than offset by estimation errors.

To address the widespread failure towards truly "optimal" portfolio construction, Lopez de Prado (2016) proposed Hierarchical Risk Parity (HRP), a machine-learning based technique that uses the information contained in the covariance matrix without requiring its inversion or positive-definiteness. Intuitively, return series in a selected universe of assets form a vector space that can be modelled as a completed, fully-connected graph. One reason for the instability of quadratic optimizers is that the vector space is modelled as a complete (fully connected) graph, where every node is a

potential candidate to substitute another. Algorithmically, inverting the matrix means evaluating every single correlation pair which magnifies estimation errors that lead to instability of portfolio weights. By identifying a hierarchical structure (dropping nodes and edges), HPR effectively limits the estimation error.

The second issue related to optimal portfolio construction concerns the forecast of the variance of the returns. The sample-based estimation means that, in order to obtain a covariance matrix of size 50 we would need $50 * 51/2$ i.i.d. return observations which is worth of 5 years of daily return time series. Since asset returns empirically exhibit heteroskedasticity with volatility clustering (Zakamulin 2015; Lopez de Prado 2016) and due to the time variant nature of the correlation structures, we resort to a GARCH(1,1) type dynamic conditional correlation (DCC) model proposed by Engle (2002). We first estimate the conditional volatility for each one of the n return series (GARCH), in preparation for a Constant Conditional Correlation (CCC) Estimator of the standardized residuals, with a stepup to capture the dynamics in the correlation (DCC).

The remainder of the paper is structured as follows. In Section 2, we introduce the fundamental methodology for our analysis, namely equal-weighted, risk-based and HPR portfolio construction, along with GARCH-DCC model for correlation matrix forecasting. Selection of assets and detailed implementation will be covered in Section 3, where we compare the performance of our portfolios in respect of Sharpe ratio and annual volatility. Python and related libraries were the main source of computational analysis throughout this study.

---

2 Methodology

For our empirical results, we look at the following portfolio construction methods, namely, equal weighting, traditional risk-based and HPR. Our main focus for covariance matrix estimation will be Dynamic Conditional Correlation GARCH (DCC-GARCH), based on the assumption that the conditional covariance Σ_t can be decomposed into two latent processes D_t and R_t, the conditional variance and the conditional correlation:

$$\mathbf{\Sigma_t} = \mathbf{D_t R_t D_t}$$

where D_t can be further modelled as a GARCH process and R_t can be estimated with the unconditional correlation.

2.1 Setup

Our objective is to construct portfolios based on the weighting schemes defined by $N \times 1$ vector of asset weights $\mathbf{w} := (w_1, ..., w_N)'$, constrained by $\Sigma_1^N w_i = 1$; the $N \times 1$ returns time series $\mathbf{r_t} := (r_{1t}, ..., r_{Nt})'$; the $N \times N$ covariance matrix $\mathbf{\Sigma}$ of the returns forecasted for the desired holding horizon.

We consider long only portfolios in our entire analysis.

2.2 Equal Weighted Portfolio

Intuitively, in an equal weighting scheme we distribute equal weight to each one of the selected assets with $1/N$, where N denotes the number of assets in the portfolio. As we will see later, this weighting scheme actually outperforms most of the risk-based methods, likely due to the instability of covariance matrix estimates for our selection of assets.

$$\mathbf{w}_{ew} = \left(\frac{1}{N}, ..., \frac{1}{N}\right)$$

2.3 Risk-based Portfolios

2.3.1 Inverse Variance Portfolio.

Also known as the Inverse Volatility Weighted Portfolio (IVP), this weighting scheme computes the each asset's inverse variance, as the proportion of the sum of all inverse variances.

In particular, let $\sigma := diag(\Sigma)$ be the $N \times 1$ vector of standard deviation of arithmetic returns, then the IVP assigns the following weights to the N assets

$$\mathbf{w}_{iv} = \left(\frac{\frac{1}{\sigma_1}}{\sum_{j=1}^{N} \frac{1}{\sigma_j}}, ..., \frac{\frac{N}{\sigma_1}}{\sum_{j=1}^{N} \frac{1}{\sigma_j}}\right)$$

2.3.2 Min Variance Portfolio.

This is the classical Markowitz's CLA algorithm which we formalize mathematically as follows. Given the usual setup, we denote $\mathbf{m} := (\mu_1, ..., \mu_N)'$, the mean return vector of each time series.

Then $\Sigma := S'S$, where S is the demeaned return matrix. We aim to solve the quadratic program (QP) where

$$\begin{aligned} \text{minimize} \quad & J := \tfrac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} \\ \text{subject to } & \mathbf{m}^T\mathbf{w} > \mu_b, \text{ and } \mathbf{e}^T\mathbf{w} = 1 \end{aligned}$$

where μ_b is the acceptable baseline expected rate of return and e denotes the vector of ones.

The KKT conditions for this quadratic program are

$$0 = \mathbf{\Sigma}\mathbf{w} - \lambda\mathbf{m} - \gamma\mathbf{e} \tag{1}$$

$$\mu_b < \mathbf{m}^T\mathbf{w}, \mathbf{e}^T\mathbf{w} = 1, 0 \leq \lambda \tag{2}$$

$$\lambda^T(\mathbf{m}^T\mathbf{w} - \mu_b) = 0 \tag{3}$$

for some $\lambda, \gamma \in \mathbb{R}$. Since the covariance matrix is symmetric and positive definite, we know that if (w, λ, γ) is any triple satisfying the KKT conditions then $\mathbf{w}$ is necessarily a solution to J. Indeed, it is easily shown that if J is feasible, then a solution to J must always exist and so a KKT triple can always be found for J.

Equation 3 implies $\lambda = 0$. And combining equation 1 and equation 2 gives

$$\gamma = (\mathbf{e}^T \mathbf{\Sigma}^{-1} \mathbf{e})^{-1}$$

From which we finally get the optimal weight vector

$$\bar{\mathbf{w}} = \frac{\mathbf{\Sigma}^{-1}\mathbf{e}}{\mathbf{e}^T \mathbf{\Sigma}^{-1}\mathbf{e}}$$

2.3.3 Mean Variance Portfolio.

The above provides us with a nice analytical solution to the min-variance quadratic minimizer, given that the covariance matrix is invertible. However, in reality, one would ideally like to maximize return while minimizing risk, which leads to the mean variance weighting scheme. Consider the QP:

$$\begin{aligned} \text{minimize} \quad & J_\lambda := \tfrac{1}{2}\mathbf{w}^T \mathbf{\Sigma} \mathbf{w} - \lambda \mathbf{m}^T \mathbf{w} \\ \text{subject to} \quad & \mathbf{e}^T \mathbf{w} = 1 \end{aligned}$$

The KKT conditions now collapse to

$$0 = \mathbf{\Sigma}\mathbf{w} - \lambda \mathbf{m} - \gamma \mathbf{e} \tag{4}$$

$$\mathbf{e}^T \mathbf{w} = 1 \tag{5}$$

Solving from the equation 4 and 5 gives

$$\gamma = \frac{1 - \lambda \mathbf{m}^T \mathbf{\Sigma}^{-1}\mathbf{e}}{\mathbf{e}^T \mathbf{\Sigma}^{-1}\mathbf{e}}$$

and

$$\bar{\mathbf{w}}_\lambda = (1-\alpha)\frac{\mathbf{\Sigma}^{-1}\mathbf{e}}{\mathbf{e}^T \mathbf{\Sigma}^{-1}\mathbf{e}} + \alpha \frac{\mathbf{\Sigma}^{-1}\mathbf{m}}{\mathbf{m}^T \mathbf{\Sigma}^{-1}\mathbf{e}}$$

where $\alpha = \lambda \mathbf{m}^T \mathbf{\Sigma}^{-1}\mathbf{e}$

2.4 Hierarchical Risk Parity

Finally, we introduce the HRP method, which is superior to the traditional risk-based method in the sense that it uses the information contained in the covariance matrix without requiring its inversion or positive-definitiveness (Lopez de Prado, Marcos, 2016), a highly desirable feature when the covariance matrix is singular or exhibits high condition number.

The algorithm consists of three stages: Tree clustering, quasi-diagonalization and recursive bisection.

2.4.1 Tree Clustering.

This lack of hierarchical structure allows weights to vary freely in unintended ways, which is a root cause of CLA's instability.

The intuition for tree clustering is that trimming a fully connected complete graph down to a hierarchical structure allows weights to be distributed top-down, consistent with how asset managers usually build their portfolios from asset class to sectors to individual securities. A tree structure also implies that N nodes are connected by $N-1$ edges, meaning weights only rebalance among groups of the same hierarchy.

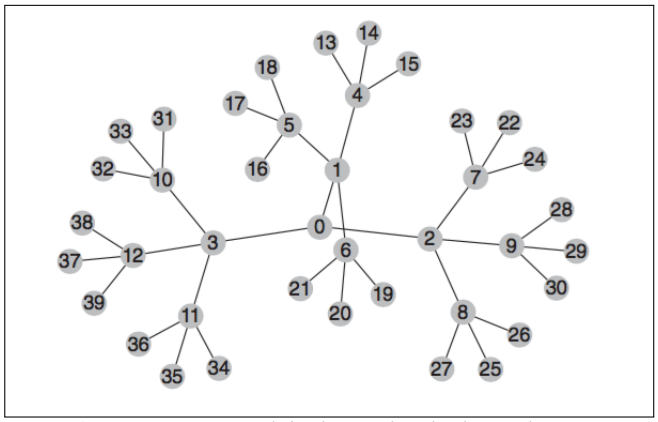


Figure 1. Tree structures exhibit hierarchical relationships.

Consider a TxN matrix of returns series of N variables over T periods. We would like to combine these N column-vectors into a hierarchical structure of clusters.

2.4.1.1 Step 1: compute correlation distance.

This step measure the similarity between time series, and therefore to form clusters of similar assets based on the original correlation matrix.

Given the correlation matrix $\rho := \{\rho_{i,j}\}_{i,j=1,\ldots,N}$, where $\rho_{i,j}$ is the Pearson's correlation coefficient of i^{th} and j^{th} asset, we define the distance matrix $D := \{d_{i,j}\}_{i,j=1,\ldots,N}$, with the distance measure given by

$$d_{i,j} = \sqrt{\frac{1}{2}(1-\rho_{i,j})}$$

D defined in such a way is a proper metric space (see De Prado 2018 for a detailed proof)

$$\{\rho_{ij}\} = \begin{bmatrix} 1 & .7 & .2 \\ .7 & 1 & -.2 \\ .2 & -.2 & 1 \end{bmatrix} \rightarrow \{d_{ij}\} = \begin{bmatrix} 0 & .3873 & .6325 \\ .3873 & 0 & .7746 \\ .6325 & .7746 & 0 \end{bmatrix}$$

2.4.1.2 Step 2: compute Euclidean distance of the correlation distance.

Since D is a proper metric space, we can further compute the Euclidean distance matrix $\tilde{D} := \{\tilde{d}_{i,j}\}_{i,j=1,...,N}$ measuring the distance between each pair of columns of the encoded correlation matrix, where

$$\tilde{d}_{i,j} := \sqrt{\sum_{n=1}^{N}(d_{n,i} - d_{n,j})^2}, i, j = 1...N$$

$$\{d_{i,j}\} = \begin{bmatrix} 0 & .3873 & .6325 \\ .3873 & 0 & .7746 \\ .6325 & .7746 & 0 \end{bmatrix} \rightarrow \{\tilde{d}_{i,j}\}_{i,j=\{1,2,3\}} = \begin{bmatrix} 0 & .5659 & .9747 \\ .5659 & 0 & 1.1225 \\ .9747 & 1.1225 & 0 \end{bmatrix}$$

2.4.1.3 Step 3: form a cluster.

With the distance matrix computed, we form the first cluster: picking the pair that gives the least distance, i.e. we cluster (i^*, j^*) such that $(i^*, j^*) = argmin_{i,j,i\neq j}\{\tilde{d}_{i,j}\}$ and denote this cluster as $u[1]$

$$\{\tilde{d}_{i,j}\}_{i,j=\{1,2,3\}} = \begin{bmatrix} 0 & .5659 & .9747 \\ .5659 & 0 & 1.1225 \\ .9747 & 1.1225 & 0 \end{bmatrix} \rightarrow u[1] = (1, 2)$$

2.4.1.4 Step 4: link clusters.

Having formed at least one cluster, we would like to proceed with the merging process in order to build on the current hierarchy. The measure of dissimilarity between the clusters is known as the linkage criterion.

In this study, we examine two agglomerative clustering linkage criteria:

- Single Linkage, a.k.a. the nearest point algorithm. SL keeps the distance between two clusters as the minimum of the distance between any two points in the two clusters C_i, C_j

$$d_{C_i,C_j} = min\{\tilde{D}(x \in C_i, y \in C_j)\}$$

- Average Linkage, on the other hand, links clusters based on the average of the distance between any two points in the clusters

$$d_{C_i,C_j} = mean\{\tilde{D}(x \in C_i, y \in C_j)\}$$

Below is a numerical example computed with the SL technique.

$$u[1] = (1, 2) \rightarrow \{\dot{d}_{i,u[1]}\} = \begin{bmatrix} \min[0, .5659] \\ \min[.5659, 0] \\ \min[.9747, 1.1225] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ .9747 \end{bmatrix}$$

The visualisation of nearest point linkage and farthest point linkage as shown below exhibits different hierarchical structures of our selected universe of assets:

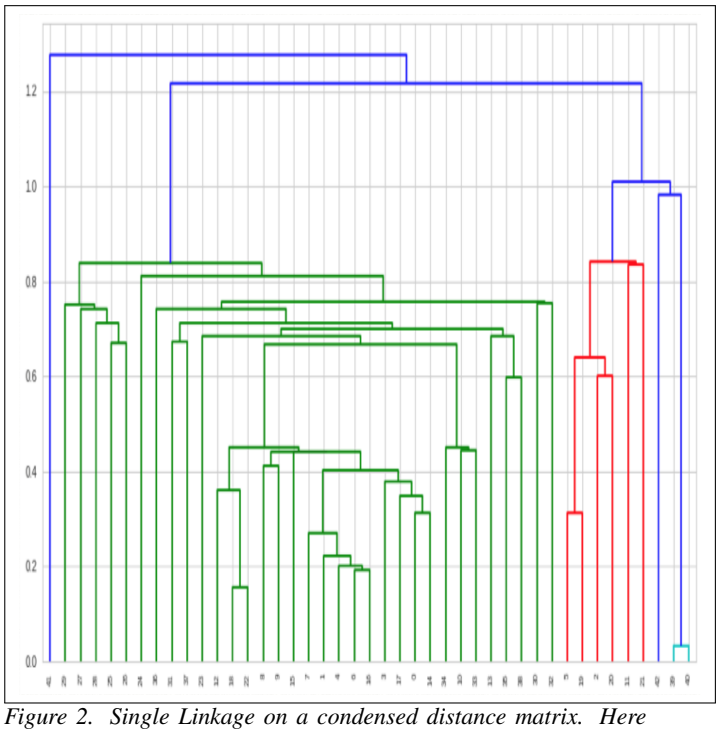


Figure 2. Single Linkage on a condensed distance matrix. Here Y-axis measures the distance between the two merging leaves.

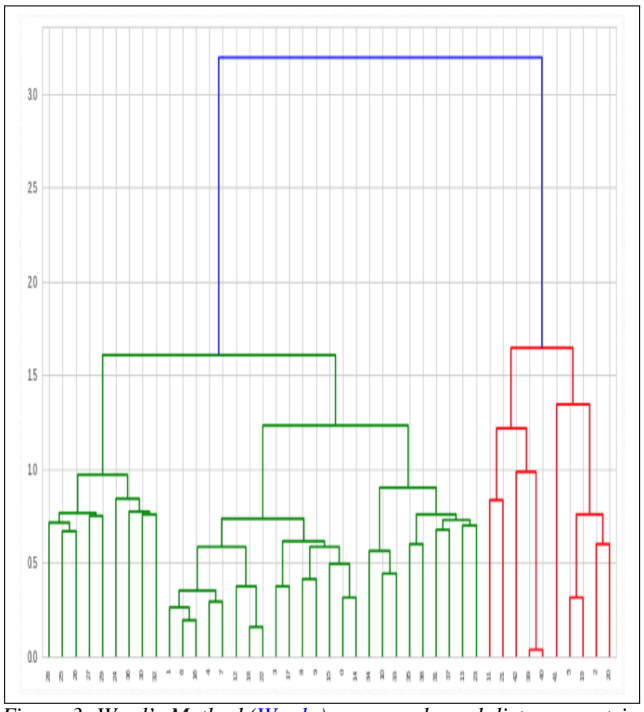


Figure 3. Ward's Method (Ward,) on a condensed distance matrix. Here Y-axis measures the distance between the two merging leaves.

2.4.1.5 Step 5: update distance matrix.

Next, we update the Euclidean distance matrix $\{\tilde{d}_{i,j}\}$ with the new cluster u. This is done by appending d_{C_i,C_j} and dropping the clustered columns and rows $j \in u[1]$

$$\{\tilde{d}_{i,j}\}_{i,j=\{1,2,3,4\}} = \begin{bmatrix} 0 & .5659 & .9747 & 0 \\ .5659 & 0 & 1.1225 & 0 \\ .9747 & 1.1225 & 0 & .9747 \\ 0 & 0 & .9747 & 0 \end{bmatrix}$$

$$\{\tilde{d}_{i,j}\}_{i,j=\{3,4\}} = \begin{bmatrix} 0 & .9747 \\ .9747 & 0 \end{bmatrix}$$

The final step is a recursion of step $3-5$, appending all $N-1$ clusters to the original D matrix, until all nodes have been included. In this example,

$$\{\tilde{d}_{i,j}\}_{i,j=\{3,4\}} = \begin{bmatrix} 0 & .9747 \\ .9747 & 0 \end{bmatrix} \rightarrow u[2] = (3,4) \rightarrow \text{Stop}$$

The resulting distance matrix $\tilde{D}$ gives us a hierarchical structure of the clusters of assets

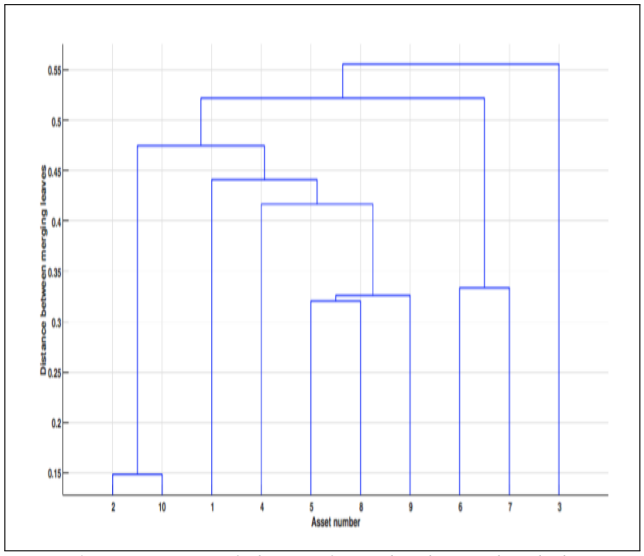


Figure 4. A Sequence of clusters formed in hierarchical clustering represented as a dendrogram with y-axis representing the distance between the two merging leaves. Source: (Jain & Jain,)

2.4.2 Quasi-Diagonalization.

This step rearranges the rows and columns of the covariance matrix using the information from the clustering algorithm, so that similar investments are placed together, and dissimilar investments are placed far apart. It allows us to allocate weights optimally following an inverse-volatility allocation. The algorithm works as follows: We know that each row of the linkage matrix merges two branches into one. We replace clusters in $(y_{N-1,1}, y_{N-1,2})$ with their constituents recursively until no clusters remain. The output is a sorted list of original nodes.

Note that the rendering requires no change of basis, and the order of the clustering is preserved, as shown below in the implementation.

```
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()
```

Figure 5. Quasi-diagonalization implementation

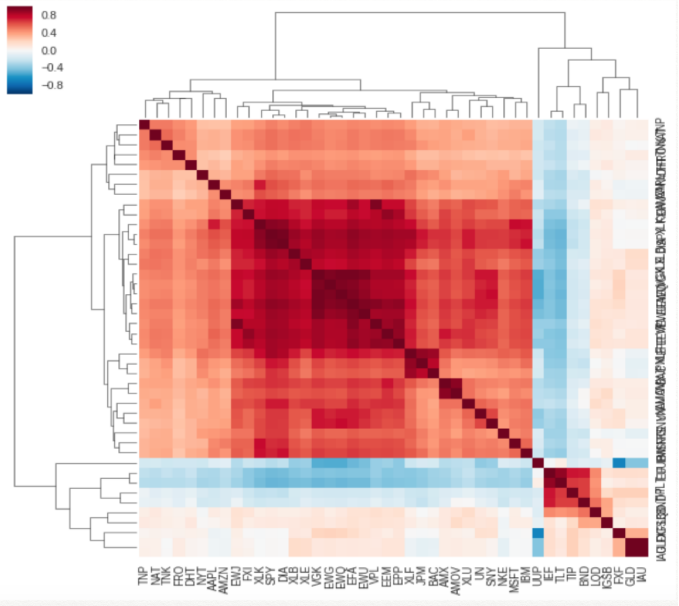


Figure 7. Quasi-diagonalization implemented with min distance linkage

Based on this algorithmic logic, we present the following dignonalization, given by our selected linkage criteria, namely Single Linkage and Mean Linkage.

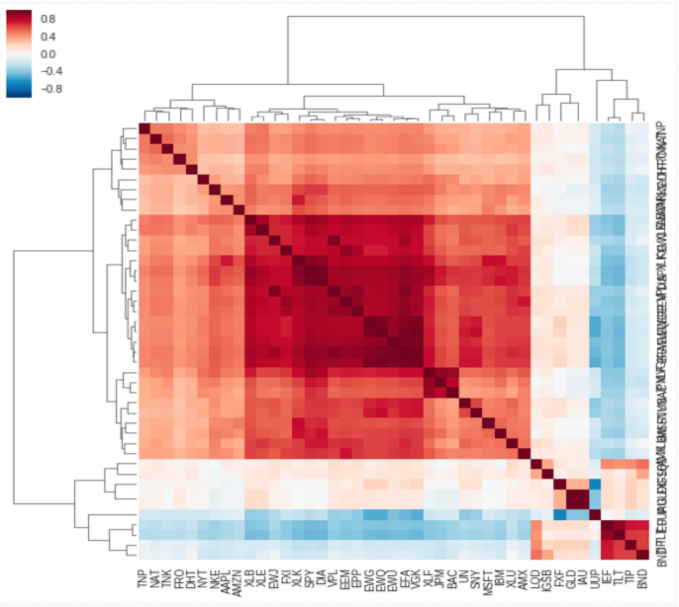


Figure 8. Quasi-diagonalization implemented with mean distance linkage

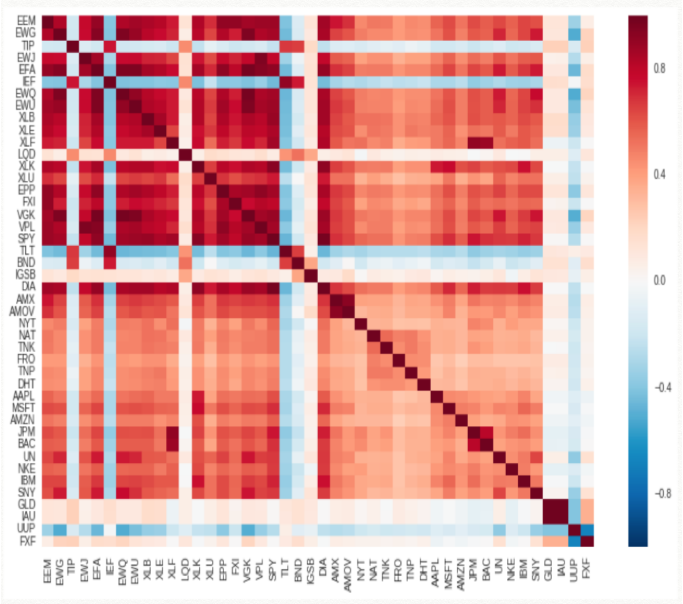


Figure 6. The original covariance matrix

2.4.3 Recursive Bisection.

The previous step quasi-diagonalization has delivered a quasi-diagonal matrix. The weights are allocated by the inverse-variance technique which can be shown to be optimal for a diagonal covariance matrix as follows:

Consider the standard quadratic optimization problem of size N. As we have seen previously, the quadratic optimizer

for this program is given by

$$\bar{\mathbf{w}} = \frac{\mathbf{\Sigma}^{-1}\mathbf{e}}{\mathbf{e}^T\mathbf{\Sigma}^{-1}\mathbf{e}}$$

now with Σ being diagonal, the weights vector can be written as

$$\bar{\mathbf{w}}_n = \frac{\mathbf{\Sigma}_{n,n}^{-1}}{\sum_{i=1}^{N}\mathbf{\Sigma}_{i,i}^{-1}}$$

In this case, we would want to split a weight between two bisections of a subset, i.e. the case where $N = 2$, giving us the weight formula

$$\bar{\mathbf{w}}_1 = 1 - \frac{\sigma_{1,1}}{\sigma_{1,1} + \sigma_{2,2}}$$

We can make use of these facts to perform two-fold execution: (1) bottom-up, given as in step 3b, where the variance of the subset $L_i^{(j)}$ is defined by the inverse-variance weightings $\tilde{w}_i^{(j)}$); or (2) top-down (given as in step 3c), to split allocations between adjacent subsets in inverse proportion to their aggregated variances. The weights are thus scaled down as each cluster is recursively bisected until a single asset is left in each cluster.

Formally, the iterative bisection algorithm goes as follows

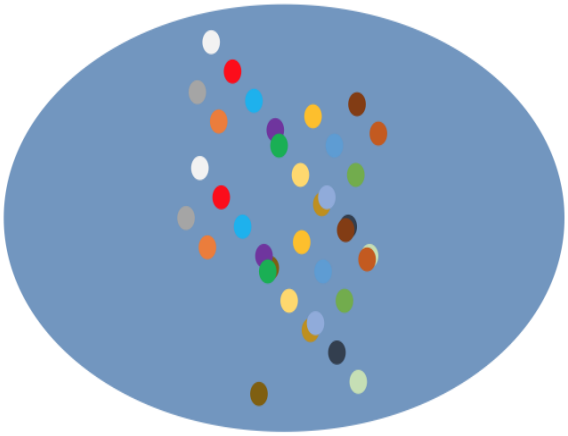
1. Initialize a list of assets in the portfolio with $L = \{L_0\}$, with $L_0 = \{n\}_{n=1,\ldots,N}$.
2. Initialize a vector of weights as $w_i = 1, i = 1, \ldots, N$.
3. Stop if $|L_i| = 1, \forall L_i \in L$
4. For each $L_i \in L$ such that $|L_i| > 1$
   - Bisect L_i into two subsets, $L_i^1 \cup L_i^2 = L_i$, where $|L_i^1| = int[\frac{1}{2}|L_i|]$
   - Calculate the variance of L_i^j, $j = 1, 2$ as $\tilde{V}_i^j = \tilde{w}_i^j V_i^j \tilde{w}_i^{j\prime}$, where V_i^j is the covariance m the elements within cluster j, and

$$\tilde{w}_i^j = \frac{tr[V_i^j]^{-1}}{\sum_i tr[V_i^j]^{-1}},$$

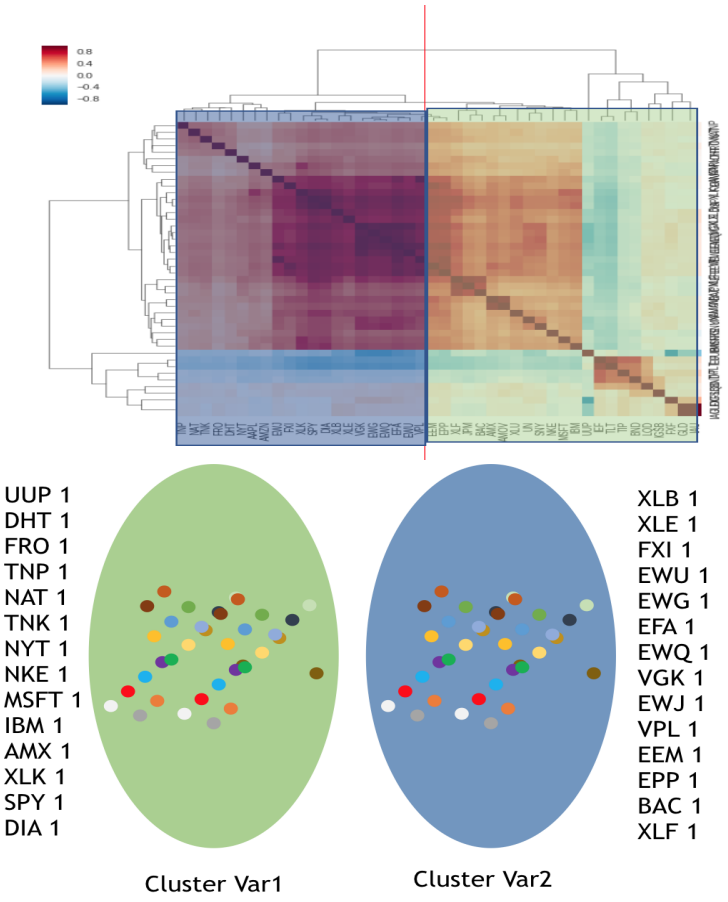
     which is the inverse volatility weight for the elements of the cluster.
   - Compute the split factor $\alpha_i = 1 - \frac{\tilde{V}_i^1}{\tilde{V}_i^1 + \tilde{V}_i^2}$
   - Rescale allocations w_n by a factor of $\alpha_i \forall\, n \in L_i^1$
   - Rescale allocations w_n by a factor $(1 - \alpha_i) \forall\, n \in L_i^2$
5. Loop to Step 2.

Below is an illustration of the application of the algorithm to recursively bisect our selected universe of assets:

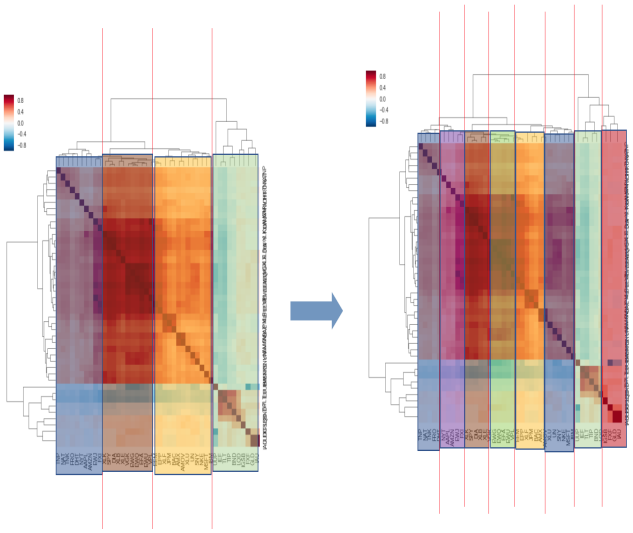
Assume we have the original "blob" of assets, unstructured,



Applying the algorithm once to our quasi-diagonalized covariance matrix yields a new division of two subsets of assets.



We iterate the process until there is no single asset left out.



2.5 Covariance Matrix Forecasting

Asset returns exhibit heteroskedasticity with volatility clustering. For this reason, we used dynamic conditional correlation (DCC) GARCH model for covariance matrix forecasting, which allows the correlation structures to be time varying.

$$\mathbf{\Sigma_t = D_t R_t D_t}$$

where

$$\boldsymbol{D_t} = \begin{bmatrix} \sqrt{h_{1t}} & 0 & \cdots & 0 \\ 0 & \sqrt{h_{2t}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{h_{nt}} \end{bmatrix}$$

$$\boldsymbol{R_t} = \begin{bmatrix} 1 & \rho_{12,t} & \rho_{13,t} & \cdots & \rho_{1n,t} \\ \rho_{12,t} & 1 & \rho_{23,t} & \cdots & \rho_{2n,t} \\ \rho_{13,t} & \rho_{23,t} & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \rho_{n-1,n,t} \\ \rho_{1n,t} & \rho_{2n,t} & \cdots & \rho_{n-1,n,t} & 1 \end{bmatrix}$$

For D_t, each diagonal element is the time varying standard deviation from the univariate GARCH(1,1) model, which allows for conditional heteroskedasticity:

$$h_{i,t} = \alpha_{i0} + \alpha_{i1} a_{i,t-1}^2 + \beta_{i1} h_{i,t-1}$$

For R_t, it is the conditional correlation matrix of the standardized disturbances. It is further decomposed into:

$$\mathbf{R_t = Q_t^{*-1} Q_t Q_t^{*-1}}$$

where Q_t^* is a diagonal matrix with the square root of the diagonal elements of Q_t at the diagonal:

$$\boldsymbol{Q_t^*} = \begin{bmatrix} \sqrt{q_{11t}} & 0 & \cdots & 0 \\ 0 & \sqrt{q_{22t}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{q_{nnt}} \end{bmatrix}$$

And Q_t follows the below process:

$$\mathbf{Q_t = (1 - a - b)\bar{Q} + a\epsilon_{t-1}\epsilon'_{t-1} + bQ_{t-1}}$$

where $\mathbf{\bar{Q}}$ is the unconditional covariance matrix of standardized errors ϵ_t, and estimated as:

$$\overline{\boldsymbol{Q}} = \frac{1}{T}\sum_{t=1}^{T} \epsilon_t \epsilon_t^T$$

---

3 Data Preparation

The optimal weights in a portfolio depend on the general level of correlation between the assets of the investment universe (Schumann,2013), and the specific composition of the investment universe (Bertrand and Lapointe, 2018). We try and capture different correlation and composition structures by creating two different universes. We include 43 index based assets in our first universe, as top of the hierarchy. Furthermore, 422 individual stocks constitute the second universe, giving the matrix of time series returns running from January 1, 2018 through July 1, 2016. Portfolio construction is based on 30 random assets from the two universes, and backtesting is performed on return series data from the same period.

However, note that cross-validation for backtesting purposes in this case will be biased. The intuition is that the distribution of the errors from the iterative CV process does not tell us very much about the variability of the prediction from the forecast made using the final version of the model.

Portfolio 1 prices data: data with obvious hierarchical structure.

Portfolio 2 prices data: more realistic portfolio assets.

Portfolio 3 prices data: SP500 Universe.

---

4 Results

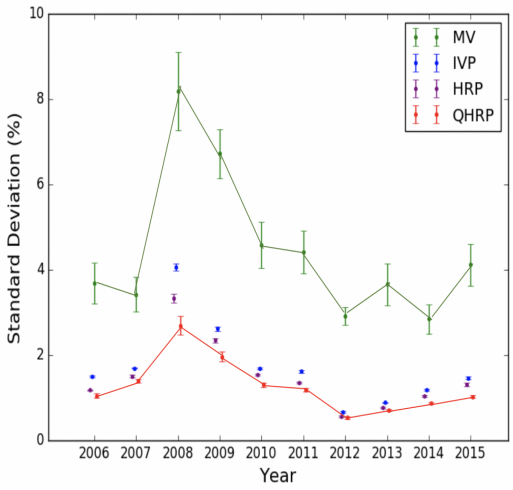


Figure 9. Backtesting result when we add DCC-Garch in the backtesting process for HRP. We can see it largely decreases out-of-sample portfolio variance.

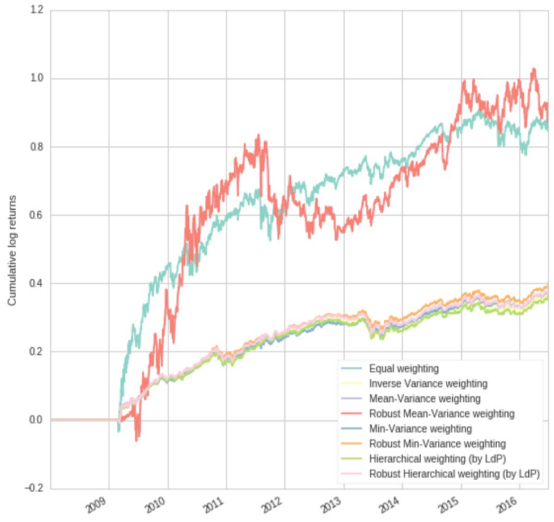


Figure 10. This is the backtesting result for the portfolio with obvious hierarchical structure. We can see that the mean-variance method performs badly out-of-sample, and Hierarchical Risk Parity and Robust Hierarchical Risk Parity portfolios perform much better and are more stable.

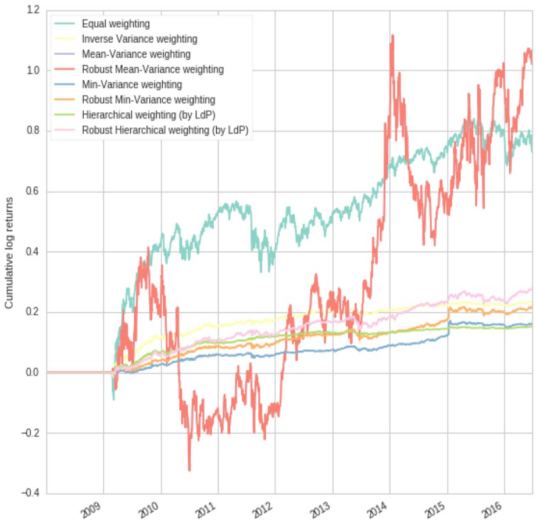


Figure 11. This is the backtesting result for the portfolio without obvious hierarchical structure.The assets selected are based on a more realistic assets-selection strategy.

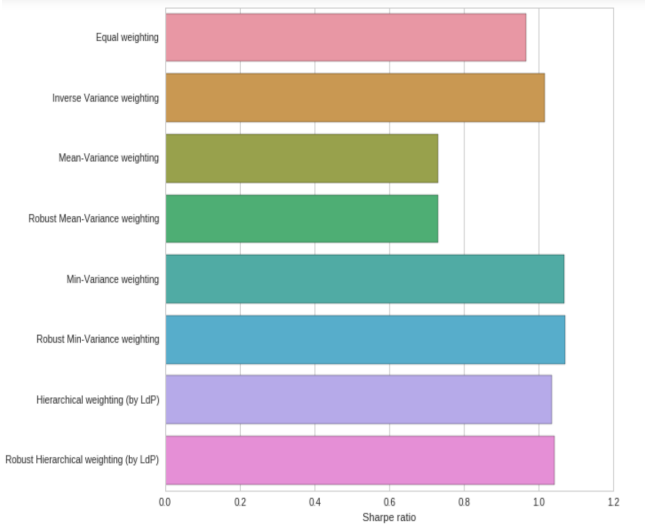


Figure 12. This graph compares the Sharpe-ratio we got from different portfolio construction methods. Hierarchical Risk Parity Method truly outperforms mean-variance method in this process. However, HRP does not seem to have a huge profit margin over other naively constructed portfolios, namely equal weighted and inverse variance weighted. Note that here in order to construct robust portfolios, we refer to LedoitWolf and OAS for shrinkage covariance estimation (Chen, Wiesel, Eldar, & Hero,)

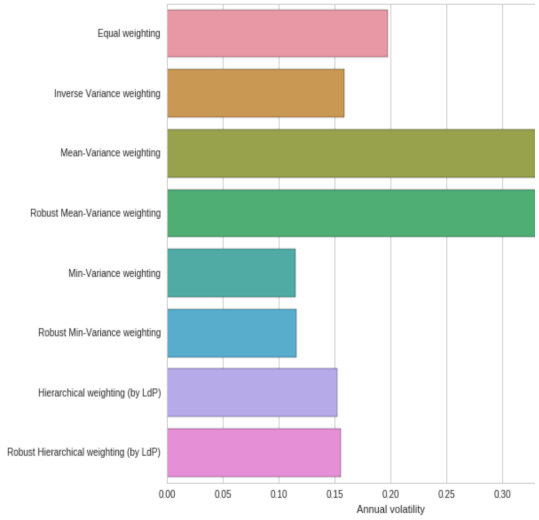


Figure 13. This graph compares the annual volatility we got from different portfolio construction methods. Hierarchical Risk Parity portfolios outperform mean-variance method in this process. The same problem is, it does not show strong evidence of advantage over other simple methods. However, in order to improve the results, we can add DCC-Garch to the process.

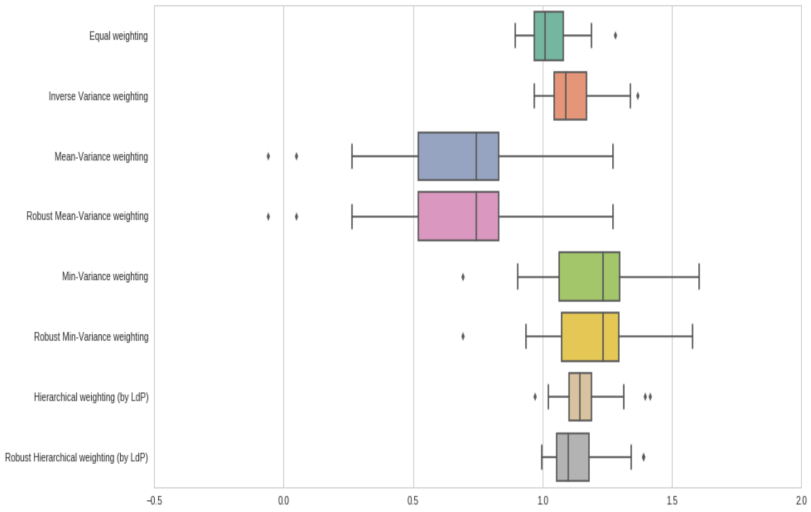


Figure 15. In the cross-validation step we randomly selected 30 stocks from the SP500 universe and calculated their returns, volatility, and Sharpe ratios. This figure presents the box-plot of Sharpe ratios across 30 assets for different weighting schemes. We would like to see the consistency of different portfolio construction methods. Overall, HPR portfolios outperform the risk-based portfolios. Equal weighting and inverse variance also show superior results likely because the construction process involves no inversion of covariance matrix

df_sharpes

	Equal weighting	Inverse Variance weighting	Mean-Variance weighting	Robust Mean-Variance weighting	Min-Variance weighting	Robust Min-Variance weighting	Hierarchical weighting (by LdP)	Robust Hierarchical weighting (by LdP)
0	1.054992	1.143275	0.726629	0.726629	1.163211	1.201882	1.260787	1.185244
1	0.974817	1.074648	1.052665	1.052665	1.191031	1.211980	1.149442	1.083088
2	1.190413	1.368602	0.985192	0.985192	1.603935	1.579882	1.415239	1.391807
3	0.986838	1.082171	1.005186	1.005196	1.281057	1.314100	1.190199	1.093257
4	1.182138	1.338858	0.511592	0.511532	1.419407	1.432158	1.396916	1.392430
5	1.089535	1.145633	0.788563	0.788563	1.303389	1.277906	1.186097	1.161493
6	0.974881	0.982049	1.272891	1.272891	0.955660	0.990062	1.055134	0.998560
7	0.903811	1.043509	1.102055	1.102055	1.269903	1.236611	1.089116	1.026382
8	0.966834	1.027125	0.697086	0.696923	0.960373	0.998780	1.029833	1.057417
9	1.283448	1.296829	0.761074	0.761074	1.123144	1.162663	1.312619	1.342978
10	1.020001	1.095951	0.656165	0.656165	1.252892	1.254765	1.139311	1.143095
11	0.997864	1.062864	0.800460	0.800460	1.023385	1.031440	1.098240	1.076625
12	0.993855	1.098668	0.791847	0.791847	1.291476	1.297918	1.179202	1.147413
13	0.943917	1.025680	1.023889	1.023891	1.297050	1.290090	1.113128	1.077280
14	1.070612	1.120673	0.600101	0.600101	1.227803	1.236611	1.177776	1.151085
15	1.026426	1.065551	0.604556	0.604657	1.101514	1.096119	1.125708	1.105998

Figure 14. This table shows the numerical values of the Sharpe ratios across 15 out of 30 assets from our universe, based on different weighting schemes

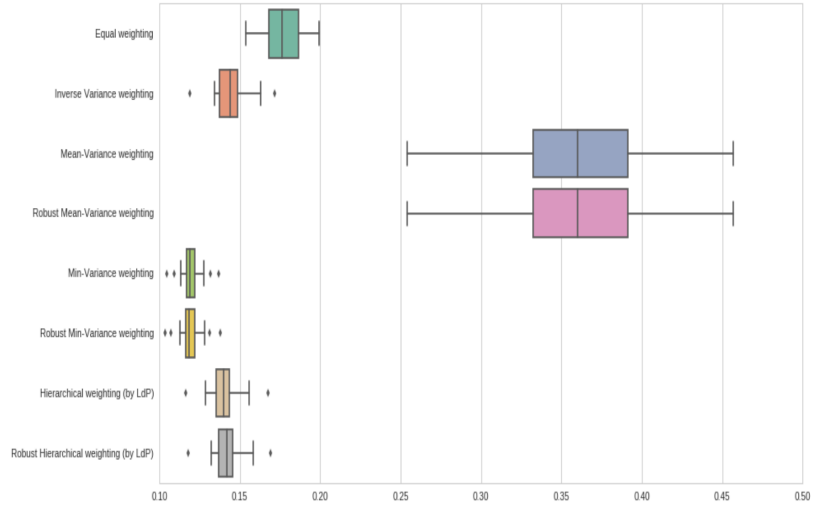


Figure 16. This figure presents the box-plot of annual volatilities across 30 assets for different weighting schemes. We can see the high instability of the portfolio performance for the mean-variance weighted portfolios

4.1 Comparison of portfolios

Hierarchical Risk Parity portfolio construction method does have its advantages over other portfolio methods. It performs especially well when we use robust covariance matrix

and DCC-Garch/DECO-Garch for covariance matrix prediction.

5 Conclusion

The key strength of HRP method lies when the portfolio is constructed with many underlying assets, where inversion of the covariance matrix becomes particularly challenging. Thus, we consider the ETF market a field that is potentially suitable for this portfolio construction approach. From the backtest results, HRP clearly performs better than mean-variance portfolio construction out of sample. However, some naïve portfolio construction methods seem to perform just as well or even better. It is sensitive to the way we predict covariance matrix. Better performance if DCC-GARCH is used to forecast covariance matrix. It performs especially well with daily rebalancing frequency. The longer the rebalancing period, the worse the portfolio performance.

6 References

Chen, Y., Wiesel, A., Eldar, Y. C., Hero, A. O. (2010, Oct). Shrinkage algorithms for mmse covariance estimation. *IEEE Transactions on Signal Processing*, *58*(10), 5016-5029. doi: 10.1109/TSP.2010.2053029

Choueifaty, Y., Coignard, Y. (2008). Toward maximum diversification. *The Journal of Portfolio Management*, *35*(1), 40–51. Retrieved from https://jpm.pm-research.com/content/35/1/40 doi: 10.3905/JPM.2008.35.1.40

Clarke, R. G., de Silva, H., Thorley, S. (2006). Minimum-variance portfolios in the u.s. equity market. *The Journal of Portfolio Management*, *33*(1), 10–24. Retrieved from https://jpm.pm-research.com/content/33/1/10 doi: 10.3905/jpm.2006.661366

Jain, P., Jain, S. (2019, 07). Can machine learning-based portfolios outperform traditional risk-based portfolios? the need to account for covariance misspecification. *Risks*, *7*, 74. doi: 10.3390/risks7030074

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*(301), 236–244. Retrieved from http://www.jstor.org/stable/2282967