BACHELOR'S DEGREE THESIS

Degree in Statistics

Title: Hierarchical Portfolio Optimization

Author: Francisco De Lio Pérego

Advisor: José Antonio González Alaustré

Department: Statistics and Operations Research (UPC)

Academic year: 2020-2021

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Acknowledgements

It is not often that one finds himself in a position to express his gratitude like this. To begin with I find myself with the disposition to embarrass myself a little and express my gratefulness not to anyone nor anything specific. One has to be grateful for the whole if he is to be grateful for any of its parts. So I shall be grateful to the root of all joy and beauty and suffering. Not because of anything, but in spite of anything. One who is at peace with the inner workings of the world needs no permission nor excuse to reap its sweetest fruits, and that's something to strive for.

On a more concrete and less theatrical note, I want to thank José Antonio González Alaustré for allowing me to go where I felt the wind blew and for his patience and consistent dedication as well as support across the 4 months during which this project was carried out. It has been a pleasure to have him as an advisor. I recognize this thesis not only as my work but as the work of all the people and circumstances in my life as well. I want to express all the gratitude to my lovely family and friends, as well as to the research community for enlightening my path through this project.

Lastly, to anyone who will be reading this thesis whether for curiosity, obligation or preferably both. Thank you.

Abstract

The field of Portfolio Optimization has historically had a very hard time as the Mathematical Models at its availability are based on certain assumptions one can not afford to make in the financial markets, making naive approaches all-too enticing. In this project we have introduced the assumption that the different stocks in the financial markets have a hierarchical structure and have allowed ourselves to be inspired by it to build portfolios through a Machine Learning approach. We have employed the Hierarchical Risk Parity algorithm and tested minor variations relating to the dissimilarity measure it makes use of. The tests were conducted with historical daily closing price data from 2014 to 2020 for 440 stocks in the S&P 500 index. Results suggest most of the tested Hierarchical Risk Parity variants are robust and can compete with the Equal Weights Portfolio. We mainly encourage the use of two dissimilarity measures, the standard one, a correlation based metric and Dynamic Time Warping. The former is suggested to the pessimistic investor while the latter to the hopeful yet conservative investor. To optimistic investors with a high risk tolerance the recommendation would be to use the traditional Equal Weights portfolio among the asset allocation methods considered in this project.

Keywords: Portfolio Optimization, Clustering, Time Series Analysis, Markowitz's Model, Hierarchical Risk Parity, Dissimilarity Measures, S&P 500.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

If a statistician had a nightmare it would be comprised of intricate random processes, fat-tailed distributions, unforeseeable paradigm-shifting events, highly chaotic non-linear relationships and an inability to reproduce experimental conditions. Just like the financial markets.

Statistics as a discipline generally relies on being able to simplify very complex empirical phenomena through naive yet clever probabilistic assumptions in order to make inferences or predictions. In the context of the financial markets the probabilistic assumptions one can realistically afford to make about the underlying phenomena are generally not simple nor practical, specially when building portfolios. Portfolios being collections of financial investments that can be formed by a variety of financial instruments. For the sake of this project we will limit the scope of portfolios to that of securities known as stocks and we will not allow the practice of shorting, essentially trying to benefit from stocks going down in price.

In the past 40 years several asset allocation methodologies have been proposed in the literature, but due to estimation and computational complications even the most well thought off methods struggle to perform better than naive methods, such as the equal weights portfolio. So far the seeds provided by the theory struggle to grow and flourish on the soils of reality. Here we will explore recent developments in portfolio optimization inspired by the field of Unsupervised Learning that work under the, credible and widely discussed in the literature, assumption that the stock returns follow a hierarchical relationship such that the topology of the stock market can not be meaningfully represented as as complete graph. We will apply some variations to the proposed method in order to relax its implicit assumptions and hopefully enhance its capabilities and make it more robust.

In the following thesis we will discuss modern portfolio theory and its pitfalls. We will explore alternatives that improve upon it, taking from fields ranging from Random Matrix Theory to Information Theory. The primary objective will be to explore Clustering, more specifically Hierarchical Clustering, as an alternative to allocate assets among a universe of stocks and see how it stacks up to more traditional methods in terms of realized risk and returns. For that we will conduct an empirical study using the daily returns of 440 stocks in the S&P 500 index from 2014 to 2021.

Chapter 2

Theoretical Framework

2.1 Times Series and their Characteristics

Time Series are a very special type of data which arises from experimental conditions where phenomena are observed at different points in time. As opposed to cross-sectional data, it has the very interesting property of being structured sequentially, with the implication that the order of the observations carries important information. The language of that structure, furthermore, can be viewed from two alternative, complementary, lenses; the time domain and the frequency domain. However, despite this additional information Times Series data provides, it can be a particularly tricky type of data to work with given the experimental conditions are in continuous development and structural changes in the data are commonplace. Something being a certain way in the past does not necessarily imply it will continue that way onto the future. Furthermore Time Series data is usually requires filtering to be usable and the temporal dimension, besides capturing the sequential structure of the data, encompasses a lot of sources of variation that can not be properly accounted for, which further obfuscates things.

A time Series y_t can be understood in terms of 4 fundamental components: the trend T_t, the baseline level at which the series is moving; the cyclical C_t, oscillatory movements around the trend that take place beyond the seasonal period; the seasonal S_t, which captures oscillatory movements inside the period of the time series; and the noise component E_t, which represents the random variation in the series that can not be attributed to the previous 3 components.

$$y_t = f(T_t, C_t, S_t, E_t) \tag{2.1}$$

The function f describes how the variability of the series evolves over time. For instance, if two components are linked over addition they will be homocedastic with respect to time, whereas if they are linked by multiplication those two components will be heterocedastic. In practice however it can be hard to discern between the trend and the cyclical component so they are usually regarded jointly as the trend-cyclical component TC_t.

2.1.1 Stochastic Process Representation

Let $(\Omega, \mathcal{F}, P)$ be a probability space and (S, Σ) a measurable space. A stochastic process [1] is a collection of random variables $\{X(t),\ t \in I\}$ that takes values in S and is indexed by time, represented by a given set I which can either be continuous or discrete. Time Series are usually modeled under the assumption of following discrete-time stochastic processes. A Time Series is to a a Stochastic Process what a coin toss is to a Bernoulli random variable.

Definition 1 *The autocovariance of a random process is defined, in the general case, as*

$$\gamma(t_1, t_2) := Cov(X(t_1), X(t_2)) = \mathbb{E}[X(t_1) - \mu_{t_1}, X(t_2) - \mu_{t_2}] \qquad t_1, t_2 \in I \tag{2.2}$$

Definition 2 *A discrete-time stochastic process $\{X_t,\ t \in \mathbb{Z}^+\}$ is stationary in a weak sense if it has the following properties*

1. $\mathbb{E}[X_t] = \mu_X \in \mathbb{R}, \forall t \in \mathbb{Z}^+$
2. $\mathbb{E}[|X_t|^2] < \infty,\ \forall t \in \mathbb{Z}^+$
3. $\gamma(t_1, t_2) = \gamma(|t_2 - t_1|, 0)$ *given* $t_1, t_2 \in \mathbb{Z}^+$

A stationary process in the weak sense will then have constant expected value and a finite second moment for all points in time, and lastly an autocovariance that does not depend on time per say but on the difference between the time points , also known as lag and denoted by h.

Weak stationarity is a very desirable assumption to be able to afford for Time Series. It ensures the structure of the series does not meaningfully change over time, which allows us to make good estimations and predictions.

(probably add partial correlation+ change definition + lemma in acf h)

Definition 3 *The autocovariance function of a weakly-stationary discrete-time stochastic process $\{X_t,\ t \in \mathbb{Z}^+\}$ is defined only in terms of the lag*

$$\gamma(h) = Cov[X_{t+h}, X_t] = \mathbb{E}[(X_{t+h} - \mu_X)(X_t - \mu_X)] \tag{2.3}$$

Definition 4 *The spectral density function of a weakly-stationary discrete-time stochastic process $\{X_t,\ t \in \mathbb{Z}^+\}$ is defined as*

$$f(\omega) = \sum_{h=-\infty}^{\infty} \gamma(h) e^{-2\pi i \omega h} \qquad |\omega| \leq 1/2 \tag{2.4}$$

Note that given weak stationarity the spectral density is the inverse Discrete Fourier Transform of the autocorrelation function. The unicity property of the Fourier Transform leads us to the following relationship

$$\gamma(h) = \int_{|\omega| \leq 1/2} f(\omega) e^{2\pi i \omega h} d\omega \qquad h \in \mathbb{Z} \tag{2.5}$$

This goes to show that the autocorrelation function and the spectral density carry the same information. While the autocorrelation function describes the stochastic process' sequential structure in terms of lags in the time domain the spectral density does so via frequencies in the frequency domain.

2.2 Unsupervised Learning

Unsupervised Learning is a branch of Statistical/Machine Learning where techniques are employed to infer the characteristics of the data without making use of reference variables to be modeled in terms of predictors. In the case of Unsupervised Learning the interest is on, given a set of N observations from random probability-vector X with a joint probability density $\Pr(X)$, modeling the properties of this density without the help of a supervisor providing correct answers and an incentive to decrease some measure of error in the form of a loss function.

2.2.1 Clustering

Clustering [2], also known as segmentation, is a form of Unsupervised Learning that has a variety of goals relating to grouping collections of objects in terms of a predefined notion of similarity. Similar objects should be placed in the same group, or cluster, while dissimilar objects should be placed apart. The clustering process, some say, is more of an art than a science as there is a strong subjective component to it. Selecting a good dissimilarity measure is not straightforward and neither is finding the most appropriate number of clusters to consider. Furthermore, different types of algorithms will have different philosophies with respect to how to understand the notion of 'cluster', so their approaches will vary significantly. The ideal way to segment a given dataset will vary depending on the research question at hand.

The clustering algorithms most commonly applied to structured data could be categorized in the following way [3, 4]:

1. **Centroid Based Clustering:** Each cluster is defined on the basis of a central vector, or centroid. The main preoccupation of this type of algorithms is finding a predefined number of points in space such that observations near to those points will be assigned to their cluster in a way that minimizes intra-cluster variation. Algorithms of this type include k-means, its many variants, Spectral Clustering and Affinity Propagation.

2. **Hierarchical Clustering:** This family of methods does not look for a specific partition to segment the dataset but provide an extensive hierarchy of clusters suggesting a big set of different partitions. After inspecting the resulting hierarchy, a partition is chosen out of the set of proposed partitions. Some examples include divisive clustering, agglomerative clustering and their many variants, Ward or BIRCH being some of them.

3. **Density-based Clustering:** In this approach to clustering, clusters are understood as high density areas, separated by low density areas that are considered noise. It is pioneered by algorithms like DBSCAN, OPTICS and MeanShift.

4. **Model-based Clustering:** Being the type of clustering more statistical in nature, it constructs clusters according to what distribution might have generated each observation. Each specific cluster will be assumed to come from a different distribution. The most important of model-based approaches is the Gaussian Mixture Model which given a set correlation structure and number of clusters converges to a solution through the EM algorithm.

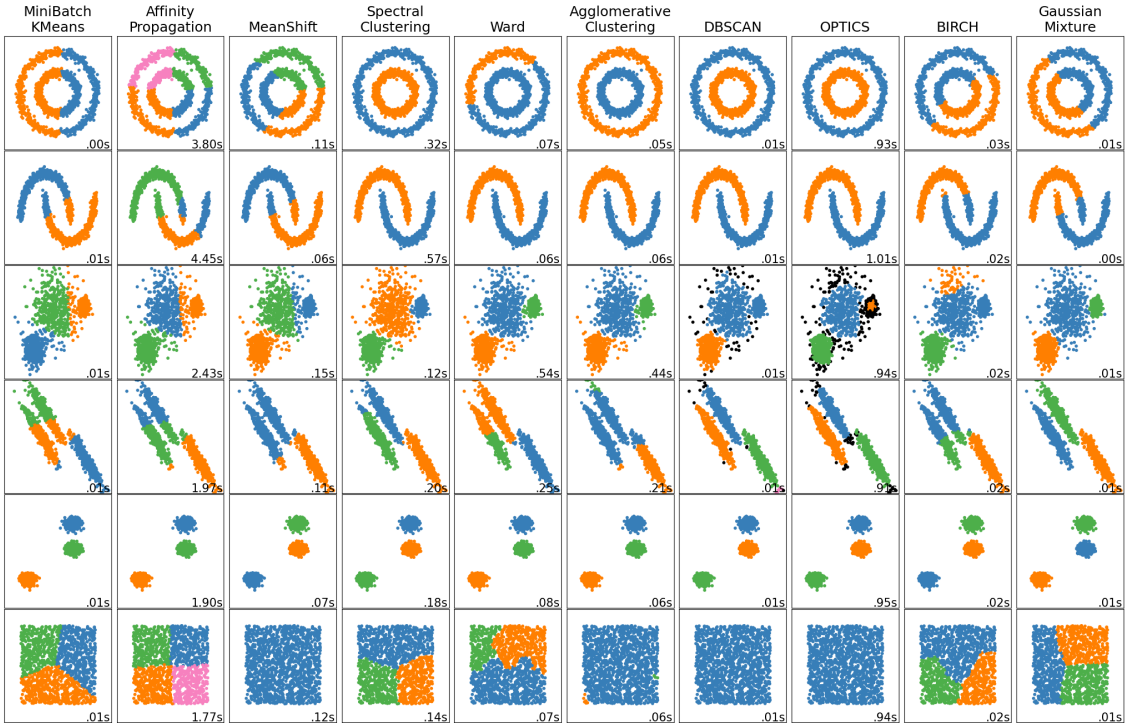


Figure 2.1: Demonstration of the clustering algorithms in Python's Scikit-learn.

Agglomerative Clustering

Agglomerative clustering [2] is central to this project and takes a bottom-up approach to hierarchical clustering, which does not require to pre-specify the number of clusters to construct. Each observation starts as an independent cluster, clusters are then merged two by two by minimizing pairwise inter-cluster distance, in a greedy fashion, until all observations are in the same cluster.

Given a set $S = \{1...N\}$ corresponding to the observations to cluster there will be a fixed number of $N - 1$ iterations. Suppose a dissimilarity matrix D whose elements d_{ij} are the pairwise distances between observations $i, j \in S \times S$. Now let G and H be two disjoint clusters formed by observations in S, agglomerative clustering defines an inter-cluster distance measure called linkage. The most commonly used linkages are the Single Linkage (SL)

$$d_{SL}(G, H) = \min_{i,j} \; d_{ij} \qquad i \in G, j \in H$$

which represents the minimum pairwise distance between elements of the two clusters; the complete linkage (CL)

$$d_{CL}(G, H) = \max_{i,j} \; d_{ij} \qquad i \in G, j \in H$$

which is the maximum pairwise distance between elements of the different clusters; and the average linkage (AL)

$$d_{AL}(G,H) = \frac{1}{|G||H|}\sum_{i\in G}\sum_{j\in H} d_{ij} \qquad i \in G, j \in H$$

the average of all pairwise distances between all the elements of the different clusters.

The algorithm will then start with N distinct clusters and merge iteratively the clusters that minimize the linkage function until only one cluster is left. This process can be conveniently visualized via dendograms, graphs that show what clusters merge at each individual iteration and, at the perpendicular axis, the value of the linkage function at the point of each merger. The dendogram can be a good tool to guide the choice of the number of clusters, as the greater the linkage function at a merger, the greater the difference between the clusters.

Single linkage will have a tendency to combine, at low thresholds, observations linked by a series of close immediate observations and result in diametrically large clusters, a property that generally is not attractive. Complete linkage represents the opposite extreme, it will construct compact clusters with small diameters, but it will not be unusual for particular observations to be much further apart from their respective clusters than from some other one, a surely undesirable property. Average linkage presents a compromise between single and complete linkages, and presents a more measured approach. It results in relatively compact clusters that are relatively far apart, but it still comes with its drawbacks, the main one being non-invariance with respect to monotonic scaling transformations.

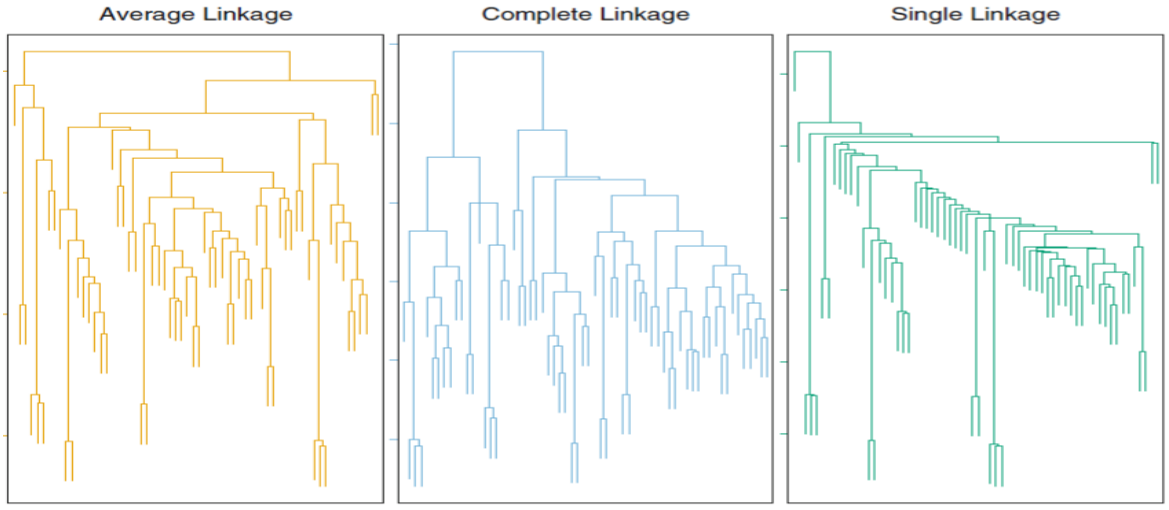


Figure 2.2: Dendograms built through the discussed linkages, from Elements of Statistical Learning

2.2.2 Dimensionality Reduction

Dimensionality reduction techniques are also a form of Unsupervised Learning. The focus is on transforming data from a high-dimensional space to a lower-dimensional

representation that preserves meaningful properties of the original data. It is very useful for feature engineering, visualization of high-dimensional data and general information compression that removes redundancy and improves the signal-to-noise ratio. The more popular approaches include Factor Analysis in its many variants, Manifold Learning Techniques like Kernel-PCA or Isomap and Artificial Neural Networks such as Autoencoders.

Principal Component Analysis

Principal Component Analysis, more commonly referred to as PCA, is by far the most well known Dimensionality Reduction technique. Consider a set of observations $\{x_n\}$ where $n = \{1, ..., N\}$ and each observation x_n is a variable of dimensionality D. PCA's goal is usually finding an orthogonal linear projection of the data to a space of dimension $M \leq D$ while maximizing the variance or information in that lower dimensional representation. We will now consider a projection to a space where $M = 1$ and a D dimensional unit vector u_1 [5]. Given a covariance matrix $\mathbf{S}$ we want to find the directions u_1 associated to the D variables such that the quadratic form representing the variance of the D variables considered jointly is maximized. With the restriction that the vector u_1 has to be unitary.

$$\begin{aligned} \max_{u_1} \quad & u_1^t \mathbf{S} u_1 \\ \text{s.t.} \quad & u_1^t u_1 \end{aligned} \tag{2.6}$$

This problem is solved in via its dual formulation in a straightforward way

$$\max_{u_1, \lambda_1} \quad \mathcal{L}(u_1, \lambda_1) = u_1^t \mathbf{S} u_1 + \lambda_1 (1 - u_1^t u_1) \tag{2.7}$$

By setting the gradient of u_1 equal to 0 we find that there is a stationary point when u_1 and λ_1 are respectively an eigenvector and its associated eigenvalue of $\mathbf{S}$

$$\mathbf{S} u_1 = \lambda_1 u_1 \tag{2.8}$$

From this it follows that $u_1^t \mathbf{S} u_1 = \lambda_1$. The variance in the direction u_1 will then be equal to its eigenvalue and so the eigenvector we are specifically looking for will be associated to the biggest eigenvalue.

$\mathbf{S}$ is a real symmetric matrix and its eigenvalues will always be orthogonal. In the general case where we wish to project to a general dimension $M \leq D$ the $u_2, ..., u_m$ principal components can simply be defined in terms of the $M - 1$ eigenvectors with the largest eigenvalues other than u_1. However the principal components other than u_1 can be chosen arbitrarily such that $u_1, ..., u_m$ provide an orthogonal basis.

One of the most attractive attributes of PCA is that one can tell how much information from the original data each one of the new latent variables represents. The proportion

of the information in the original data, measured in terms variance, contained alongside the direction u_i where $u_i \in \{1, ..., D\}$ is given by the following expression

$$\frac{\lambda_i}{\sum_{k=1}^{D} \lambda_k} \tag{2.9}$$

2.3 Times Series Clustering

Time Series and Clustering are, on a practical level, some of the least tractable areas in Statistics. In their intersection lies Time Series Clustering, a research area which combines both the mathematical richness and practical complications in both of its progenitors. Additional challenges posited to Time Series Clustering by the nature of the problems it tends to tackle are high dimensionality, very high feature correlation and concerningly low signal to noise ratios.

In Time Series Clustering [6], the baseline algorithms are usually borrowed from clustering, while the dissimilarity measures employed are inspired by Time Series Analysis and its sibling, Signal Processing. Model-based approaches are much more varied and are usually based on derivatives of ARIMA and Hidden Markov Models (refs would be cool*). The euclidean distance, by far the most popular dissimilarity measure when it comes to regular clustering problems, is not considered viable where observations are Time Series, which makes k-means lose its favor when compared to k-medioids and Ward's linkage lose its appeal. Dissimilarities based on Fourier and Wavelet analysis are popular given the connection between weakly stationary Time Series and the frequency domain.

An interesting and very popular clustering paradigm that emerges is the shape-based approach which intends to find Time Series, or Signals, that behave similarly irrespective of their phase component. It measures the relationship between two Time Series without consideration to the time of occurrence of patterns, but the patterns themselves. Dissimilarity in time is a special case of dissimilarity in shape. Dynamic Time Warping is at the core of shape-based clustering, and is very celebrated dissimilarity measure as it is based on a very meaningful conception of similarity that is very much appropriate to Time Series, and is robust to random fluctuations in periodicity in a way that other dissimilarities are not. Another clustering approach that is perhaps not unique to Time Series Clustering, but is the most notable inside its scope is feature based clustering, where objects are clustered in terms of dissimilarities constructed through a variety of statistics that attempt to represent their fundamental characteristics in a more compact way.

More standard time domain dissimilarities, based on autocovariances and cross-covariances, have of course found success in Time Series Clustering and by courtesy of the field of econometrics, granger causality and transfer entropy have naturally also found successful application to measuring association between Time Series.

2.4 Portfolio Optimization

2.4.1 Markowitz's Model

Before Markowitz's contributions [7] assets were analyzed individually in order to construct a portfolio and the relationship between risk and returns was not well understood. Markowitz proposed that to model the market properly we should not consider the assets independently but we should do so holistically, accounting for how assets interplay with each other. He also showed that diversification is generally very useful when it comes to finding optimal portfolios and formalized volatility as the standard deviation of returns.

Let us define

1. $P_{t,i}$ is the price of stock i at time t.
2. $R_{t,i}$ are the returns of stock i at time t, defined as $\frac{P_{t,i}-P_{t-1,i}}{P_{t-1,i}}$
3. μ is the vector of expected returns where $\mu_i = \mathbb{E}(R_{*i})$
4. $\mathbf{\Sigma}$ is defined as $\mathbb{E}\left[(R-\mu)\otimes(R-\mu)\right]$, the covariance matrix of the stocks.
5. w are the weights of the portfolio where w_i is the proportion of the capital in the portfolio to be trusted to stock i.
6. N is the number of stocks in the portfolio.

Under the assumption that investors want to find the perfect compromise between minimizing risk and maximizing return Markowitz posed the mean-variance optimization problem with the purpose of finding the minimum variance portfolio for a set profitability, r :

$$\begin{aligned} \min_{w} \quad & w^t \mathbf{\Sigma} w \\ \text{s.t.} \quad & w^t \mu = r \\ & w^t 1_N = 1 \\ & w \geq 0 \end{aligned} \tag{2.10}$$

This restricted optimization problem is easily solved on its dual formulation by solving for the associated Lagrangian, where the lagrange multipliers $\lambda_1, \lambda_2 \geq 0$

$$\mathcal{L}(w, \lambda_1, \lambda_2) = w^t \mathbf{\Sigma} w + \lambda_1 (w^t \mu - r) + \lambda_2 (w^t 1_N - 1) \tag{2.11}$$

Taking the partial derivatives of $\mathcal{L}$ and setting them equal to 0 leads us to the following system of equations, represented in matrix form

$$\underbrace{\begin{bmatrix} 2\boldsymbol{\Sigma} & \mu & 1_N \\ \mu^t & 0 & 0 \\ 1_N^t & 0 & 0 \end{bmatrix}}_{\boldsymbol{\Phi}} \begin{bmatrix} w \\ \lambda_1 \\ \lambda_2 \end{bmatrix} \approx \underbrace{\begin{bmatrix} 0 \\ r \\ 1_N \end{bmatrix}}_{y} \tag{2.12}$$

Which has the following solution, that is guaranteed to be a minimum given the objective function is a convex quadratic form as per the symmetry and positive semidefinitness of $\boldsymbol{\Sigma}$

$$\begin{bmatrix} w^* \\ \lambda_1^* \\ \lambda_2^* \end{bmatrix} = \boldsymbol{\Phi}^{-1} y \tag{2.13}$$

Those portfolios that have the minimum risk for a particular return are known as efficient portfolios. In the cartesian space formed by the volatilities and returns, the respective volatilities and returns of the efficient portfolios form what's known as the efficient frontier, which is a good representation of the relationship between the return of a portfolio and its volatility.

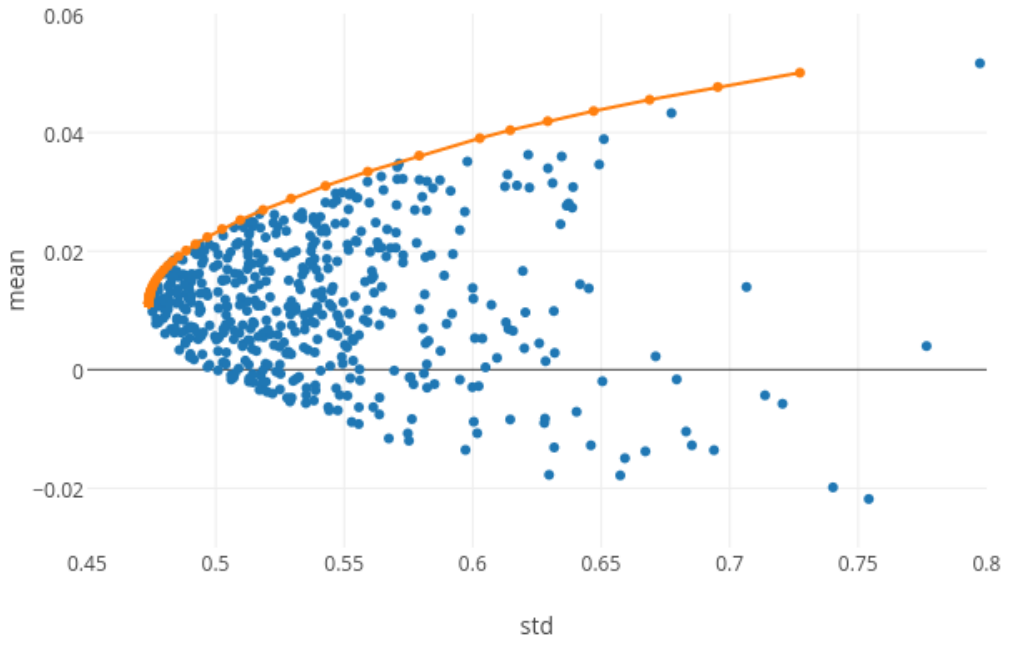


Figure 2.3: Simulated portfolios in terms of their risk and returns. The orange region conforms the efficient frontier

2.4.2 Major Criticisms of Mean-Variance Optimization

Markowitz's model could seem at glance like it would have been the silver bullet to solve the problem of asset allocation entirely. Turns out it consistently overpromises and therefore underdelivers with respect to both returns and risk. Markowitz's theory rests on many modelling assumptions one simply can not afford to make about the financial markets and, furthermore, the numerical computation of the weights is generally poorly conditioned [8].

The most problematic presupposition is that the universe of stocks we are considering in the optimization problem is completely determined by its mean vector as well as its covariances, which implies that stocks' returns can be represented as time invariant and stationary stochastic processes formed by sequences of identical independently distributed random variables. Of course that's very restrictive and reductionist. Stocks' returns are often not stochastically independent nor stationary, and their relationships are beyond the scope of linearity. Most importantly they suffer from many structural breaks, they are not ergodic time series, the statistical properties of the returns can not easily be deduced. The estimation of expected returns as well as risk and covariances will thus not represent the underlying market processes but in a very reductionist sense and will have a lot of noise associated, which is specially unfortunate given in the financial markets the signal to noise ratio is already notoriously low. The main limitation from the computational front is that quadratic programming algorithms we'll rely on to get the solution of the optimization problem require the numerical inversion of the covariance matrix of the returns, $\mathbf{\Sigma}$. This matrix will generally have a very big conditioning number with the implication that inverting it will come at a cost of non-trivial calculation errors which will add to the already present estimation errors.

Because of the aforementioned complications not only will the portfolios constructed via this model will underperform out of sample but the assigned weights will be very sensitive to the estimation of the expected returns. Small changes in their estimation will oftentimes result in drastically different portfolios, which is a very undesirable property, particularly given the significant errors that are made when estimating the expected values. The mean-variance portfolio will also fail to diversify, as it will reserve the non-zero weights to a minority of the stocks considered.

2.4.3 Alternatives to Classical Mean-Variance Portfolios

Some interesting alternatives to Markowitz's model include the Black-Litterman model [9], founded in bayesian statistics it allows for the merging of both investor insights and market knowledge and the data used to fit the model. The Black-Litterman model has been applied with success and the prior information added to the model makes the outputed weights much more robust than those of classic mean-variance approaches. We will not explore this method further as we lack the financial domain knowledge to select meaningful bayesian priors.

Random Matrix Theoretic results [10] have been applied in order to denoise the co-

variance matrix [11]. They've been shown to be very effective at reducing the noise in the estimations of Σ via singling out the eigenvalues of the correlations matrix which represent noise to be taken out. As one would expect, using the denoised estimations of the covariance matrix to build portfolios improves their out of sample performance [12].

A Volatility Allocation approach became popular in the light that using the expected returns in order to construct portfolios resulted in very unstable weights and severe under-performance. This approach consists on dropping the expected returns from the analysis altogether and focusing on the estimated volatility alone when it comes to allocating the assets. Empirical evidence has been found to suggest that the efficient frontier as shown by mean-variance optimization does not generalize that well out of sample and that building a minimum variance portfolio unrestricted by returns will generally give substantially greater returns than the efficient frontier would suggest.

On the famous "Cluster analysis for portfolio optimization" [11] the authors, inspired by Econophysics, used a correlation based metric distance to represent the dissimilarity of stocks and applied hierarchical clustering procedure in order to find a filtered correlation matrix. Several papers have noted the hierarchical structure of stocks. They found applying Markowitz's model with their filtered correlation matrix, via average linkage, significantly improved results of the constructed portfolios. Lately there's been a resurgement of the interest in clustering based asset allocation algorithms, particularly ones that work within the assumption of a hierarchical structure on the financial markets [13, 14].

Formulating the Volatility Allocation Problem

The problem can be stated as follows:

$$\begin{aligned} \min_{w} \quad & w^t \mathbf{\Sigma} w \\ \text{s.t.} \quad & w^t 1_N = 1 \\ & w \geq 0 \end{aligned} \tag{2.14}$$

This optimization problem [15] can very easily be solved in the same vein as mean-variance optimization, via the dual formulation. The solution, which is guaranteed to be a minima, is the following:

$$w^* = \frac{\mathbf{\Sigma}^{-1} 1_N}{1_N^t \mathbf{\Sigma}^{-1} 1_N} \tag{2.15}$$

1. For a covariance matrix $\mathbf{\Sigma}$ with all equal entries the solution to the problem will be $w^* = 1_N/N$, known as the equal weights, or naive, portfolio.

2. For a diagonal covariance matrix $\mathbf{\Sigma}$ with heterogeneous variances the optimal portfolio will be the inverse-variance portfolio as $w_i^* = \frac{1/\sigma_i^2}{\sum_{n=1}^{N} 1/\sigma_n^2}$. This will be important later on.

3. For a covariance matrix $\mathbf{\Sigma}$ with heterogeneous variances and covariances we will call the solution the minimum variance portfolio.

It's important to stress the relevance of these portfolios. Despite their simplicity and their disregard of the expected returns they produce stable weights, do a good job at diversifying and are surprisingly hard to beat. Specially in the case of the naive portfolio, which routinely outperforms mean-variance optimization [16]. As discussed earlier their place in the efficient frontier has been observed to be in practice significantly higher than Markowitz's model predicts when it comes to their returns.

2.4.4 Performance Measures

There are many criteria one could employ when measuring how good a portfolio is. Before Markowitz the main and only criteria considered where the predicted returns. There was not a formal notion of risk nor an appreciation for what a diversified portfolio could do. Markowitz's criteria, as we know, was to set level of risk one could reasonably afford and construct a portfolio that would give you the most bang for your risk. Besides the returns $r'w$ or the volatility as Markowitz defined it, $w'\mathbf{\Sigma}w$, the performance measures that have been more influential historically are arguably the Sharpe Ratio, the Value-at-Risk and the Conditional Value-at-Risk.

Sharpe Ratio

William Sharpe [17] suggested a more general metric to evaluate the performance of a portfolio, which simultaneously takes into account the overall returns of a portfolio and its variability, assuming normally distributed identical and independent returns and accounting for a risk-free rate of interest.

$$S_\phi = \frac{r_\phi - r_f}{\sigma_\phi} \tag{2.16}$$

where ϕ represents the portfolio being evaluated, r_ϕ and σ_ϕ its overall return and standard deviation respectively and r_f is the best rate of return achievable through an investment with no perceived risk. The Sharpe Ratio is motivated by the implicit assumption that $S_\phi \sim N(0,1)$, which is usually not reasonable.

Value-at-Risk

More often referred to as VaR, it was motivated by the desire for a performance measure that was non-necessarily-parametric, did not depend on the return and so could be employed to evaluate portfolios a priori under volatility allocation approaches, and took into account only the scariest part of the variance, the draw-down variance, that of the far left tail of the distribution of the portfolio's daily returns. Given a distribution of the daily portfolio overall return X, its quantile function F_X^{-1} and a level $\alpha \in (0,1)$, VaR is generally defined as

$$VaR_\alpha(X) = F_X^{-1}(\alpha) \tag{2.17}$$

and represents the value for the daily portfolio returns in the quantile α. $VaR_\alpha(X)$ will then be the expected minimum loss given a portfolio active for $(1-\alpha)*100$ days. However, the VaR is often discouraged is it is not a coherent risk measure [18].

Conditional Value-at-Risk

Better known as Expected Shortfall, or just CVaR, was developed as a response to VaR not having all the mathematical properties appealing in a risk measure and not taking accounting for the losses once the $VaR_\alpha(X)$ is breached, when the portfolio is at its worst, precisely when a down-ward risk measure is most crucial. It is defined as

$$ES_\alpha(X) = \mathbb{E}[X|X \leq VaR_\alpha(X)] \tag{2.18}$$

and represents the expected loss below the quantile α.

Chapter 3

Methodology

3.1 The Stock Market and its Hierarchy

Financial Markets are well defined complex systems that we can only hope to model through the theory of stochastic processes. The stock market can be understood as a Multivariate Time Series, however that multivariate structure is very hard to exploit. As we have established stock returns are particularly pathological among Time Series data, which already stands out for being hard to treat. Stock returns' fluctuations are very hard to model individually via either parametric, non-parametric or semi-parametric approaches. One would think our understanding of the Stock Market would benefit from the additional information provided by a Multivariate Time Series approach, one would probably be right, however, modeling the underlying dynamics between the different stocks in way that would make such approach useful is a very hard problem because of the insurmountable noise that is immanent to the stock market. Consideration should then be paid to inferring the topological structure of any given big set of stocks, which is to say, how would the connections between the stocks be represented on a graph. One thing is to be sure, the way in which the stock market is interconnected ought generally not to be represented as a complete graph because of 2 reasons, the former being the main one; first of all it is unrealistic, fluctuations in stocks from completely unrelated industries will not be directly influencing each other but through connections with other stocks; second of all a complete graph representation is very complex and ill-conditioned to analysis, there is too many moving parts as to extract from it any meaningful understanding or stable predictions.

Rosario Mantegna [19] hypothesized the stocks are structured hierarchically, and that looking at that structure could be helpful when modeling the financial markets. The study employs a correlation based metric, designed to represent the pairwise influence between stocks, and given 443 stocks in the S&P index built an associated distance matrix. Subsequently, a Minimum Spanning Tree is computed on the graph whose adjacency matrix is the calculated distance matrix. The result is a parsimonious hierarchical tree that represents the ability of stocks to interact with each other, directly or indirectly. The paper found the resulting tree found a meaningful economic taxonomy such that the stocks where usually grouped in branches of the tree alongside stocks from companies in the same industry.

3.2 Hierarchical Risk Parity

Hierarchical Risk Parity (HRP) is an heuristic algorithm, proposed by Marcos López de Prado [13], that intends to build portfolios in a way that sidesteps the main problems tormenting Modern Portfolio Theory. Inspired by the hypothesized hierarchical structure of the financial markets, and based on Hierarchical Clustering, it makes the simplifying assumption that the direct pairwise relationships between different stocks are not all existent such that not all stocks are interchangeable, and the connections between most stocks are indirect and dependent on a chain of other stocks to relate them. That imposed structure has the implications that similar stocks will compete only among each other and whatever estimation errors we have, those errors won't propagate in a chaotic fashion and the weights of our portfolio will be more stable. Small changes in the input will not correspond to large changes in the output, in contrast with mean-variance optimization, where weights are left to vary freely in unintended and unintuitive ways. The stability of the weights is further improved by the dispensation of the returns when it comes to building the portfolio. As discussed earlier, optimization solvers produce solutions that are extremely sensitive to the inputted estimation of the future returns. That is unacceptable given the fact that predicting future returns is a very hard problem and is bound to be done with a very sizeable out-of-sample error, which means the in-sample solution to the portfolio weights is bound to be drastically different to the out-of-sample solution, the optimum that we are actually interested in.

Another major advantage of Hierarchical Risk Parity is the fact that it does not require the inversion of a covariance matrix $\mathbf{\Sigma}$, specially given covariance matrix of sets of stocks are usually very badly conditioned and are not necessarily normal. That leads to substantial numerical errors that increase fast alongside the number of stocks considered for the portfolio. It is very unfortunate that with multicollinear stocks the need for diversification increases, as does the condition number of the covariance matrix increases. When portfolio optimization is most critical, non-linear optimization algorithms will give the more unstable, and sub-optimal solutions.

There's been many attempts to use Clustering in order to build well-performing portfolios. Many of those approaches require to find a sensible partition according to which segmenting the stocks. It is not uncommon to find an underwhelming optimal number of clusters according to Shilouette or intra-variance criterias that would not be too useful in this kind of application. HRP allows us to consider not just a fix number of clusters but the whole set of its subclusters.

Some other important points are that HRP does not fail to diversify, unlike Markowitz's model, which tends to concentrate the entirety of the invested capital in a small number of the candidate stocks, and that the algorithm is not limited in its scope to working with covariances as the codependence measures between stocks but can easily be tweaked to work with much more general relationships that could capture in more useful ways what it means for two stocks to be dissimilar in the context of portfolio optimization.

The HRP algorithm has three distinct steps, tree clustering, quasi-diagonalization and recursive bisection which we will detail next.

3.2.1 Tree Clustering

Consider a multivariate time series comprised of the returns of N stocks for a number of T points in time, with a matrix representation of $R_{T\times N}$ and a covariance matrix $\mathbf{\Sigma}$. We compute $\mathbf{P} = \big(diag(\mathbf{\Sigma})\big)^{-\frac{1}{2}}\mathbf{\Sigma}\big(diag(\mathbf{\Sigma})\big)^{-\frac{1}{2}}$, the correlation matrix, with elements ρ_{ij}.

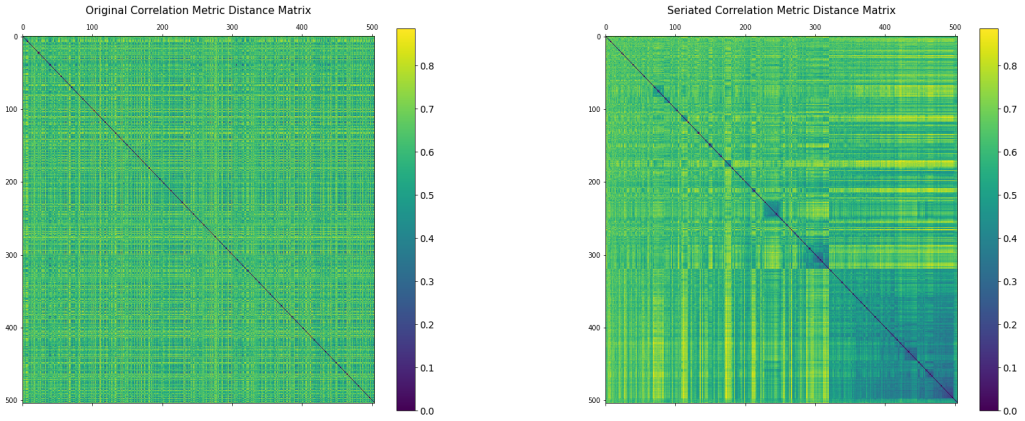
We define a distance measure $d : (R_{*i}, R_{*j}) \to \mathbb{R} \in [0, 1]$ where $i, j \subset M$, the cartesian product of the set of stocks, $\{1, ..., N\}$. $d_{ij} = d(R_{*i}, R_{*j}) := \sqrt{\frac{1}{2}(1 - \rho_{ij})}$. The distance d will measure the similarity of 2 stocks in terms of correlations. Stocks will be the closer together the stronger their linear relationship is, assuming it is positive, while they will be further apart the stronger their linear relationship is, assuming it is negative. The choice of d implies the philosophy that stocks are similar if they tend to move in the same direction, and dissimilar if they tend to move in the opposite direction which is one way of looking at it.

Next we define the distance matrix D, constituted by the distances d_{ij} as elements, and we apply the euclidean distance between all pairs of its column vectors. More formally we define a new distance measure $\widetilde{d} : (D_{*i}, D_{*j}) \to \mathbb{R} \in [0, \sqrt{N}]$ where $i, j \subset M$. Then we will get a new distance matrix $\widetilde{D}$ formed by the elements $\widetilde{d}_{ij} = \widetilde{d}(D_{*i}, D_{*j}) := \sqrt{\sum_{n=1}^{N}(d_{ni} - d_{nj})^2}$. This new distance of distances measures dissimilarity in terms of how 2 stocks move with respect to the universe of stocks considered. Two stocks are more similar the more similar their relationships are with the universe of stocks. We will proceed to apply the hierarchical clustering algorithm with single linkage over the distance matrix $\widetilde{D}$. We will find clusters of stocks not on the basis of how similarly stocks move with each other, in terms of direction, but how similarly they move with respect to the market as a whole.

3.2.2 Quasidiagonalization

Better known as matrix seriation, this step aims to rearrange the indexes of the stocks in the covariance matrix Σ in a way that it will become closer to a diagonal matrix, and so the largest covariances will lie along the diagonal. This is achieved through the resulting tree from the previously used hierarchical clustering procedure. We traverse the tree from top to bottom. Each time 2 branches merge we find a partition for 2 sets of indexes, S_1 and S_2, of sizes n_1 and n_2 respectively. We assign the n_1 indexes in subset S_1 to the left and the n_2 in the subset S_2 to the right of an array of length $n_1 + n_2$. The 2 subsets are locked in position to the specific positions of the orderings array of length N that are being contested by the two sub-clusters depending on the sub-clusters before them. We do this recursively for each branch until there is no subsets left to traverse.

Similar investments will be placed together while dissimilar ones will be placed apart.



(a) Original Correlation Metric Distance Matrix (b) Seriated Correlation Metric Distance Matrix

Figure 3.1: Before and After Quasidiagonalization

3.2.3 Recursive Bisection

The core and final step of Hierarchical Risk Parity takes advantage of hierarchical tree from step 1 and the seriated covariance matrix in step 2 to allocate the portfolio weights. We will traverse the dendogram with a top-down approach and assign the weights as if it were recursively at every split. The algorithm proceeds as follows:

1. The procedure is initialized.
   (a) all the weights are set to 1, $w = 1_N$
   (b) we set a list of items: $L = \{L_0\}$ where $L_0 = \{1, ..., N\}$
2. If $|L_i| = 1, \forall L_i \in L$, then **STOP**.
3. For each $L_i \in L$ such that $|L_i| > 1$:
   (a) Bisect L_i into two subsets of equal cardinality where $L_i^{(1)} \cup L_i^{(2)} = L_i$. The subsets correspond to the clusters originated from the bisection.
      i. Define the provisionary weights for the subset L_i^j, with an associated covariance matrix $\Sigma_i^{(j)}$

$$\widetilde{w}_i^{(j)} = \frac{diag[\Sigma_i^{(j)}]^{-1}}{trace\big(diag[\Sigma_i^{(j)}]^{-1}\big)} \tag{3.1}$$

Here we exploit the fact that for a diagonal covariance matrix, the inverse-variance portfolios are optimal. It makes sense to use inverse-volatility allocation for this subset since we are dealing with a quasi-diagonal covariance matrix.

ii. Define the variance of each cluster $L_i^{(j)}, j = 1, 2$, as

$$\widetilde{V}_i^{(j)} \equiv (\widetilde{w}_i^{(j)})^t \Sigma_i^{(j)} \widetilde{w}_i^{(j)} \tag{3.2}$$

(b) Compute the weighting factor

$$\alpha_i = 1 - \frac{\widetilde{V}_i^{(j)}}{\widetilde{V}_i^{(j)} + \widetilde{V}_i^{(j)}} \tag{3.3}$$

Which represents the weight that the cluster $L_i^{(1)}$ would be assigned, as a whole, while competing against cluster $L_i^{(2)}$ under the inverse-variance allocation paradigm.

4. Update the weights of the stocks in subsets 1 and 2 respectively

$$w_k = \alpha_1 * w_k, \ \forall k \in L_i^{(1)} \tag{3.4}$$

$$w_s = \alpha_2 * w_s = (1 - \alpha_1) * w_s, \ \forall s \in L_i^{(2)} \tag{3.5}$$

5. Return to **Step 2**.

At the end of the process, marked by 'stop' at step 2, we are left with a vector w containing the weights associated to any given stock.

3.3 Alternative Dissimilarity Measures

During the Tree Clustering stage in the Hierarchical Risk Parity algorithm two dissimilarities are used. First a correlation-based metric. Then a distance of distances based on that first dissimilarity. Variations of HRP can be made via changing that first dissimilarity. Here we will present alternative dissimilarities we will integrate into HRP with the intent of improving its performance. Each of the four suggested dissimilarities has distinct virtue that makes it stand out. The feature-based metric, for instance, intends to take advantage of the information the series of returns carry in an indirect way, through its fundamental characteristics. Dynamic Time Warping, meanwhile, is a shape based dissimilarity and should be able to detect similarities between series despite them being out of phase or having a random variations in periodicity. As we know the covariance matrix of the returns are very noisy, the Constant Residual Eigenvalue method is used to build a correlation-based metric based on a denoised correlation matrix. Lastly, the main selling point of the mutual information-based metric is that it should allow HRP to be sensitive to non-linear relationships.

3.3.1 Feature-based Metric

As discussed, feature-based approaches to clustering occupy a specially relevant place in the realm of Time Series Clustering given the inherent high dimensionality of time series data. The idea is representing the series via several of their fundamental characteristics via the so called features, which are essentially statistics in the sense that they are functions of the sampled time series. Some examples of features that can be extracted from a time series are the mean, the variance and the number of structural breaks. Many of those characteristics might be providing the same information, just in different ways.

Summarizing time series in terms of features can be thought of very loosely as a way of dimensionality reduction or information compression. However the resulting features will not be independent from each other and many will be conveying the same information. The information compression the feature based approach makes is intra-series, and not inter-series, in contrast it with dimensionality reduction techniques such as PCA, kernel PCA or Autoencoder Neural Networks.

In the particular feature-based measure we will employ, in order to give the same weight to all the distinct information conveyed by the initial characteristics extracted from the series, their information will be compressed via Principal Component Analysis, and we will engineer a new orthogonal set of features, thus not leaving place for redundancy.

Given N stocks, a time period T, a number of features K and a multivariate time series of the stock's returns $\mathbf{R}_{T\times N}$ we compute the feature matrix $\mathbf{F}_{N\times K}$ whose elements f_{ij} will represent the value associated to the j-th feature for the return series associated to the i-th stock.

Suppose a time index $t \in \{1, ..., n\}$ and a time series $y_t = TC_t + S_t + E_t$, with an additive integration scheme such that it is weakly-stationary; where TC_t, S_t and E_t are, respectively, the trend-cyclical, seasonal and noise components. To end with the notation, $\gamma(k)$ is the autocorrelation at lag k and $\hat{f}(\lambda)$ the estimated normalized spectral density. The features employed [20] are numerous and varied, here they are briefly presented:

Trend $= 1 - \frac{Var(E_t)}{Var(X_t - S_t)}$. Indicates how strong is the trend-cyclical component in the series.

Spike $= 1 - \frac{Var(E_t)(n-1)-(E_t-\bar{E}_t)^2}{n-2}$. Mesures the "spikiness" of time series data.

Linearity $= \beta_1^*$ and **Curvature** $= \beta_2^*$, where they are the non-intercept parameters that give solution to the linear model $T_t = \beta_0^* + \beta_1^* t + \beta_2^* t^2$ through the Ordinary Least Squares approach.

xACF1 $= \gamma(1)$. Value taken by the first lag.

xACF10 $= \sum_{k=1}^{10} \gamma(k)^2$. Sum of squares of the autocorrelations for the 10 first lags.

Spectral Entropy $= -\int_{-\pi}^{\pi} \hat{f}(\lambda) \log(\hat{f}(\lambda)) d\lambda$. Measures how hard a time series is to forecast. Takes values between 0 and 1. Higher values correspond to lower

signal to noise ratios.

Stability and **Lumpiness** These features are based on non-overlapping windows. Stability is the variance of the means of those windows, while Lumpiness is defined as the variance of their variances.

After calculating the feature matrix **F** with the R package 'tsfeatures' [21] we perform PCA not to reduce dimensionality but in order to separate the underlying information in the original features in orthogonal components such that we avoid redundancy in the information carried by the original features when doing the clustering. Then we calculate the pairwise euclidean distance between the PCA's projections' observations, which are associated to stocks, and get our sought after dissimilarity matrix. As we do not standardize the projections before calculating the euclidean distance matrix the weight of any principal component in the clustering will be proportional to the variability associated with that component. The more information a dimension carries about the original features the more voice it will have.

3.3.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is one of the most popular algorithms to measure similarity between temporal sequences which may vary in either speed, phase, or both. Being at the core of shape based clustering it considers dissimilarity in terms of occurrence patterns irrespective of their time. It allowed for significant contributions in speech recognition and robotics, before being adopted by statisticians for Time Series Analysis [22].

Algorithm 1: Basic Dynamic Time Warping

Result: $\mathcal{M}_{nm}$
Initialize with x and y, vectors of lengths n and m respectively;
Set a two-dimensional array $\mathcal{M}_{n\times m}$ with entries := ∞ ;
$\mathcal{M}_{00}$:= 0;
for *i:= 0 to n* **do**
  for *j:= 0 to m* **do**
    cost:= $|x_i - y_j|$;
    $\mathcal{M}_{ij}$:= cost+min($\mathcal{M}_{i-1,j}, \mathcal{M}_{i,j-1}, \mathcal{M}_{i-1,j-1}$);
  end for
end for

To this procedure is usually added a time window constraint such that only nearby time points can be matched [23]. The biggest drawbacks of Dynamic Time Warping are its sensitivity to noise and its computational complexity of $\mathcal{O}(n^2/\log\log(n))$.

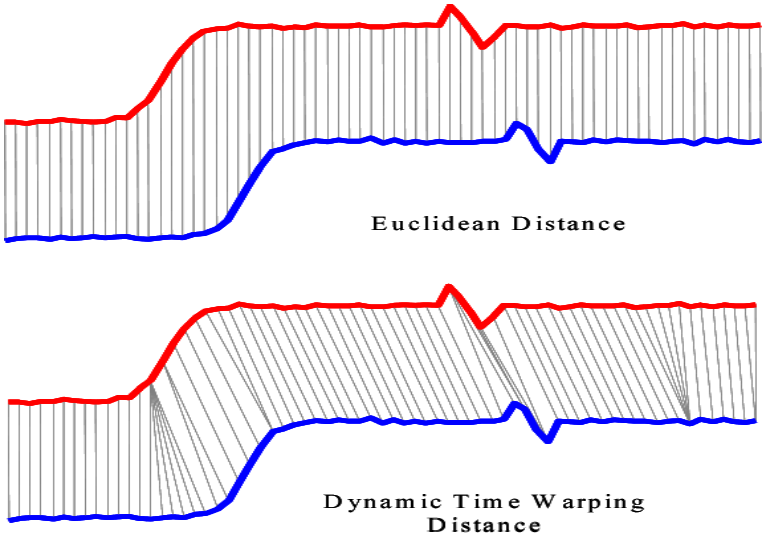


Figure 3.2: Comparison between Euclidean Distance and Dynamic Time Warping.

The dissimilarity matrix was obtained through the R package 'TSclust' [24].

3.3.3 The Constant Residual Eigenvalue Method

Random Matrix Theory, originally developed in the context of nuclear physics, has found widespread application in mathematical finance as studying the properties of random matrices has proven really useful when it comes to discerning the signal from the noise in the relationships among a universe of assets.

Suppose a correlation matrix $\mathbf{P}$ of the returns of N assets across a time period of length T, and assume those returns are centered, scaled, stochastically independent and identically distributed. A special case of the Marcenko-Pastur Theorem [10] states that the eigenvalues λ of $\mathbf{P}$, as $N \to \infty$ and $T \to \infty$ in a fixed ratio $Q = \frac{T}{N} \geq 1$, asymptotically converge to the following spectral probability density

$$\rho(\lambda) = \mathbb{1}_{\lambda \in [\lambda_-, \lambda_+]} \frac{T}{2\pi\lambda} \sqrt{(\lambda_{max} - \lambda)(\lambda - \lambda_{min})} \tag{3.6}$$

Where $\lambda_{min} = (1 - \sqrt{1/Q})^2$ and $\lambda_{max} = (1 + \sqrt{1/Q})^2$.

Every eigenvalue λ_i, $i \in \{1, 2, ..., N\}$, such that $\rho(\lambda_i) > 0$, or alternatively $\lambda_{max} > \lambda_i > \lambda_{min}$, will be consistent with white noise and thus will likely represent uninformative random fluctuations, which we will of course not be interested in.

In order to denoise a correlation matrix [15] we will set a constant value to all the eigenvalues that seem to provide no information, or that have non-zero spectral density. We will update the eigenvalues associated with white noise to their average value in order to reduce their influence while preserving the trace of the correlation matrix.

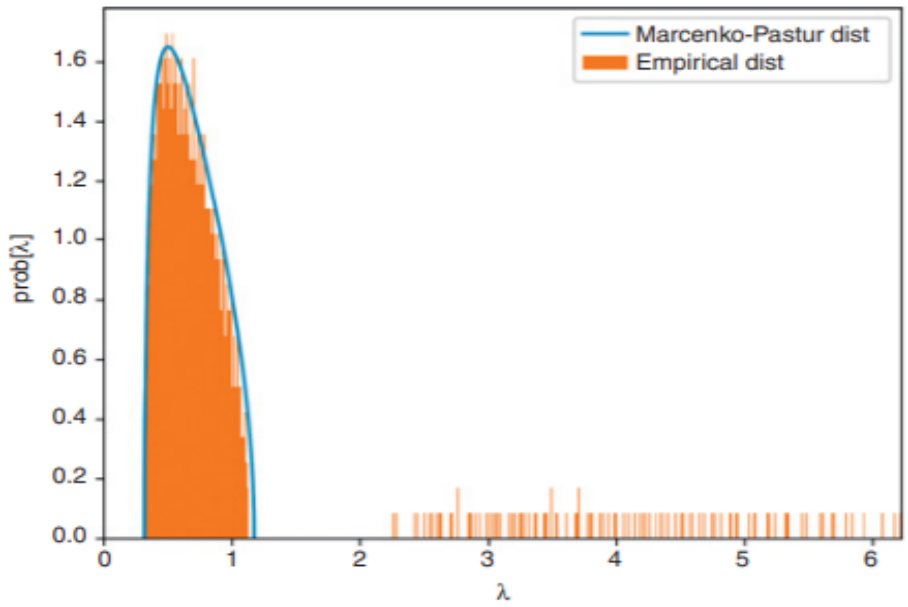


Figure 3.3: Fitting the Marcenko–Pastur PDF on a noisy correlation matrix, from Machine Learning for Asset Managers.

Let $\mathbf{K}$ be a correlation matrix and $\mathbf{W}\mathbf{\Lambda}\mathbf{W}^{-1}$ its spectral decomposition, and let $\widetilde{\mathbf{\Lambda}}$ be the updated diagonal matrix with the denoised eigenvalues

$$\widetilde{\mathbf{K}} = \mathbf{W}\widetilde{\mathbf{\Lambda}}\mathbf{W}^{-1} \tag{3.7}$$

$\widetilde{\mathbf{K}}$ will then be a denoised pseudo-correlation matrix, which we can scale in the following way such that its diagonal consists of ones in order to become a valid correlation matrix

$$\mathbf{\Psi} = \widetilde{\mathbf{K}}\left[(diag[\widetilde{\mathbf{K}}])^{\frac{1}{2}} \otimes (diag[\widetilde{\mathbf{K}}])^{\frac{1}{2}}\right]^{-1} \tag{3.8}$$

Using the clean correlation matrix $\mathbf{\Psi}$ we will establish a dissimilarity matrix just like we did with the raw, unprocessed correlation metric in the Hierarchical Risk Parity algorithm as presented by López de Prado. The elements of this new distance matrix, let's call it $\mathbf{A}$, will be defined as $a_{ij} := \sqrt{\frac{1}{2}(1 - \psi_{ij})}$, where ψ_{ij} are the elements of $\mathbf{\Psi}$.

3.3.4 Mutual Information

In 1945 Claude Shannon published his groundbreaking paper "A mathematical theory of communication" [25], which signified an paradigm shift in humanity's understanding of information, and as a standalone gave birth to a whole new scientific discipline, Information Theory. Prior to Shannon's insights information was seen as a vague, ill-defined, almost metaphysical concept. Shannon showed information is a well-defined, and above all, measurable quantity.

Shannon knew that a measure of information should necessarily have the properties of continuity, symmetry, additivity and its maximal value should be found for equally probable outcomes. He proved there is only one such measure that possesses those 4 properties, which we now know as Entropy. Given a discrete random variable X with $x_1, ..., x_m$ possible outcomes and probability mass function p, entropy, measured in bits, is defined as

$$H(X) = \sum_{i=1}^{m} p(x_i) \log_2 \frac{1}{p(x_i)} = -\mathbb{E}[\log_2 p(x)] \tag{3.9}$$

which is to be interpreted as the bits of information that X provides on average. Another more intuitive way to understand it is as the average surprise resulting from materializations of the random variable. The more uniform the probabilities of the different outcomes the less predictable and the more surprising or informative a random variable is, and the greater Entropy it has. On the other hand, the least homogeneous the probabilities associated to the different outcomes the more predictable they are and the lower the entropy. Deterministic phenomena for instance can be seen as a degenerate case of probabilistic phenomena where all the probability collapses on one outcome, in this case the entropy would be the minimum possible, 0. Deterministic phenomena is completely predictable and does not provide any information inherently.

Entropy is extended to multivariate or conditional random variables very straightforwardly. One should simply consider their associated probability mass function instead of the marginal one. Given two random variables X and Y, $H(X, Y)$, the average surprise when observing the two variables is known as their joint entropy and $H(X|Y)$ as their conditional entropy, which in this instance constitutes the average surprise on X after observing Y.

The mutual information [26] is a non-linear association measure that represents the information a collection of random variables provide of each other and given random vector (X, Y) is defined as

$$I(X, Y) = \mathbb{E}\left[\log_2 \frac{p(x, y)}{p(x)p(y)}\right] = H(X) + H(Y) - H(X, Y) \tag{3.10}$$

so it can be interpreted as the reduction in the surprise when they are considered jointly as opposed to independently, which is the information they carry about each other.

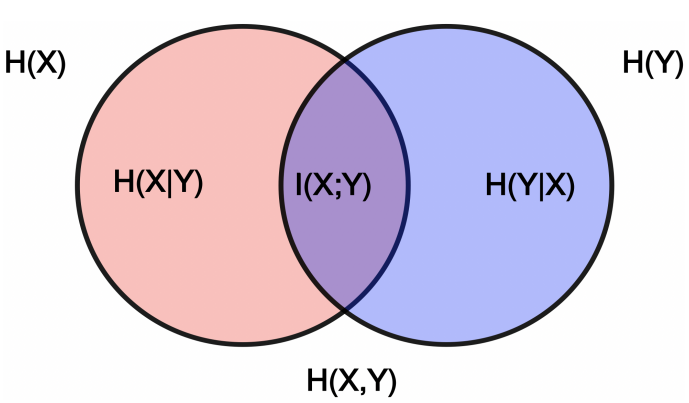


Figure 3.4: Diagram representing the relationship between information-theoretic quantities

Using mutual information as a dissimillarity measure is equivalent to using the Kullback-Leibler divergence between the joint and the marginal distributions.

$$I(X,Y) = D_{KL}\big(p(X,Y)||p(X)p(Y)\big) \tag{3.11}$$

We construct our dissimilarity matrix with the mutual information-based metric distance measure that was suggested by Kraskov et al. in the paper "Hierarchical Clustering Based on Mutual Information" [27].

$$D'(X,Y) := 1 - \frac{I(X,Y)}{\max\{H(X), H(Y)\}} \tag{3.12}$$

The estimation of the mutual information was done through the second k-Nearest-Neighbors based estimator proposed on the seminal paper by Kraskov [28] via the R package 'rmi' [29]. Individual entropies were estimated using the Chao-Shen estimator [30] via the R package 'entropy' [31].

Chapter 4

An Empirical Study with the S&P 500

The purpose of this chapter is to test the performance of Hierarchical Risk Parity (implementation by Gautier Marti [32]) and compare it to that of more traditional methods. Furthermore we aim to test the performance of HRP conditional on a variety of distance measures, each with different merits, that would seem suitable for the job. We would like to expand on the work done by Illya Barziy and Marcin Chlebus [33], about how different co-dependence measures affect Hierarchical Risk Parity's performance. Better performance when working with a specific dissimilarity would be mildly suggestive of it being more apt when it comes to inferring the topological structure in the stock market, as of course it would be reductive to pretend one could get any definitive understanding of the structure of the financial markets in terms of how useful it is for making some sort of predictions. This is, however, an very interesting opportunity given one can not easily evaluate the performance of clustering procedures on the basis of the objectives they were carried out in mind with. Here our clustering has an specific, well-defined purpose, and more importantly its performance can be measured in the sense that it improves the construction of the portfolios.

4.1 Dataset Used

The dataset used in the analysis consists of the daily adjusted closing prices for the stocks of 440 companies in the Standard and Poor's index. More commonly known as S&P 500, it features 500 of the largest public companies in the United States. The stock prices collected are from the first of January of 2014 all the way to 31st of December of 2020. The stocks prices are converted to stock returns, forming a total of 1762 observations. The data was retrieved from Yahoo Finance and the Wikipedia S&P 500 page via the following Python command:

```
import pandas_datareader as pdr
import datetime
import pandas as pd

table=pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
df0 = table[0]
for i in range(0,df0.shape[0]):
    if df0.iloc[i,0]=="BRK.B":
        df0.iloc[i,0]="BRK-B"
    if df0.iloc[i,0]=="BF.B":
        df0.iloc[i,0]="BF-B"

tickers=df0.Symbol

start = datetime.datetime(2014,1,1)
end = datetime.datetime(2020,12,31)
stock_prices = pdr.DataReader(tickers, 'yahoo',start,end)
stock_prices = stock_prices["Adj Close"]
stock_prices = data.dropna(axis=1)
```

Figure 4.1: Command used to pull the data from Wikipedia and the Yahoo Finance servers

We were able to retrieve 479 stocks. The missing stocks did not belong to the S&P for the 7-year period studied, so they were discarded, while being mindful of the risk of introducing some generalization of survivorship bias which we expect will not be of much significance given the large pool of stocks used. Out of the retrieved stocks 440 were randomly selected according to uniform probability as to accommodate for a balanced study, of the characteristics we will detail next, without introducing any new bias.

4.2 The Experiment

The trial we will conduct has as its objective to see what performance we can expect in practice for a given portfolio, while accounting for its method of construction and assuming a naive investor. One that is not savvy in terms of what stocks will work best together in a portfolio and, effectively, chooses stocks to invest at random. The asset allocation methods we will compare will be Hierarchical Risk Parity and its variants employing dissimilarities including Dynamic Time Warping, Mutual Information, the proposed feature-based approach and the Constant Residual Eigenvalue method. We will furthermore see how this Hierarchical methods fair against the infamously hard to beat Uniform Weighting Portfolio. Portfolio optimization procedures are performed on historical, or in-sample, data. According to the assumption that the in-sample qualities of the portfolio should be similar to the out-of-sample portfolios, predictions are made about the behavior of the portfolios out-of-sample. Those predictions will be in this case in terms of the expected returns and risk. In-sample performance is not of practical interest so we will focus on out-of-sample performance.

The trial is structured according to the train-test paradigm. The training set for each given portfolio will be of one year, same as the testing set. For each portfolio the test

set will be constituted by the observations associated to the subsequent year to which the portfolio was trained. For each year 11 portfolios will be trained per method, with non-overlapping, randomly sampled sets of 40 stocks. Each year the 11 sets of stocks used to construct the portfolios will be generated anew such that we control for variability in the quality of the sets of stocks on which the portfolios are trained across time. Maintaining constant the sets of stocks would introduce bias as the luck we get in the construction of the 11 sets of stocks would have too great of an impact on the study. How well specific groups of stocks work together in a portfolio is not a factor of interest right now. The sets of stocks will vary yearly but not across the different methods employ to construct the portfolio as that would add variability that we should like to spare.

We chose to select 11 sets of non-overlapping stocks each year as opposed to a Monte-carlo approach that sampled and indefinite number of randomly sampled overlapping sets of stocks, so that every stock is equally represented on the analysis and we do not add uninteresting variability associated to unequal representation. The bias associated to the limited non-overlapping sets of stocks should not be worrying given those sets vary across the years.

We have historical data for 7 years through which we will be able to calculate in-sample portfolios from 2014 to 2019 to be tested out-of sample. We can not take full advantage of the historical returns from 2020 because the in-sample portfolios from that year could not be tested out-of-sample.

To summarize, we will compute the weights for 11 sets of stocks per method, per year that has an out-of-sample reference. Given we are applying 6 methods and have 6 years for out-of-sample testing we will be computing $11 \times 6 \times 6 = 396$ distinct portfolios.

4.3 Results

The out-of-sample performance of the portfolios across the entirety of the testing period can be summarized via the following table. It features the estimator for each performance metric with its average intra-year standard error, right beside, within brackets. Using the regular standard error would be misleading given it does not account in any way for the development of stock returns across the years and we are not interested in the inter-year effects as a whole. The Conditional Value-at-Risk is calculated for $\alpha = 0.05$.

Method	Mean Returns	Mean Volatility	Mean CVaR
Correlation Metric	14.49% (0.035)	0.0071 (0.00046)	-2.43% (0.0015)
Denoised Correlation Metric	14.34% (0.033)	0.0071 (0.00043)	-2.43% (0.0015)
Dynamic Time Warping	14.63% (0.034)	0.0074 (0.00044)	-2.50% (0.0015)
Feature-based Metric	14.51% (0.032)	0.0072 (0.00043)	-2.46% (0.0015)
Mutual Information	14.52% (0.033)	0.0073 (0.00045)	-2.47% (0.0016)
Uniform Portfolio	16.10% (0.034)	0.0084 (0.00048)	-2.70% (0.0017)

Table 4.1: Mean Performance by Portfolio Construction Method, and Standard Error

We will have to abstain from making any remarks about the statistical significance of these differences given the observations of daily returns per portfolio generated is not well suited to any meaningful statistical analysis because of obvious issues related to statistical independence and complex heterocedasticity. A meta-analysis that tries to find statistical differences between the estimators of the performance measures themselves is neither feasible according to the main approaches on the Generalized Linear Modeling framework. All the differences we will discuss will be in terms of financial significance as is standard in the Portfolio Optimization scientific literature.

When it comes to the returns we can see clear, financially significant, differences. The equal weights portfolio trumps all with a 16.1% mean returns. Far behind lags the rest of the pack. At 14.63% returns the Dynamic Time Warping approach excels among the Hierarchical Risk Parity variants. The following three the methods all perform very similarly in terms of profitability. The Mutual Information approach seems to do better among them at 14.52%, but its difference with the feature-based approach is minimal and still very minor with respect to the base Hierarchical Risk Parity, at 14.49%. Denoising the correlation matrix via the results of Marcenko and Pastur prior to establishing a correlation based dissimilarity did not turn out well as the approach struggled in terms of profitability with respect to the rest of portfolio construction methods at a low 14.34% returns.

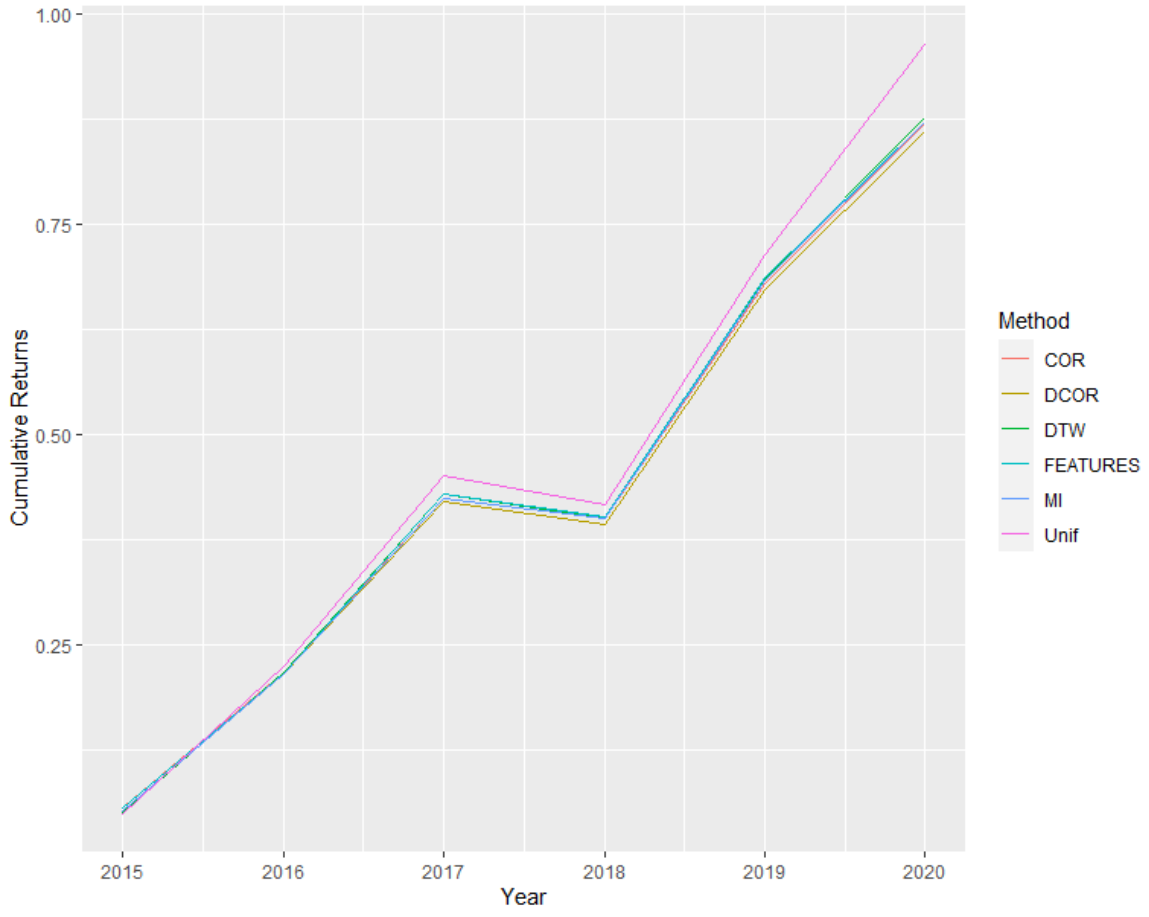
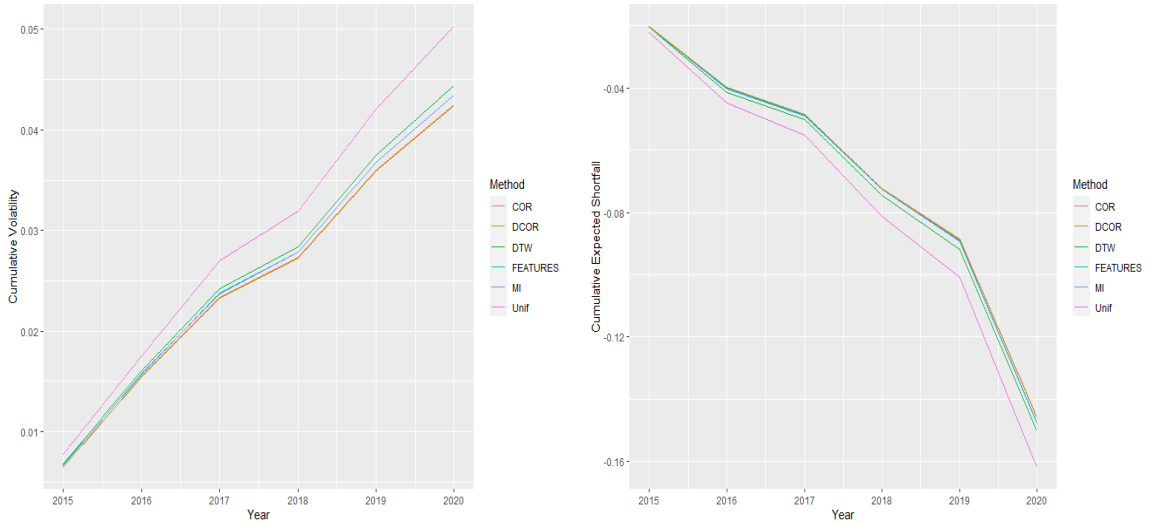


Figure 4.2: Yearly Cumulative Sum of Returns per Method



(a) Cumulative Sum of Volatility per Method (b) Cumulative Sum of CVaR per Method

Figure 4.3: Yearly Cumulative Sum of Risk Measures per Method

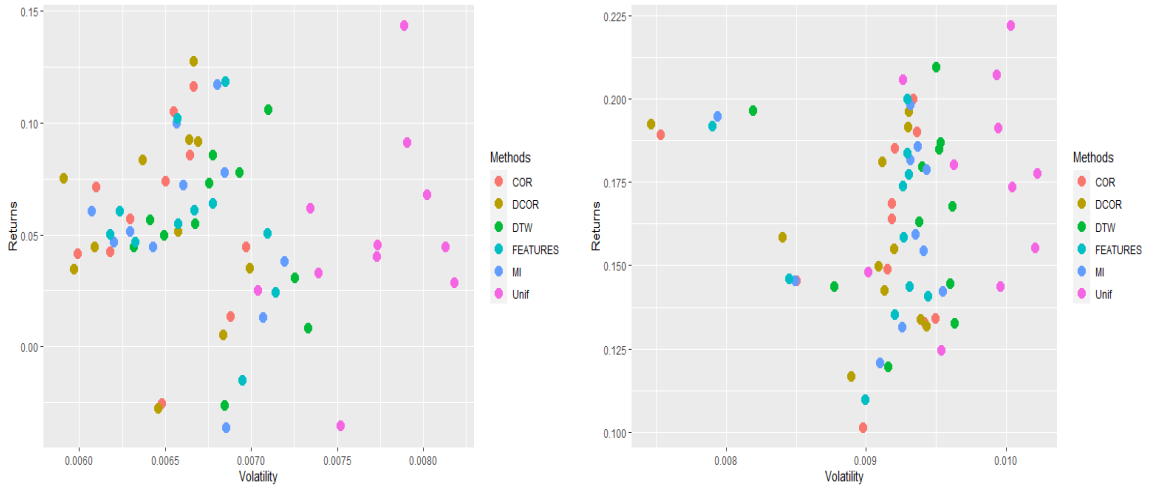
The association between out-of-sample returns and volatility seems to certainly be positive. While the HRP variants performed all very similarly on this front, the uniform weights portfolio had a comparatively high volatility. It being a 15% more than the next highest, that of the shape-based HRP. The rest of the methods sit around an average volatility of 0.0072, they behave very similarly in this respect. The higher a method ranked in terms of returns the higher it ranked in terms of volatility. And the higher it ranked in terms of returns the higher it ranked in terms of the magnitude of its Expected Shortfall, or Conditional Value-at-Risk. At the 5% level of draw-down volatility, along the lines of regular volatility the equal weights portfolio is a clear outlier. The expected left tail movement under the 5th percentile implies a loss 0f 2.7% in the value of the naive portfolio. Next up is the DTW-based approach with a CVaR of -2.5%, shortly followed by the Mutual Information and feature-based methods. The HRP had the lower draw-down volatilities among the correlation based dissimilarities, at a 2.43% expected 5th percentile loss.

We have represented graphically all of the 396 portfolios according to their profitability and their volatility through 6 scatterplots. Each scatter plot contains the representation of the portfolios tested out-of-sample for a given year in terms of their returns and volatility. We can see as expected the clearest separation among points is found on whether the portfolios are Equal Weights or they are HRP-adjacent. The behavior among hierarchical methods seems to be relatively similar between themselves, but very different to that of the naive portfolio. The most visible trend sustained over time is that the Equal Weights portfolio almost always will imply greater volatility and in a way that can be pretty significant. The second thing that comes to mind is that there are many small groupings of 5 associated colors that correspond all to the HRP-adjacent portfolios calculated on a same set of stocks. On those various little clusters we can see there is not a sustained pattern across time, the volatilities seem to behave along the lines of what could be anticipated by the table 4.1. However that table does not tell the whole story when it comes to how the returns behave, as

the ranking of the methods with respect to profitability varies notoriously across the different testing periods.

Method	2015	2016	2017	2018	2019	2020
Correlation Metric	5.69%	16.01%	20.69%	-2.34%	27.86%	19.01%
Denoised Correlation Metric	5.59%	15.91%	20.56%	-2.81%	27.95%	18.85%
Dynamic Time Warping	5.11%	16.63%	21.23%	-3.00%	28.52%	19.26%
Feature-based Metric	5.62%	16.02%	21.30%	-2.72%	28.35%	18.48%
Mutual Information	5.32%	16.30%	20.76%	-2.34%	28.23%	18.81%
Uniform Portfolio	4.97%	17.55%	22.56%	-3.43%	29.73%	24.96%
Overall Mean Returns	5.38%	16.40%	21.18%	-2.77%	28.44%	19.89%

Table 4.2: Returns by Portfolio Construction Method, and Year



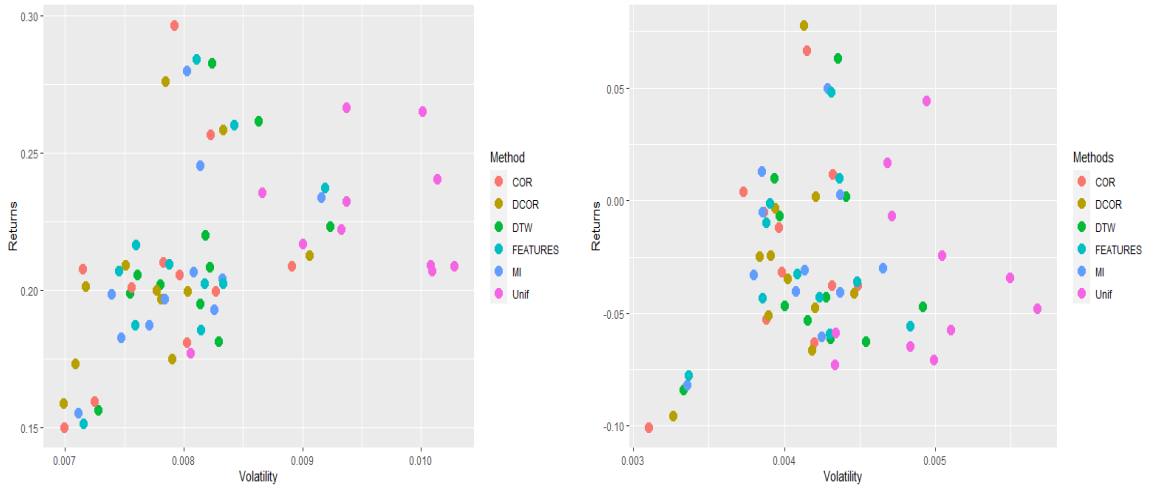
(a) Out-of-sample Performance 2015

(b) Out-of-sample Performance 2016

Figure 4.4: Years 1 and 2 of testing

On the first year of testing for instance, despite the unfortunate general performance of the HRP portfolios constructed via correlation-based metrics they seem to have done consistently better than the Mutual Information and DTW approaches. The standard HRP algorithm shined above all with a very low volatility and the highest returns among all. The uniform weights portfolio did very poorly by comparison, it had the lowest returns and the highest volatilities.

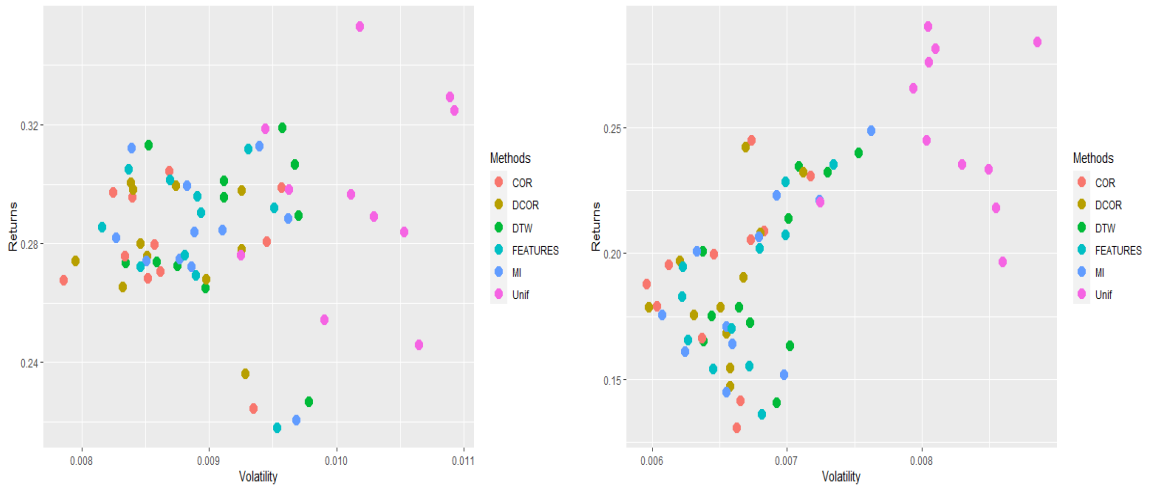
On the second year and the third year of testing we can appreciate the more general patterns in table 4.1. On 2018, which was a very rough year for the S&P 500, all the vast majority of the portfolios lost money. The standard correlation-based metric shines once again, with the highest returns and the lowest volatility. The Mutual Information dissimilarity excels as well, slightly improving on the crisis management of the base HRP, it achieved even lower losses at a very similar volatility. Once again the naive portfolios performed very poorly.



(a) Out-of-sample Performance 2017 (b) Out-of-sample Performance 2018

Figure 4.5: Years 3 and 4 of testing

The performance in the year 2019 shares the spirit of table 4.1. The correlation-based HRPs have lower payback as well as volatility, and the DTW approach and the Uniform portfolio do best in terms of returns and worse when it comes to volatility. The equal weights allocation is very profitable, a 1.5% more than the next best, which we implied is the Dynamic Time Warping based Hierarchical Risk Parity. The other 2 methods are just a compromise between the more risky and profitable ones, and the more predictable less profitable ones. For the last year, the uniform weights portfolio simply blew it out of the park. Its average returns were roughly of a 25%. Next highest method in terms of profitability was the DTW Hierarchical Risk Parity, and sat lowly by comparison at 19.26% returns. It is worth noting the base HRP was third in terms of returns, at a very low volatility, and the feature-based approach had the lowest returns.



(a) Out-of-sample Performance 2019 (b) Out-of-sample Performance 2020

Figure 4.6: Years 5 and 6 of testing

The volatilities of the per year per method are very uninteresting beyond providing confirmation of the general tendencies already apparent in table 4.1 being consistent across time. It is the case too when it comes to draw-down volatility as it behaves very similarly. However, in the case of the returns we can see that their relationship across the years is more interesting.

Kendall Correlation	Mean Returns per year
COR Centered Returns per year	-0.733
DCOR Returns per year	-0.333
DTW Returns per year	0.200
FEATURES Centered Returns per year	-0.200
MI Centered Returns per year	-0.600
Unif Centered Returns per year	0.600

Table 4.3: Kendall Correlations

We calculated the Kendall correlations between the Mean Returns per year across all methods, and the Centered Mean Returns for each specific method. Centered as to be able to compare the methods at a baseline, as to whether there is variation with respect to the rest of methods. Here we see what we intuited. There is reasonable empirical evidence suggesting that for more hostile years in the market the standard Hierarchical Risk Parity with the default correlation metric, will tend to work better. Not to such an extent, but he same seems to be true for Mutual Information approach. Not without the implication, however, that these methods will have a hard time when the returns to be had are generally high. On the other hand, the celebrated Uniform weighting portfolios will struggle during hard times, but excel during those that are more favorable to the investor.

Chapter 5

Conclusions

This work lies on the intersection of Time Series Analysis, Clustering and Portfolio Optimization. Our intent was to integrate some popular techniques in Time Series Analysis, Information Theory and Random Matrix Theory within the Hierarchical Risk Parity algorithm in order to improve its performance, explore the viability of the aforementioned techniques in the field of Portfolio Optimization and investigate the value in the hierarchical conception of the financial markets when it comes to asset allocation.

We tested the performance of 396 distinct portfolios in total, constructed via 6 different methodologies across a 6 year period. The methods employed include the standard Hierarchical Risk Parity algorithm, based on a correlation-based metric and four different variants employing different dissimilarity measures according to which build the hierarchical tree. One variant employed a metric based on denoised correlations. Another one used a shape-based dissimilarity know as Dynamic Time Warping. The third variant considered employed a distance measure based on Mutual Information, a general, non-linear, association measure intimately connected to the Pearson correlation. The fourth and last variant was inspired by feature-based clustering and resulted on a dissimilarity matrix that was built on the euclidean distance of the scores of the principal components given a matrix that summarized particularly relevant characteristics of the series of stocks' returns. Lastly we computed the Uniform Weights portfolios to serve as a reference for a respected, well-performing method of asset allocation, in order to compare it to the HRP-adjacent methods studied.

We found that the behavior of the HRP-adjacent methods was relatively similar. Their volatilities and down-ward volatilities, measured by the standard deviation and the CVaR with $\alpha = 0.05$, sustained the same pattern across all the years of testing. The correlation-based metrics consistently built lower risk portfolios while the Dynamic Time Warping HRP portfolios tended to have higher risks among the hierarchical methods. The Mutual Information and feature-based approaches were middle of the road when it comes to risk. On the other hand the Uniform Weights portfolios were characterized by risks that are significantly higher than that of any of the Hierarchical Risk Parity variants.

More interesting findings are associated to the behavior of the returns across time.

The general trend is that the Equal Weights Portfolio is by far the most profitable. The DTW-based portfolios would be the next best by a good margin, followed by the portfolios constructed in terms of the features and mutual information and those built through the standard HRP. The portfolios making use of the Constant Residual Eigenvalue method did very poorly overall in terms of returns when compared to the rest. However these general returns related trends do not tell the whole story. We found that in testing periods where the S&P 500 struggled, comparatively, and had average returns lower than a 7% more or less the default HRP performed the best in terms of returns with very low risks too. The Mutual Information Portfolios excelled too during harder times but not to the same extent. The Equal Weights portfolios despite their overall success had a really rough time when the markets did not run so smoothly. For years where the expected profits are on the lower side the performance of these naive portfolios was very bad, on the other hand during easier times its performance was really good, only contested by Dynamic Time Warping portfolios in the eyes of the conservative investor.

We have found that all the HRP-adjacent methods but the one employing a metric constructed over a denoised correlation matrix would be justifiable in application. The base HRP is an awesome choice for the skeptical investor that has the opinion that the markets will struggle during the coming year. The feature-based metric and the mutual information based distance can be used by conservative investors that do not want risk but want an reasonable level of profitability. The choice on the two would be based on how skeptical a portfolio manager is about the coming year, if the returns are expected to be on the higher end the recommendation would be to employ the feature-based approach while alternatively the information theoretic one would be encouraged. The obvious choice for the optimistic investor that expects the markets to have a good year and wants higher returns but still low risk is the Dynamic Time Warping way while the method of asset allocation of choice for the optimistic investor that has a high tolerance to risk and chases after very high returns is the Equal Weights portfolio.

It is clear that the hierarchical conception of the financial markets, via the standard correlation metric, helps us make portfolios that work very well when other heuristic approaches fail. We also found evidence that when stock returns fly high Dynamic Time Warping and perhaps, more generally, other shape based dissimilarities are helpful to build a hierarchy that models the relationships among stocks in a way that helps us build better risk-adjusted, more robust, competitive portfolios.

Chapter 6

Bibliography

[1] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Apllications*. Springer. Davis CA and Pittsburgh PA, 2010.

[2] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning*. Springer. Stanford, California, 2008.

[3] Wikipedia, Clustering.

`https://en.wikipedia.org/wiki/Clustering`

[4] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow*. O'Reilly. 2019.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning.*. Springer. 2010.

[6] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi and Teh Ying Wah. *Time-series clustering – A decade review*. Elsevier, Information Systems. 2015.

[7] Harry Markowitz. *Portfolio Selection*. Journal of Finance. 1952.

[8] Marcos López de Prado *Advances in Financial Machine Learning*. Wiley. 2016.

[9] Fisher Black and Robert Litterman. *Asset Allocation Combining Investor Views with Market Equilibrium*. Journal of Fixed Income. 1992.

[10] V. A. Marcenko and L. A. Pastur. *Distribution of eigenvalues for some sets of random matrices*. Mathematic USSR-Sbornik. 1967.

[11] Vincenzo Tola, Fabrizio Lillo, Mauro Gallegati and Rosario N. Mantegna. *Cluster analysis for portfolio optimization*. Catania, Italy. Santa Fe, USA. 2005.

[12] Olivier Ledoit. *Honey, I Shrunk the Sample Covariance Matrix.*. London, 2003.

[13] Marcos López de Prado. *Building Diversified Portfolios that Outperform out-of-sample*. 2016.

[14] Thomas Raffinot. *Hierarchical Clustering-Based Asset Allocation*. Journal of Portfolio Management. 2018.

[15] Marcos López de Prado *Machine Learning for Asset Managers*. Cambridge University Press. 2020.

[16] Roger Clarke, Harindra de Silva, and Steven Thorley. *Minimum-Variance Portfolios in the U.S. Equity Market*. 2006.

[17] Sharpe, W. F. *Mutual Fund Performance*. Journal of Business. 1966.

[18] P. Artzner, F. Delbaen, J. M. Eber and D. Heath. *Coherent Measures of Risk*. Mathematical Finance. 1999.

[19] Rosario N. Mantenga. *Hierarchical Structure in Financial Markets*. Palermo, Italy, 1998.

[20] Yangzhuoran Yang and Rob J. Hyndman. *Introduction to the tsfeatures package*.

`https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html`

2020.

[21] Rob J. Hyndman et al. *tsfeatures: Time Series Feature Extraction*.

`https://cran.r-project.org/web/packages/tsfeatures/index.html`

2020.

[22] Donald J. Bemdt and James Clifford. *Using Dynamic Time Warping to Find Patterns in Time Series*. New York, USA. 1994.

[23] Wikipedia, Dynamic Time Warping.

`https://en.wikipedia.org/wiki/Dynamic_time_warping`

[24] Pablo Montero Manso and Jose Vilar Fernández. *TSclust: Time Series Clustering Utilities*.

`https://cran.r-project.org/web/packages/TSclust/index.html`

2020.

[25] Claude Shannon. *A Mathematical Theory of Communication*. The Bell System Technical Journal. 1948.

[26] James V. Stone. *Information Theory: A Tutorial Introduction*. Sebtel Press. 2015.

[27] Alexander Kraskov, Harald Stogbauer, Ralph G. Andrzejak, and Peter Grassberger. *Hierarchical Clustering Based on Mutual Information*. 2008.

[28] Alexander Kraskov, Harald Stogbauer, and Peter Grassberger. *Estimating Mutual Information*. 2004.

[29] Isaac Michaud. *rmi: Mutual Information Estimators*.

`https://cran.r-project.org/web/packages/rmi/index.html`

2018.

[30] A. Chao and T.J. Shen. *Nonparametric estimation of Shannon's index of diversity when there are unseen species in sample*. Environ. Ecol. Stat. 2003.

[31] Jean Hausser and Korbinian Strimmer *entropy: Estimation of Entropy, Mutual Information and Related Quantities.*

`https://cran.r-project.org/web/packages/entropy/index.html`

2021.

[32] Gautier Marti . *Hierarchical Risk Parity.*

`https://gmarti.gitlab.io/qfin/2018/10/02/hierarchical-risk-parity-part-1.html`

2018.

[33] Illya Barziy and Marcin Chlebus. *HRP performance comparison in portfolio optimization under various codependence and distance metrics.* Warsaw, Poland. 2020.

Chapter 7

Additional Tables + Code

7.1 Extra Tables

Method	2015	2016	2017	2018	2019	2020
Correlation Metric	0.0065	0.0090	0.0078	0.0040	0.0087	0.0065
Denoised Correlation Metric	0.0065	0.0090	0.0078	0.0040	0.0087	0.0065
Dynamic Time Warping	0.0068	0.0093	0.0081	0.0042	0.0091	0.0069
Feature-based Metric	0.0067	0.0091	0.0080	0.0041	0.0089	0.0067
Mutual Information	0.0066	0.0091	0.0080	0.0041	0.0089	0.0067
Uniform Portfolio	0.0077	0.0098	0.0095	0.0049	0.0102	0.0082

Table 7.1: Volatility by Portfolio Construction Method, and Year

Method	2015	2016	2017	2018	2019	2020
Correlation Metric	-0.0202	-0.0199	-0.0088	-0.0236	-0.0162	-0.0572
Denoised Correlation Metric	-0.0203	-0.0196	-0.0087	-0.0237	-0.0161	-0.0573
Dynamic Time Warping	-0.0206	-0.0208	-0.0088	-0.0243	-0.0173	-0.0584
Feature-based Metric	-0.0205	-0.0195	-0.0087	-0.0239	-0.0166	-0.0581
Mutual Information	-0.0205	-0.0198	-0.0086	-0.0238	-0.0168	-0.0585
Uniform Portfolio	-0.0221	-0.0228	-0.0102	-0.0260	-0.0198	-0.0608

Table 7.2: CVaR at 0.05 by Portfolio Construction Method, and Year

7.2 Python: Data Retrieval + Preprocessing

```
import pandas_datareader as pdr
import datetime
import pandas as pd
import numpy as np
```

```
table=pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
df0 = table[0]
for i in range(0,df0.shape[0]):
    if df0.iloc[i,0]=="BRK.B":
        df0.iloc[i,0]="BRK-B"
    if df0.iloc[i,0]=="BF.B":
        df0.iloc[i,0]="BF-B"

tickers=df0.Symbol
start = datetime.datetime(2014,1,1)
end = datetime.datetime(2020,12,31)

print("The amount of stocks chosen to observe: " + str(len(tickers)))

stock_prices = pdr.DataReader(tickers, 'yahoo',start,end)
stock_prices = stock_prices["Adj Close"]
stock_prices = data.dropna(axis=1)

stock_returns=stock_prices.copy()
for j in range(0,(stock_returns.shape[1])):
    for i in range(stock_returns,(r.shape[0])):
        stock_returns.iloc[i,j]=stock_prices.iloc[i,j]/stock_prices.iloc[i-1,j]
#percentual returns

stock_returns=stock_returns[1:]
stock_returns=stock_returns-1
stock_returns.to_csv('bigrS&P500.csv')
```

7.3 R: Computing Dissimilarity Matrices

```
snp <-  read.csv("~/Datasets/bigrS&P500.csv")
snp$Date=as.Date(snp$Date)

K=7 #nbatches
batch=rep(1,length(snp$Date))
batch[snp$Date>=as.Date("2014-01-1") & snp$Date<as.Date("2015-01-01")]=1
batch[snp$Date>=as.Date("2015-01-1") & snp$Date<as.Date("2016-01-1")]=2
batch[snp$Date>=as.Date("2016-01-1") & snp$Date<as.Date("2017-01-01")]=3
batch[snp$Date>=as.Date("2017-01-01") & snp$Date<as.Date("2018-01-1")]=4
batch[snp$Date>=as.Date("2018-01-01") & snp$Date<as.Date("2019-01-1")]=5
batch[snp$Date>=as.Date("2019-01-01") & snp$Date<as.Date("2020-01-1")]=6
batch[snp$Date>=as.Date("2020-01-01")]=K
snp=cbind(batch,snp[,-1])
table(snp$batch)
```

```
N=440
stocks.per.g=40
n.groups=11
sampled=sort(sample(2:479,N))
snp=snp[,c(1,sampled)]
I=matrix(sample(1:N),ncol=stocks.per.g)
for(i in 1:5){
I=rbind(I,matrix(sample(1:N),ncol=stocks.per.g))
}
I=cbind(rep(1:n.groups,6),I)
I=cbind(rep(1:6,each=n.groups),I)
write.csv(I,"Id3.csv", row.names = FALSE)
#no seed, sorry. Id3 File on my github: https://github.com/FranDeLio/RCode
write.csv(snp,"snpgreat.csv", row.names = FALSE)
dim(I)

#cov
#do a for that does this all methods too
K=66
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
for(j in 1:K){#unique(snp$batch)){
distance=cov(snp[snp$batch==I[j,1], unlist(I[j,-c(1,2)])+1])
colnames(distance)=paste(1:stocks.per.g)
stacked=rbind(stacked,distance)
}
stacked=cbind(from.subbatch,stacked)
stacked=cbind(from.batch,stacked)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#stacked=cbind(from.batch,stacked)
write.csv(stacked,"COVright3.csv", row.names = FALSE)
dim(stacked)

#do a for that does this 4 all methods
K=66
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
for(j in 1:K){#unique(snp$batch)){
distance=cor(snp[snp$batch==I[j,1], unlist(I[j,-c(1,2)])+1])
distance=sqrt((1-distance)/2)
colnames(distance)=paste(1:stocks.per.g)
```

```
stacked=rbind(stacked,distance)
}
stacked=cbind(from.subbatch,stacked)
stacked=cbind(from.batch,stacked)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#stacked=cbind(from.batch,stacked)
write.csv(stacked,"CORMETright3.csv", row.names = FALSE)

Marcenko.Pastur=function(R){
Q=ncol(R)/nrow(R)
Kr=cor(R)
max.lambda=(1+sqrt(Q))^2
spectral=eigen(Kr)
eigenvalues=spectral$values
constant.eigenvalues=mean(eigenvalues[max.lambda>=eigenvalues])
spectral$values[max.lambda>=spectral$values]=constant.eigenvalues
Kr=spectral$vectors%*%diag(spectral$values)%*%t(spectral$vectors)
Kr=Kr%*%solve(sqrt(diag(diag(Kr)))%*%t(sqrt(diag(diag(Kr)))),tol=1e-16)
Kr=(Kr+t(Kr))/2
return(list(denoised_cov=Kr,sum_eigs=sum(max.lambda<eigenvalues)))
}


K=66
eig=c()
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
for(j in 1:K){#unique(snp$batch)){
marcenko=Marcenko.Pastur(snp[snp$batch==I[j,1], unlist(I[j,-c(1,2)])+1])
distance=marcenko$denoised_cov
#distance=sqrt((1-distance)/2), done in python
colnames(distance)=paste(1:stocks.per.g)
eig=c(eig,marcenko$sum_eigs)
stacked=rbind(stacked,distance)
}
stacked=cbind(from.subbatch,stacked)
stacked=cbind(from.batch,stacked)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#stacked=cbind(from.batch,stacked)
write.csv(stacked,"MARCOVright3.csv", row.names = FALSE)
dim(stacked)
```

```
#pca features
library(cluster)
library(tsfeatures)
K=66
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
characteristika=data.frame()
for(i in 1:K){ #unique(snp$batch)){
desired.batch=snp[snp$batch==I[j,1], unlist(I[j,-c(1,2)])+1]
pca=prcomp(as.matrix(tsfeatures(as.list(ts(desired.batch)),
features = c("stl_features","acf_features","entropy","stability",
"lumpiness"))[,-c(1:2,11:14)]),center=TRUE,scale=TRUE)
characteristika=rbind(characteristika, as.matrix(daisy(pca$x,metric="euclidean")))
}
characteristika=cbind(from.subbatch,characteristika)
characteristika=cbind(from.batch,characteristika)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
dim(characteristika)
write.csv(characteristika,"FEATURESright3.csv", row.names = FALSE)


#dtwarp
library(TSclust)
K=66
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
for(j in 1:K){#unique(snp$batch)){
    distance=diss(snp[snp$batch==I[j,1], unlist(I[j,-c(1,2)])+1],METHOD="DTWARP")
distance=as.data.frame(as.matrix(distance))
colnames(distance)=paste(1:stocks.per.g)
stacked=rbind(stacked,distance)
}
stacked=cbind(from.subbatch,stacked)
stacked=cbind(from.batch,stacked)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#stacked=cbind(from.batch,stacked)
write.csv(stacked,"DTWright3.csv", row.names = FALSE)
dim(stacked)
```

```
#making MI dissimilarity
library(copent)
library(rmi)
library(entropy)
copentNorm=function(M){
M=as.matrix(M)
N=ncol(M)
distance=data.frame(matrix(0,N,N))
for(i in 1:N){
for(j in 1:i){
if(i!=j){
distance[i,j]=knn_mi(M[,c(i,j)],splits=c(1,1),
options = list(method = "KSG2", k = 5))#alternative is copent(M[,c(i,j)])
} else { distance[i,j]=entropy::entropy(M[,i],method="CS")}
}
}
for(i in 1:N){
for(j in 1:i){
distance[i,j]=1-distance[i,j]/(max(distance[i,i],distance[j,j]))
}
}
distance=distance+t(distance)
return(distance)
}


K=66
start.time <- Sys.time()
stacked=data.frame()
from.batch=rep(1:6,each=n.groups*stocks.per.g)
from.subbatch=rep(1:n.groups,6,each=stocks.per.g)
for(j in 1:K){#unique(snp$batch)){
distance=copentNorm(snp[snp$batch==I[K,1], unlist(I[K,-c(1,2)])+1])
stacked=rbind(stacked,distance)
end.time <- Sys.time()
  time.taken <- end.time - start.time
  print(time.taken)
}
stacked=cbind(from.subbatch,stacked)
stacked=cbind(from.batch,stacked)

#stacked=cbind(from.batch,stacked)
write.csv(stacked,"MIright3.csv", row.names = FALSE)
dim(stacked)
```

7.4 Python: Computing Portfolios + Performance

```
import pandas as pd
import numpy as np
from scipy.linalg import block_diag
from scipy.cluster.hierarchy import linkage
from scipy.spatial.distance import squareform
import matplotlib.pyplot as plt


def seriation(Z, N, cur_index):
    """Returns the order implied by a hierarchical tree (dendrogram).

       :param Z: A hierarchical tree (dendrogram).
       :param N: The number of points given to the clustering process.
       :param cur_index: The position in the tree for the recursive traversal.

       :return: The order implied by the hierarchical tree Z.
    """
    if cur_index < N:
        return [cur_index]
    else:
        left = int(Z[cur_index - N, 0])
        right = int(Z[cur_index - N, 1])
        return (seriation(Z, N, left) + seriation(Z, N, right))


def compute_serial_matrix(dist_mat, method="single"):
    """Returns a sorted distance matrix.

       :param dist_mat: A distance matrix.
       :param method: A string in ["ward", "single", "average", "complete"].

        output:
            - seriated_dist is the input dist_mat,
              but with re-ordered rows and columns
              according to the seriation, i.e. the
              order implied by the hierarchical tree
            - res_order is the order implied by
              the hierarhical tree
            - res_linkage is the hierarhical tree (dendrogram)

        compute_serial_matrix transforms a distance matrix into
        a sorted distance matrix according to the order implied
        by the hierarchical tree (dendrogram)
    """
    N = len(dist_mat)
```

```
    flat_dist_mat = squareform(dist_mat)
    res_linkage = linkage(flat_dist_mat, method=method)
    res_order = seriation(res_linkage, N, N + N - 2)
    seriated_dist = np.zeros((N, N))
    a,b = np.triu_indices(N, k=1)
    seriated_dist[a,b] = dist_mat[[res_order[i] for i in a],
[res_order[j] for j in b]]
    seriated_dist[b,a] = seriated_dist[a,b]

    return seriated_dist, res_order, res_linkage



def compute_HRP_weights(covariances, res_order):
    weights = pd.Series(1, index=res_order)
    clustered_alphas = [res_order]

    while len(clustered_alphas) > 0:
        clustered_alphas = [cluster[start:end] for cluster in clustered_alphas
                            for start, end in ((0, len(cluster) // 2),
                                               (len(cluster) // 2, len(cluster)))
                            if len(cluster) > 1]
        for subcluster in range(0, len(clustered_alphas), 2):
            left_cluster = clustered_alphas[subcluster]
            right_cluster = clustered_alphas[subcluster + 1]

            left_subcovar = covariances[left_cluster].loc[left_cluster]
            inv_diag = 1 / np.diag(left_subcovar.values)
            parity_w = inv_diag * (1 / np.sum(inv_diag))
            left_cluster_var = np.dot(parity_w, np.dot(left_subcovar, parity_w))

            right_subcovar = covariances[right_cluster].loc[right_cluster]
            inv_diag = 1 / np.diag(right_subcovar.values)
            parity_w = inv_diag * (1 / np.sum(inv_diag))
            right_cluster_var = np.dot(parity_w, np.dot(right_subcovar, parity_w))

            alloc_factor = 1 - left_cluster_var /
(left_cluster_var + right_cluster_var)

            weights[left_cluster] *= alloc_factor
            weights[right_cluster] *= 1 - alloc_factor

    return weights

def compute_MV_weights(covariances):
    inv_covar = np.linalg.inv(covariances)
    u = np.ones(len(covariances))
```

```
    return np.dot(inv_covar, u) / np.dot(u, np.dot(inv_covar, u))


def compute_RP_weights(covariances):
    weights = (1 / np.diag(covariances))

    return weights / sum(weights)


def compute_unif_weights(covariances):

    return [1 / len(covariances) for i in range(len(covariances))]

def Expected_Shortfall(weights,returns,alpha):
    x=np.dot(returns,HRP_weights)
    return np.mean(x[x<np.quantile(x,alpha)])


##Execution

covright=pd.read_csv("COVright3.csv")
cormet=pd.read_csv("CORMETright3.csv")
marcovright=pd.read_csv("MARCOVright3.csv")
marcovright.iloc[:,2:42]=np.sqrt((1-marcovright.iloc[:,2:42])/2)
dtwright=pd.read_csv("DTWright3.csv")
miright=pd.read_csv("MIright3.csv")
dfright=pd.read_csv("snpgreat.csv")
features=pd.read_csv("FEATURESright3.csv")
Id=pd.read_csv("Id3.csv")
Id2=Id.drop(['V1','V2'],axis=1).copy()


#cormet
dr2=pd.DataFrame(np.zeros([6,11]))
dv2=pd.DataFrame(np.zeros([6,11]))
des2=pd.DataFrame(np.zeros([6,11]))
diss=cormet
k=0
for i in range(1,7):
    for j in range(1,12):
        distances=diss.loc[(diss["from.batch"]==i) &
(diss["from.subbatch"]==j),:].copy()
        distances=distances.drop(["from.batch","from.subbatch"],axis=1)
        distances.columns=list(range(0,40))
        distances.index=list(range(0,40))
        covi=covright.loc[(covright["from.batch"]==i) &
```

```
 (covright["from.subbatch"]==j),:].copy()
        covi=covi.drop(["from.batch","from.subbatch"],axis=1)
        covi.columns=list(range(0,40))
        covi.index=list(range(0,40))
        ordered_dist_mat, res_order, res_linkage =
compute_serial_matrix(distances.values, method='single')
        HRP_weights=compute_HRP_weights(covi, res_order)
        HRP_weights=HRP_weights[range(0,40)]
        #ds.loc[i-1,:]=HRP_weights
        a=dfright.iloc[:, Id2.iloc[k,:].values].copy()
        a=a[dfright['batch']==i+1]
        dr2.loc[i-1,j-1]=np.dot(HRP_weights,a.apply(sum,axis=0))
        dv2.loc[i-1,j-1]=np.sqrt(np.dot(HRP_weights,covi).dot(HRP_weights))
        des2.loc[i-1,j-1]=Expected_Shortfall(HRP_weights,a,0.05)
        k+=1
dr2.apply(np.mean,axis=1)


#marcenko
dr3=pd.DataFrame(np.zeros([6,11]))
dv3=pd.DataFrame(np.zeros([6,11]))
des3=pd.DataFrame(np.zeros([6,11]))
diss=marcovright
k=0
for i in range(1,7):
    for j in range(1,12):
        distances=diss.loc[(diss["from.batch"]==i) &
(diss["from.subbatch"]==j),:].copy()
        distances=distances.drop(["from.batch","from.subbatch"],axis=1)
        distances.columns=list(range(0,40))
        distances.index=list(range(0,40))
        covi=covright.loc[(covright["from.batch"]==i) &
 (covright["from.subbatch"]==j),:].copy()
        covi=covi.drop(["from.batch","from.subbatch"],axis=1)
        covi.columns=list(range(0,40))
        covi.index=list(range(0,40))
        ordered_dist_mat, res_order, res_linkage =
compute_serial_matrix(distances.values, method='single')
        HRP_weights=compute_HRP_weights(covi, res_order)
        HRP_weights=HRP_weights[range(0,40)]
        #ds.loc[i-1,:]=HRP_weights
        a=dfright.iloc[:, Id2.iloc[k,:].values].copy()
        a=a[dfright['batch']==i+1]
        dr3.loc[i-1,j-1]=np.dot(HRP_weights,a.apply(sum,axis=0))
        dv3.loc[i-1,j-1]=np.sqrt(np.dot(HRP_weights,covi).dot(HRP_weights))
        des3.loc[i-1,j-1]=Expected_Shortfall(HRP_weights,a,0.05)
        k+=1
```

```
dr3.apply(np.mean,axis=1)


#mi
dr4=pd.DataFrame(np.zeros([6,11]))
dv4=pd.DataFrame(np.zeros([6,11]))
des4=pd.DataFrame(np.zeros([6,11]))
diss=miright
k=0
for i in range(1,7):
    for j in range(1,12):
        distances=diss.loc[(diss["from.batch"]==i) &
 (diss["from.subbatch"]==j),:].copy()
        distances=distances.drop(["from.batch","from.subbatch"],axis=1)
        distances.columns=list(range(0,40))
        distances.index=list(range(0,40))
        covi=covright.loc[(covright["from.batch"]==i) &
 (covright["from.subbatch"]==j),:].copy()
        covi=covi.drop(["from.batch","from.subbatch"],axis=1)
        covi.columns=list(range(0,40))
        covi.index=list(range(0,40))
        ordered_dist_mat, res_order, res_linkage =
 compute_serial_matrix(distances.values, method='single')
        HRP_weights=compute_HRP_weights(covi, res_order)
        HRP_weights=HRP_weights[range(0,40)]
        #ds.loc[i-1,:]=HRP_weights
        a=dfright.iloc[:, Id2.iloc[k,:].values].copy()
        a=a[dfright['batch']==i+1]
        dr4.loc[i-1,j-1]=np.dot(HRP_weights,a.apply(sum,axis=0))
        dv4.loc[i-1,j-1]=np.sqrt(np.dot(HRP_weights,covi).dot(HRP_weights))
        des4.loc[i-1,j-1]=Expected_Shortfall(HRP_weights,a,0.05)
        k+=1
dr4.apply(np.mean,axis=1)

#features
dr5=pd.DataFrame(np.zeros([6,11]))
dv5=pd.DataFrame(np.zeros([6,11]))
des5=pd.DataFrame(np.zeros([6,11]))
diss=features
k=0
for i in range(1,7):
    for j in range(1,12):
        distances=diss.loc[(diss["from.batch"]==i) &
 (diss["from.subbatch"]==j),:].copy()
        distances=distances.drop(["from.batch","from.subbatch"],axis=1)
        distances.columns=list(range(0,40))
        distances.index=list(range(0,40))
```

```
        covi=covright.loc[(covright["from.batch"]==i) &
 (covright["from.subbatch"]==j),:].copy()
        covi=covi.drop(["from.batch","from.subbatch"],axis=1)
        covi.columns=list(range(0,40))
        covi.index=list(range(0,40))
        ordered_dist_mat, res_order, res_linkage =
 compute_serial_matrix(distances.values, method='single')
        HRP_weights=compute_HRP_weights(covi, res_order)
        HRP_weights=HRP_weights[range(0,40)]
        #ds.loc[i-1,:]=HRP_weights
        a=dfright.iloc[:, Id2.iloc[k,:].values].copy()
        a=a[dfright['batch']==i+1]
        dr5.loc[i-1,j-1]=np.dot(HRP_weights,a.apply(sum,axis=0))
        dv5.loc[i-1,j-1]=np.sqrt(np.dot(HRP_weights,covi).dot(HRP_weights))
        des5.loc[i-1,j-1]=Expected_Shortfall(HRP_weights,a,0.05)
        k+=1
dr5.apply(np.mean,axis=1)

#unif
dr6=pd.DataFrame(np.zeros([6,11]))
dv6=pd.DataFrame(np.zeros([6,11]))
des6=pd.DataFrame(np.zeros([6,11]))
diss=features
k=0
for i in range(1,7):
    for j in range(1,12):
        distances=diss.loc[(diss["from.batch"]==i) &
(diss["from.subbatch"]==j),:].copy()
        distances=distances.drop(["from.batch","from.subbatch"],axis=1)
        distances.columns=list(range(0,40))
        distances.index=list(range(0,40))
        covi=covright.loc[(covright["from.batch"]==i) &
 (covright["from.subbatch"]==j),:].copy()
        covi=covi.drop(["from.batch","from.subbatch"],axis=1)
        covi.columns=list(range(0,40))
        covi.index=list(range(0,40))
        ordered_dist_mat, res_order, res_linkage =
 compute_serial_matrix(distances.values, method='single')
        HRP_weights=compute_unif_weights(covi)
        #ds.loc[i-1,:]=HRP_weights
        a=dfright.iloc[:, Id2.iloc[k,:].values].copy()
        a=a[dfright['batch']==i+1]
        dr6.loc[i-1,j-1]=np.dot(HRP_weights,a.apply(sum,axis=0))
        dv6.loc[i-1,j-1]=np.sqrt(np.dot(HRP_weights,covi).dot(HRP_weights))
        des6.loc[i-1,j-1]=Expected_Shortfall(HRP_weights,a,0.05)
        k+=1
dr6.apply(np.mean,axis=1)
```

```
df1=dr1.append(dr2).append(dr3).append(dr4).append(dr5).append(dr6)
df1.to_csv("returnsHRP.csv",index=False)
df2=dv1.append(dv2).append(dv3).append(dv4).append(dv5).append(dv6)
df2.to_csv("volatilityHRP.csv",index=False)
df3=des1.append(des2).append(des3).append(des4).append(des5).append(des6)
df3.to_csv("cvarHRP.csv",index=False)
```

7.5 R: Final Analysis

```
rHRP <- read.csv("~/Datasets/returnsHRP.csv")
volHRP <- read.csv("~/Datasets/volatilityHRP.csv")
esHRP <- read.csv("~/Datasets/cvarHRP.csv")
rHRP=cbind("method"=as.factor(rep(c("DTW","COR","DCOR","MI","FEATURES","Unif"),
each=6)),"year"=rep(1:6,6),rHRP)
volHRP=cbind("method"=as.factor(rep(c("DTW","COR","DCOR","MI","FEATURES","Unif"),
each=6)),"year"=rep(1:6,6),volHRP)
esHRP=cbind("method"=as.factor(rep(c("DTW","COR","DCOR","MI","FEATURES","Unif"),
each=6)),"year"=rep(1:6,6),esHRP)

#mean, standard error according to method

tapply(apply(rHRP[,-c(1,2)],1,mean),rHRP$method,mean)
tapply(apply(rHRP[,-c(1,2)],1,sd),rHRP[,1],mean)

tapply(apply(volHRP[,-c(1,2)],1,mean),volHRP[,1],mean)
tapply(apply(volHRP[,-c(1,2)],1,sd),volHRP[,1],mean)

tapply(apply(esHRP[,-c(1,2)],1,mean),esHRP[,1],mean)
tapply(apply(esHRP[,-c(1,2)],1,sd),esHRP[,1],mean)

(M=tapply(apply(rHRP[,-c(1,2)],1,mean),list(rHRP$method,rHRP$year),mean))
(t=tapply(apply(rHRP[,-c(1,2)],1,mean),rHRP$year,mean))
cor(t(rbind(scale(M,scale=F),t)))

library(RColorBrewer)
library(cowplot)
library(ggplot2)
qplot(df2[df2$year==1,3],df[df$year==1,3],color=df[df$year==1,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+ labs(colour = 'Methods')
qplot(df2[df2$year==2,3],df[df$year==2,3],color=df[df$year==2,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+ labs(colour = 'Methods')
qplot(df2[df2$year==3,3],df[df$year==3,3],color=df[df$year==3,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+ labs(colour = 'Method')
```

```
qplot(df2[df2$year==4,3],df[df$year==4,3],color=df[df$year==4,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+labs(colour = 'Methods')
qplot(df2[df2$year==5,3],df[df$year==5,3],color=df[df$year==5,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+labs(colour = 'Methods')
qplot(df2[df2$year==6,3],df[df$year==6,3],color=df[df$year==6,1],geom="point",
ylab="Returns",xlab="Volatility",size=I(3.5))+ labs(colour = 'Methods')

dd=data.frame(year=rep(2015:2020,6),Method=as.factor(rep(c("DTW","COR","DCOR","MI",
"FEATURES","Unif"),each=6)),returns=apply(rHRP[,-c(1,2)],1,mean))
dd2=data.frame(year=rep(2015:2020,6),Method=as.factor(rep(c("DTW","COR","DCOR","MI",
"FEATURES","Unif"),each=6)),returns=apply(volHRP[,-c(1,2)],1,mean))
dd3=data.frame(year=rep(2015:2020,6),Method=as.factor(rep(c("DTW","COR","DCOR","MI",
"FEATURES","Unif"),each=6)),returns=apply(esHRP[,-c(1,2)],1,mean))

for(i in c("DTW","COR","DCOR","MI","FEATURES","Unif")){
dd[dd$Method==i,3]=cumsum(dd[dd$Method==i,3])
}
for(i in c("DTW","COR","DCOR","MI","FEATURES","Unif")){
dd2[dd2$Method==i,3]=cumsum(dd2[dd2$Method==i,3])
}
for(i in c("DTW","COR","DCOR","MI","FEATURES","Unif")){
dd3[dd3$Method==i,3]=cumsum(dd3[dd3$Method==i,3])
}
ggplot(dd) +
geom_line(aes(x=year, y=returns, color=Method))+
   ylab("Cumulative Returns")+xlab("Year")
```