Bachelorarbeit

Tackling Instability in Markowitz's Minimum Volatility Portfolios

Fakultät II - Mathematik und Naturwissenschaften

Francisco Jose Manjon Cabeza Garcia
29. Dezember 2021

Erstgutachter: Prof. Dr. Jörg Liesen
Zweitgutachter: Dr. Michael Karow

Die selbständige und eigenhändige Anfertigung versichere ich an Eides Statt.

Berlin, den December 29, 2021 Francisco Jose Manjon Cabeza Garcia

Acknowledgements

I am very lucky to be able to write my thesis covering a topic I am passionate about.

I am thoroughly grateful to have had the opportunity to write my bachelor's thesis under the supervision of Prof. Dr. Jörg Liesen, who, unaware, convinced me to continue studying mathematics after a hard start through his linear algebra lecture, and later motivated me to finish my studies with this thesis. He was also instrumental in guiding me through the process of writing this thesis, and was always eager to help and give his time and advice whenever I needed it.

I am very happy to have such amazing friends and family who supported me during my work. Special thanks go to Ekaterina Simukhina for keeping me motivated, and to Ángeles García Escobar, Bernd Schönfelder and Víctor Schönfelder García for convincing and supporting me to study in Germany.

Zusammenfassung

Seit der Veröffentlichung der moderne Portfoliotheorie in 1952, wofür Harry Markowitz einen Nobelpreis für Wirtschaft erhalten hat, hat sich die Vermögensverwaltung grundlegend geändert. Der mathematische Beweis dafür, dass Portfoliorisiko durch die Hinzunahme nicht-perfekt-korrelierter Assets ins Portfolio reduziert werden kann, hat die moderne Portfoliotheorie von Markowitz einen Platz als systematischer Ansatz zur Lösung des Assetallokationsproblem gegeben. Allerdings leiden die Lösungen des mathematischen Optimierungsproblem zur Berechnung von Markowitz optimalen Portfolien an niedrige Streuung, schlechte out-of-sample Performance und Instabilität bzgl. Änderungen der Inputparametern. In dieser Arbeit geben wir einen Überblick des Assetallokationsproblems, definieren das mathematische Optimierungsproblem zur Berechnung von Markowitz optimalen Portfolien und analysieren die Grunde und Quellen der Instabilität. Außerdem, zeigen wir drei Methoden, die die Instabilität reduzieren sollen: zwei beschäftigen sich mit der Kovarianzmatrix, die als Parameter ins Optimierungsproblem reinfliesst, und die dritte Methode ist ein Alternativansatz zu den Assetallokationsproblem. Zuletzt vergleichen wir die Performance bzgl. Streuung, out-of-sample risiko-adjustierter Rendite und zeitlicher Stabilität Markowitz optimaler Portfolio mit der der anderen drei Methoden während des Jahres 2020, als die COVID-19 Pandemie ausbrach.

Abstract

Since the introduction of Modern Portfolio Theory in 1952, which earned Harry Markowitz a Nobel Prize in Economics, the asset management industry has changed a lot. By mathematically proving that portfolio risk can be reduced by simply owning imperfectly correlated assets, his theory has served as a base approach to portfolio construction from a systematic perspective. Nevertheless, the solutions to the underlying mathematical optimisation problem of Markowitz's portfolio construction method present problems with respect to the concentration, out-of-sample performance and stability with respect to the input parameters. Here we give a short overview of the asset allocation problem, define the mathematical optimisation problem linked to Markowitz's optimal portfolios and analyse the sources of its instability. Moreover, we present three methods that try to tackle the instability: two focused on the input covariance matrix and one alternative asset allocation method. Finally, we compare the performance with respect to concentration, out-of-sample risk-adjusted returns and stability across time of Markowitz's minimum volatility portfolios to the portfolios generated by the other three methods during the year 2020, which contains the market stress period caused by the COVID-19 pandemic outbreak.

Contents

1 Introduction

After identifying potential investments, asset managers face the question about how to allocate the limited resources they have to each of them, taking into account the particular constrains and goals of their investors. This task is of great importance for the asset managers' mandate to generate high risk-adjusted returns. Harry M. Markowitz was the first to develop and present a systematic approach to this asset allocation problem in 1952, earning him a Nobel Memorial Prize in Economic Sciences in 1990 [10]. However, Markowitz's portfolio construction model suffers from some pitfalls in practical implementations that limit its performance out-of-sample. For example, DeMiguel et al. [9] have shown that portfolios generated using naive methods like equal weight or inverse variance allocation can even out-perform Markowitz's optimal portfolios. Michaud and Michaud [18] argue that portfolios constructed using Markowitz's model are prone to be unstable, resulting in high transactions costs each time it has to be re-balanced, and also tend to be concentrated into a reduced number of assets.

Several authors have worked on improving Markowitz's portfolio optimisation model. To tackle instability, Ledoit and Wolf [15] have presented a method, called covariance shrinkage, that replaces the sample covariance matrix as input for Markowitz's portfolio optimisation model, because they claim that the sample covariance matrix contains too much noise. Their method relies on using a structured estimator, called shrinkage target, to transform the sample covariance matrix into a matrix with less noise. De Nard [7] has gone further than Ledoit and Wolf to define a covariance matrix shrinkage method that takes into account the idiosyncratic differences between some of the assets available in the portfolio optimisation process. In his method, assets are first clustered into groups with similar characteristics to then apply the covariance shrinkage method to each of the groups separately. Finally, López de Prado [8] has provided a new portfolio optimisation method that tries to tackle the instability of Markowitz's optimal portfolios by introducing the concept of hierarchy among the portfolio assets. He claims that his method, called hierarchical risk parity, not only reduces portfolio instability, but also concentration and it even improves portfolio performance out-of-sample.

We analyse Markowitz's Portfolio Construction and its pitfalls to discuss the source of instability, specially what López de Prado calls *Markowitz's Curse*. Moreover, we present the two approaches to improve stability via covariance matrix shrinkage introduced by Ledoit and Wolf and De Nard. Finally, we describe López de Prado's hierarchical risk parity method. We compare the results of all these optimisation methods with respect to stability (average portfolio turnover), concentration (normalised Herfindahl-Hirsch-Index) and risk-adjusted performance (annualised Sharpe Ratio) on the basis of multi-asset portfolios with the help of ETPs. The details of the selected assets for the practical implementation can be found in appendix A.

Apart from this introduction, this bachelor thesis consists of 3 sections which are briefly described in the following paragraphs.

In section 2 we give a brief introduction to the problem of finding the optimal portfolio for an investor and define 2 naive portfolio allocation models and Markowitz's portfolio allocation model. Moreover, we also analyse the practical results of Markowitz's optimal portfolios and the source of their instability.

In section 3, we define and discuss the 2 covariance shrinkage methods and the method proposed by López de Prado, and compare their practical implementation results to the Markowitz's optimal portfolios from section 2.

Finally, in section 4, we conclude that the sensitivity to the input covariance matrix' condition number in the computation of Markowitz's minimum volatility portfolios is

highly relevant and cannot be ignored, specially given the fact that the condition number can sharply rise and fall in a very short period of time. The covariance shrinkage methods described in this thesis can help reduce the condition number of the input covariance matrix as well as improve its stability across time. Moreover, alternative asset allocation models, like the hierarchical risk parity method, as well as naive models, like equal weight and inverse variance allocation, can be good substitutes for Markowitz's minimum volatility portfolios, even if they are not optimal in-sample.

2 The Asset Allocation Problem

In this section, we provide a general definition of the asset allocation problem, and define two naive solutions as well as Markowitz's solution to this problem. Finally, we present the results of simulating these three allocation methods during the year 2020.

Assume an investor with a total wealth of $\$x \in \mathbb{R}_{\geq 0}$, i.e. a total wealth of x U.S. dollars. Let $A := \{a_1, \ldots, a_N\}$ be the set of assets, which the investor can trade (buy and sell) and let a_i be the real-valued random variable representing the return of asset $i = \{1, \ldots, N\}$. By asset, we understand anything with a price that can be bought and sold so as to profit from price differences across time and that might also produce some cash-flow (coupon or dividend payments), i.e. returns. However, here we will focus on the most widely used investment assets: for example real-estate objects, bonds, stocks, commodities and funds. We will specifically focus on *risky* assets, which are assets such that the corresponding random variable of returns is not constant, and hence present return's variance strictly greater than 0. However, this does not mean that the return's observations of risky assets cannot be constant over a certain period of time.

Definition 2.1. *A **portfolio** P_A **composed of the assets in** A is defined as*

$$P_A = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \in \mathbb{R}^N, \quad \textit{such that} \quad \sum_{i=1}^{N} w_i = 1,$$

where each weight w_i represents the relative allocation of the investor's total wealth to asset $i = \{1, \ldots, N\}$. The real-valued random variable X which represents the returns of the portfolio P_A is given by

$$X = \sum_{i=1}^{N} w_i a_i.$$

One of the most important questions that an investor faces is how much of their total wealth $\$x$ they should allocate to each of the assets in A, i.e. which portfolio $P_A \in \mathbb{R}^N$ they should hold. This question has infinite discretionary and many systematic answers. For example, wealthy investors might prefer to hold a portfolio focused on protecting their total wealth against inflation and other risks, while less affluent investors might prefer a portfolio with a high growth potential at the expense of assuming higher risks.

2.1 Naive Portfolio Allocation Models

One of the most naive answers to this question is to allocate an equal amount of wealth to each asset. Another naive answer is to allocate wealth to each asset according to their return's variance, so that each asset is allocated in inverse proportion to their return's variance.

Definition 2.2 (Equal Weight Portfolio. Page 1922, [9])**.** *Consider the set of investable assets $A = \{a_1, \ldots, a_N\}$. The **equal weight allocation portfolio (EWP)** is defined as*

$$EWP_A = [w_i] \in \mathbb{R}^N_{\geq 0}, \quad \textit{such that} \quad w_i := \frac{1}{N} \quad \textit{for all} \quad i = 1, \ldots, N.$$

The equal weight portfolio is the most widely used allocation method when no information about the assets in the portfolio is available to rank them or to compute some measure to define an optimisation problem. It is also the most naive way of allocating capital across the N assets considered. However, this naive method has proven to perform even better than other, even very sophisticated, methods as shown in [9].

Definition 2.3 (Inverse Variance Portfolio. Pages 431-432 [11]). *Consider the set of investable assets $A = \{a_1, \ldots, a_N\}$ and let $\mathbb{V}[a_i] \in \mathbb{R}$ be the variance of returns of the i-th asset. If $\mathbb{V}[a_i] > 0$ for all $i = \{1, \ldots, N\}$, the* ***inverse variance allocation portfolio (IVP)*** *is defined as*

$$IVP_A = [w_i] \in \mathbb{R}^N_{\geq 0}, \quad \text{such that} \quad w_i := \frac{\frac{1}{\mathbb{V}[a_i]}}{\sum_{k=1}^{N} \frac{1}{\mathbb{V}[a_k]}} \quad \text{for} \quad i = 1, \ldots, N.$$

The IVP can be understood as the allocation method that weights each asset in the portfolio so that each of them contributes the same amount to the total portfolio risk if the returns of assets are considered stochastically independent (assets with higher variance are allocated a smaller portion of the portfolio than assets with lower variance). Moreover, the IVP can only be defined for risky assets, because assets with constant returns have variance equal to 0.

Although IVP is more complex than the EWP model because the estimation of some parameters is needed, it is still considered a naive method.

2.2 Markowitz's Portfolio Optimisation Model

Harry M. Markowitz was the first to develop and present a systematic approach to mathematically define *superior portfolios*, i.e. portfolios that an investor would prefer to hold irrespective of the investor's goals or constrains. Moreover, he gave a mathematical definition for measuring risk by defining it as the standard deviation of past returns and also a mathematical definition for measuring future returns by using the expected value of past returns. His work earned him a Nobel Memorial Prize in Economic Sciences in 1990 [10]. Markowitz's portfolio construction model laid the ground for quantitative asset allocation models. Many asset allocation models today are developed using Markowitz's model, adjusting it to account for modern investors' needs. For example, by using risk measures other than the standard deviation of past returns, or by targeting functions other than expected returns.

Markowitz (page 129, [17]) considers a portfolio *inefficient* if:

1. it is possible to obtain higher expected (or average) return with no greater variability of return,

2. or obtain greater certainty of return with no less average or expected return.

Mathematically, the set of superior portfolios is defined by all portfolios that are not inefficient, which are those such that

1. the expected (or average) return is maximal for a fixed level of return's variance,

2. or the return's variance is minimal for a fixed level of expected or average return.

We will focus on the second definition of efficient portfolios, even though both are equivalent.

The most important parameter in the optimisation problem that defines the elements in the set of superior portfolios that Markowitz defined is the sample covariance matrix of asset returns.

Definition 2.4 (Sample Covariance, Variance, Standard Deviation and Correlation Matrices of Asset Returns). *Let $R = [r_{n,t}] \in \mathbb{R}^{N \times T}$ be the matrix that contains $T \in \mathbb{N}$*

observations of the $\{a_1, \ldots, a_N\}$ real-valued random variables of asset returns. Moreover, let the matrix of centred returns be defined as

$$\tilde{R} = [\tilde{r}_{n,t}] \in \mathbb{R}^{N \times T}, \quad \textit{where} \quad \tilde{r}_{n,t} := r_{n,t} - \frac{1}{T} \sum_{i=1}^{T} r_{n,i}.$$

*Then, the T-**periods sample covariance matrix of asset returns** is defined as*

$$\Sigma = [\sigma_{i,j}] := \frac{1}{T} \left(\tilde{R} \tilde{R}^T \right) \in \mathbb{R}^{N \times N},$$

*where $\sigma_{i,j}$ represents the T-periods sample covariance of the i-th and j-th assets. Finally, we define the **variance matrix** $V = [v_{i,j}] := diag(\sigma_{i,j}) \in \mathbb{R}^{N \times N}$, the **standard deviation matrix** $S = [s_{i,j}] := V^{\frac{1}{2}} \in \mathbb{R}^{N \times N}$ and the **correlation matrix***

$$P = [\rho_{i,j}] := S^{-1} \Sigma S^{-1} \in \mathbb{R}^{N \times N}.$$

Remark 2.5 ([19])**.** *Because we do not know the true mean return of each asset and use the sample mean to compute the sample covariance matrix, we need to use Bessel's correction to get an unbiased estimator of the sample covariance matrix. That is an estimator that is equal to its expected value. Let $T > 1$ and Σ, V, S be the sample covariance, variance and standard deviation matrices defined above respectively, then their unbiased estimators are given by*

$$\begin{aligned} \Sigma_{unbiased} &:= \frac{T}{T-1} \Sigma, \\ V_{unbiased} &:= \frac{T}{T-1} V, \\ S_{unbiased} &:= \frac{T}{T-1} S. \end{aligned}$$

From now on, when we refer to the sample covariance, variance and standard deviation matrices, we refer to them after Bessel's correction.

Now that we have defined the sample covariance matrix of asset returns, we can proceed to define the elements of the set of superior portfolios as the solutions to a mathematical optimisation problem.

Definition 2.6 (Optimal Mean-Variance Portfolio [17])**.** *Consider the set of investable assets $A = \{a_1, \ldots, a_N\}$. Then **Markowitz's optimal portfolio for a given level of return** $\mu_P \in \mathbb{R}$ is given by the solution to the optimisation problem*

$$\begin{aligned} \arg \min_{w \in \mathbb{R}^N} \quad & w^T \Sigma w, \\ s.t. \quad & \mathbb{1}^T w = 1, \\ & \mu^T w = \mu_P, \end{aligned}$$

where $\mathbb{1} := [1, \ldots, 1]^T \in \mathbb{R}^N$, $\Sigma \in \mathbb{R}^{N \times N}$ is the sample covariance matrix of returns and $\mu \in \mathbb{R}^N$ is the vector of expected returns of the N assets.

*The set of superior portfolios given by set of pairs $(\mu_p, \sigma_P^2) \in \mathbb{R} \times \mathbb{R}_{\geq 0}$ such that, for each $\mu_P \in \mathbb{R}$, $\sigma_P^2 := w_{opt}^T \Sigma w_{opt}$, where w_{opt} is the solution to the optimisation problem above, is also called the **efficient frontier**. μ_P is the expected return and σ_P^2 is the variance of the portfolio w_{opt}.*

Please, note that, while the equal weight and inverse variance portfolios are long-only portfolios (all assets are either bought or not bought), Markowitz's optimal portfolios are long-short portfolios, i.e. short-selling[1] is allowed. Mathematically, this means that the optimal solution vector from the optimisation problem in definition 2.6 can contain negative entries, while the equal weight and inverse variance portfolio vectors only contain non-negative entries.

An investor that wants to select an optimal portfolio using Markowitz's model can then decide to either

1. choose a portfolio in the efficient frontier by specifying a target level of portfolio return μ_P (this corresponds to selecting the portfolio with the specified target expected return with the minimum variance of returns),
2. or choose a portfolio in the efficient frontier by specifying a target level of risk σ_P^2 (this corresponds to selecting the portfolio with the specified level of returns' variance with the maximum expected return).

Because most investors do not have very precise target return or target risk level for their portfolios, Markowitz applied the concept of utility function maximisation to the process of portfolio selection. Utility functions can have many forms, but Markowitz defined them as functions of the form

$$U_\alpha(w_P) := \mu_P - \frac{1}{2\alpha}\sigma_P^2 = w_P^T\mu - \frac{1}{2\alpha}w_P^T\Sigma w_P, \quad \alpha \neq 0, \tag{1}$$

using the notation from definition 2.6.

This utility function is a measure of investor happiness. The more risk averse the investor, the less happy the investor will be with high variance of portfolio returns and hence the nearer the value of α will be to 0. Each investor has his/her own utility function, and this utility function can also change over time as the financial situation of the investor changes. Markowitz assumes that all investors are risk-averse. For risk-seeking investors, i.e. investors that prefer portfolios with higher variance for a fixed given level of target return, the factor α would be negative. However, we will ignore risk-seeking ($\alpha < 0$) and risk-indifferent ($\alpha = 0$) investors here.

Markowitz argues that an investor should choose the portfolio that maximises his/her utility function. We can compute such a portfolio by taking the derivative of the general utility function defined in equation 1 and setting it to zero, because the utility function is convex for all $\alpha > 0$. The optimal portfolio is then given by the solution to the linear system

$$\frac{\partial U_\alpha}{\partial w_P} = \mu - \frac{1}{\alpha}\Sigma w_P = 0 \Leftrightarrow \Sigma w_P = \alpha\mu.$$

It is important to mention that the portfolio corresponding to the solution of the linear system above might not add up to 1, and we can understand the parameter α as a factor for the optimal leverage , i.e. if the investor should borrow money to invest in the portfolio P on top of his/her total wealth ($\sum_{i=1}^N w_{Pi} > 1$), invest all its wealth without borrowing ($\sum_{i=1}^N w_{Pi} = 1$) or only invest a part of its total wealth in the portfolio P ($\sum_{i=1}^N w_{Pi} < 1$).

Example 2.7. *Consider a set of investable assets $A = \{a_1, a_2\}$, such that their expected returns' vector is given by $\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and the covariance matrix of returns by $\Sigma = I_2$. Then the utility maximising portfolio is equal to $w_P = \alpha\mu$.*

[1]Short-selling is the act of selling an asset that you do not own by borrowing it and later re-buying it to hand it back to the borrower, so as to profit from the fall in the price of the asset.

Investors with an utility function U_α, $\alpha \in (0, 0.5)$ would only invest a portion (2α) of their total wealth in the portfolio. Investors with an utility function U_α, $\alpha = 0.5$ would invest their complete wealth in the portfolio. Finally, investors with an utility function U_α, $\alpha > 0.5$ would not only invest their complete wealth, but also borrow extra wealth to also invest it in the portfolio.

A common misconception when talking about Markowitz's optimal portfolios is that their computation assumes that returns are normally distributed. This is however wrong, as we have not used this assumption yet, and will not use it later. However, it is true that Markowitz's optimal portfolios are utility maximising only if the utility function of the investor is dependent on first and second order moments of returns, i.e. dependent on mean return and variance of returns. For investors, whose utility function also depends on variables other than portfolio mean return and variance of returns, Markowitz's optimal portfolios might not be optimal. For example, if the investor's utility is also dependent on the maximum temporal loss or the skewness of returns. In that case, Markowitz's optimal portfolios are only optimal if the returns are normally distributed, because the higher order moments can all be express as a function of mean and variance of returns (page 1, [20]).

The main critique point of this optimisation problem is the high sensitivity to small changes in the expected returns vector μ, which, as mentioned before, is defined as the vector of sample mean of the past T-periods asset returns, as well as to small changes in the covariance matrix. Specially small perturbations in the vector of expected returns can lead to significantly different portfolio allocations even with an unchanged sample covariance matrix (pages 185-188, [20]), which in turn leads to a high portfolio turn-over and to concentrated portfolios, especially when correlations rise. Because expected returns are difficult to forecast accurately enough to keep the sensitivity within certain limits, some asset managers abstain from using them as input parameter. Portfolio allocations are then computed on the basis of the sample covariance matrix only, which is a risk parameter, hence the name *risk-based* asset allocation. In the case of Markowitz's portfolio optimisation model, risk-based optimal portfolios correspond to minimum volatility portfolios.

Definition 2.8 (Minimum Volatility Portfolio. Page 3, [20])**.** *Consider the set of investable assets $A = \{a_1, \dots, a_N\}$. Then the* ***minimum volatility portfolio***[2] *is given by the solution to the optimisation problem*

$$\begin{aligned} MMVP_A := \arg\min_{w \in \mathbb{R}^N} \quad & w^T \Sigma w, \\ s.t. \quad & \mathbb{1}^T w = 1. \end{aligned}$$

The minimum volatility portfolio is also a long-short portfolio, i.e. the portfolio might contain negative weights for some assets.

Theorem 2.9. *Assuming that Σ is non-singular, the minimum volatility portfolio allocation vector is given by*

$$w_{opt} := \frac{1}{\mathbb{1}^T \Sigma^{-1} \mathbb{1}} \Sigma^{-1} \mathbb{1}.$$

Proof. We define the Lagrangian corresponding to the minimum volatility portfolio optimisation problem as

$$\mathcal{L}(w, \lambda) := w^T \Sigma w - \lambda(\mathbb{1}^T w - 1).$$

[2]Markowitz defines volatility as the standard deviation of returns. Therefore, the minimum volatility portfolio is also the minimum variance portfolio.

Because a non-singular sample covariance matrix is positive-definite and we have a quadratic optimisation problem, we only need to take the first derivatives with respect to both w and λ, and set them to zero to find the solution of the optimisation problem, i.e.,

$$0 = 2\Sigma w - \lambda \mathbb{1} \Leftrightarrow w = \frac{\lambda}{2}\Sigma^{-1}\mathbb{1},$$
$$\text{and } 0 = 1 - \mathbb{1}^T w \Leftrightarrow \lambda = \frac{2}{\mathbb{1}^T\Sigma^{-1}\mathbb{1}}.$$

Hence,

$$w_{opt} := \frac{\frac{2}{\mathbb{1}^T\Sigma^{-1}\mathbb{1}}}{2}\Sigma^{-1}\mathbb{1} = \frac{1}{\mathbb{1}^T\Sigma^{-1}\mathbb{1}}\Sigma^{-1}\mathbb{1}.$$

□

The analytical expression for w_{opt} presented above contains the inverse of the sample covariance matrix. Because covariance matrices are only positive-semidefinite, i.e. not necessarily non-singular, and because computing the inverse of a matrix is computationally very expensive, we prefer to define w_{opt} as the solution for a linear system.

Algorithm 1: Markowitz Minimum Variance Portfolio Computation

Input : N-Asset return matrix $R \in \mathbb{R}^{N\times T}$
Output: Markowitz Minimum Variance Portfolio w_{opt}

1 Compute the sample covariance matrix $\Sigma \in \mathbb{R}^{N\times N}$;
2 **if** Σ *is non-singular* **then**
3 | Set $\tilde{w}_{opt} \in \mathbb{R}^N$ to the solution to $\Sigma w = \mathbb{1}$;
4 **end**
5 **else**
6 | Set $\tilde{w}_{opt} \in \mathbb{R}^N$ to one of the solutions to the least-squares problem $\Sigma w = \mathbb{1}$;
7 **end**
8 Set $w_{opt} = \frac{1}{\mathbb{1}^T\tilde{w}_{opt}}\tilde{w}_{opt}$;

Remark 2.10. *For the special case of a covariance matrix that is invertible and diagonal, i.e. if all assets are uncorrelated and with return variances strictly greater than zero, the inverse variance allocation portfolio is equivalent to the Markowitz minimum variance portfolio. This is because in such case, we would only need to solve N linear equations at step 3 of algorithm 1 of the form*

$$\sigma_{i,i}w_i = 1 \quad \textit{for} \quad i = \{1, \dots, N\},$$

where $\sigma_{i,i} = \mathbb{V}[a_i] \in \mathbb{R}_{>0}$ is the return's variance of the i-th asset which lies in the diagonal of Σ.

Finally, at step 8, we get that

$$w_{opt} = \frac{1}{\sum_{k=1}^N \frac{1}{\sigma_{k,k}}}\begin{bmatrix}\frac{1}{\sigma_{1,1}}\\ \vdots \\ \frac{1}{\sigma_{N,N}}\end{bmatrix} = \frac{1}{\sum_{k=1}^N \frac{1}{\mathbb{V}[a_k]}}\begin{bmatrix}\frac{1}{\mathbb{V}[a_1]}\\ \vdots \\ \frac{1}{\mathbb{V}[a_N]}\end{bmatrix} = IVP_A.$$

2.2.1 Stability and Markowitz's Curse

Note that if Σ is singular, we can always use the least-squares method to get an optimal portfolio at step 6 of algorithm 1. However, in that case, only $\|\Sigma\tilde{w}_{opt} - \mathbb{1}\| = \min_{w\in\mathbb{R}^N}\|\Sigma w - \mathbb{1}\|$ holds, and $\tilde{w}_{opt}$ might not be unique, although at least one such $\tilde{w}_{opt}$ always exists. In practical applications singular matrices can appear in two cases:

1. If the observed returns of some asset are constant over the whole look-back period. This can happen even if the random variable of the asset returns is not constant. For example, let a_1, a_2 be two real-valued random variables representing the returns of two assets. If a_1 is constant, then the covariance of asset returns is given by

$$Cov[a_1, a_2] = \mathbb{E}[(a_1 - \mathbb{E}[a_1])(a_2 - \mathbb{E}[a_2])] = \mathbb{E}[0(a_2 - \mathbb{E}[a_2])] = 0.$$

   Moreover, if a_1 is not constant, but its last T observations are all equal, then, using definition 2.4, the sample covariance matrix is of the form

$$\Sigma = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{T}\sum_{t=1}^{T}(\bar{r}_{2,t})^2 \end{bmatrix},$$

   because the first row of $\tilde{R}$ contains zeros only.

2. If two or more assets present very similar returns' timeseries, leading to a very high correlation between them in absolute terms. Mathematically, this corresponds to two centred vectors of past returns that are either linearly dependent or *almost* linearly dependent. For example, let $\tilde{r}_1, \tilde{r}_2 \in \mathbb{R}^T$ the vectors of past T centred return's observations of two assets, such that $\langle \tilde{r}_1, \tilde{r}_2 \rangle = \|\tilde{r}_1\|_2 \|\tilde{r}_2\|_2 (1 - \epsilon)$, for some small $\epsilon > 0$. Then the sample covariance matrix computed as in definition 2.4 using the two centred sample return's vectors is given by

$$\begin{aligned} \Sigma &= \frac{1}{T} \begin{bmatrix} \|\tilde{r}_1\|_2^2 & \|\tilde{r}_1\|_2 \|\tilde{r}_2\|_2 (1-\epsilon) \\ \|\tilde{r}_1\|_2 \|\tilde{r}_2\|_2 (1-\epsilon) & \|\tilde{r}_2\|_2^2 \end{bmatrix} \\ &= \frac{1}{T} \begin{bmatrix} \|\tilde{r}_1\|_2 & 0 \\ 0 & \|\tilde{r}_2\|_2 \end{bmatrix} \begin{bmatrix} 1 & (1-\epsilon) \\ (1-\epsilon) & 1 \end{bmatrix} \begin{bmatrix} \|\tilde{r}_1\|_2 & 0 \\ 0 & \|\tilde{r}_2\|_2 \end{bmatrix} \end{aligned}$$

   The smaller ϵ is, the *more singular* Σ is. This is a problem, even if Σ is still invertible, as we will shortly see. Moreover, we will see in section 3.2 the meaning of the assumption $\langle \tilde{r}_1, \tilde{r}_2 \rangle = \|\tilde{r}_1\|_2 \|\tilde{r}_2\|_2 (1 - \epsilon)$.

   This is actually a field of study for developing trading strategies called pairs trading. It consists in identifying pairs of assets whose returns are driven by very similar factors. An example would be Coca-Cola and Pepsi stocks or BMW and Daimler stocks, because they are stock pairs of very similar companies. Once such pairs are identified, pairs trading strategies try to profit from divergences in the return's timeseries of the two assets. For example, if Coca-Cola publishes its earnings report one week before Pepsi and the earnings happen to be better than expected, then the Coca-Cola stock price will most probably rise. Because of the similarities between the two companies, we can expect the earnings of Pepsi to also be better than expected, and already buy its stock while short selling the stock of Coca-Cola. When Pepsi publishes its earnings reports the following week, we can close both positions, and profit from the convergence in their stock-price timeseries to the previous ratio (the Coca-Cola stock price won't react to the earnings or Pepsi, but Pepsi's stock price will). Of course, the profit will depend on Pepsi actually reporting better than expected earnings. Short-selling Coca-Cola's stock is necessary to protect us from price fluctuations that are caused by the general market moving up or down, and that have nothing to do with the idiosyncrasies of the two companies. For example, if both stock prices fall or rise together in the meantime. For these type of strategies, correlations do only help identify potential pair candidates, and cointegration is the actual phenomena that is traded.

Now that we have explained why singular covariance matrices are a problem and can appear in practical applications relatively often, we can focus on non-singular covariance matrices.

Some non-singular matrices are *harder* to invert and the solutions to linear programs given by them tend to contain *more noise* than in the case of some other non-singular matrices. To quantify these two concepts mathematically, we turn to numerical linear algebra error theory. We can analyse the instability of the solution computed at step 3 in algorithm 1 using the forward error bound of the solution based on the condition number of the input covariance matrix (in this case assumed to be non-singular).

Definition 2.11 (Condition Number of a Matrix. Page 94, [22]). *If $A \in \mathbb{C}^{N\times N}$ is non-singular, and $\|\cdot\|$ is a norm on $\mathbb{C}^{N\times N}$, then*

$$\kappa(A) := \|A\|\|A^{-1}\|$$

is called the ***condition number of A with respect to the norm $\|\cdot\|$.***

For the results presented here, we compute the condition number with respect to the spectral norm, i.e. with respect to the matrix norm induced by the Euclidean vector norm.

Theorem 2.12 (Residual-based forward Error Bound. Page 61, [1]). *Let $A \in \mathbb{C}^{N\times N}$ be non-singular, $x \in \mathbb{C}^N \setminus \{0\}$ and $b = Ax$. Then for every $\tilde{x} \in \mathbb{C}^N$ we have*

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \kappa(A)\frac{\|r\|_2}{\|b\|_2},$$

where $r := b - A\tilde{x}$ is the residual and $\|r\|/\|b\|$ is the relative residual norm.

Proof. Using $x = A^{-1}b$ and the definition of the residual get

$$\|\tilde{x} - x\|_2 = \|A^{-1}(A\tilde{x} - b)\|_2 \leq \|A^{-1}\|\|r\|_2 = \kappa(A)\frac{\|r\|_2}{\|A\|_2}.$$

Moreover, $\|b\|_2 = \|Ax\|_2 \leq \|A\|_2\|x\|_2$ gives

$$\frac{1}{\|x\|_2} \leq \frac{\|A\|_2}{\|b\|_2},$$

which implies the desired inequality. □

As we can see, the forward error of the solution computed depends, not only on the relative residual norm, but also on the condition number of the matrix that defines the linear system, in our case the sample covariance matrix. That means, that even if the relative residual norm of the solution is *very small*, i.e. the computed solution appears to be very near to the actual solution, a *large* sample covariance matrix condition number implies that the computed solution might actually be very different to the actual solution.[3]

Theorem 2.13 (Page 94, [22]). *Let $A \in \mathbb{C}^{N\times N}$ be a non-singular symmetric matrix and $\lambda_1, \ldots, \lambda_N$ its eigenvalues, then the condition number of A with respect to the spectral norm is given by*

$$\kappa_2(A) := \frac{\max_{i=\{1,\ldots,N\}} |\lambda_i|}{\min_{i=\{1,\ldots,N\}} |\lambda_i|}.$$

[3]The terms *very small* and *large* are, obviously, no mathematical terms. A mathematical definition depends on the problem set-up and the error tolerance that can be assumed according to the practical use of the solution. Here, we would like to focus on how changes in the condition number lead to portfolio instability, rather than on how a high or low condition number in absolute terms affects the use of the computed solution.

Proof. If $A \in \mathbb{C}^{N\times N}$ is a non-singular symmetric matrix, then we know that A can be unitarily diagonalised:

$$A = U\Lambda U^H,$$

where $U \in \mathbb{C}^{N\times N}$ is an unitary matrix and $\Lambda := diag(\lambda_1, \ldots, \lambda_N) \in \mathbb{C}^{N\times N}$ is a diagonal matrix that contains the eigenvalues of A in the diagonal. Using the fact that the spectral norm is unitarily invariant, we get

$$\begin{aligned}
\kappa_2(A) &= \|A\|_2\|A^{-1}\|_2 \\
&= \|U\Lambda U^H\|_2\|(U\Lambda U^H)^{-1}\|_2 \\
&= \|\Lambda\|_2\|\Lambda^{-1}\|_2 \\
&= \left(\max_{i=\{1,\ldots,N\}} |\lambda_i|\right)\left(\frac{1}{\min_{i=\{1,\ldots,N\}} |\lambda_i|}\right). = \frac{\max_{i=\{1,\ldots,N\}} |\lambda_i|}{\min_{i=\{1,\ldots,N\}} |\lambda_i|}.
\end{aligned}$$

□

We now focus on the correlation matrix as source of instability. Using the decomposition of the covariance matrix from definition 2.4

$$\Sigma = SPS,$$

where S is the standard deviation matrix and P the correlation matrix, and the result of theorem 2.13, we see that condition number of the covariance matrix is minimal for a fixed standard deviation matrix when the correlation matrix is a diagonal matrix. This follows from the fact that the diagonal entries of the correlation matrix are always equal to 1, because all assets are perfectly correlated to themselves, hence the only diagonal correlation matrix that exists is the identity matrix. A diagonal correlation matrix corresponds to a set of assets that are perfectly uncorrelated. As the relative difference in the standard deviation of the assets and their correlations rise, the condition number of the corresponding sample covariance matrix also rises. This tends to happen during periods of market stress, when investors fear the fall of asset prices across asset classes, sectors and countries. However, it is exactly during those periods of market stress when investors do need to make the most out of diversification to reduce the impact of price fall in their aggregated portfolio. This issue is what López de Prado calls *Markowitz's curse*:

> "The more correlated the investments, the greater the need for diversification, and yet the more likely we will receive unstable solutions. The benefits of diversification often are more than offset by estimation errors." (Page 3, [8])

Moreover, periods of market stress tend to start suddenly, and investors reactions are drastic, which in turn can cause sudden sharp rises in the condition number of the covariance matrix. As a result, the covariance matrices used to compute the optimal portfolios at the last and first rebalancing point just before and just after the period of market stress respectively tend to be very different, potentially generating very different optimal allocations. This is one of the reasons why Markowitz's minimum volatility portfolios tend to be more unstable across time, generating higher transaction costs.

Apart from rises in correlations, the condition number of the covariance matrix is affected by the number of assets in the set of investable assets and by the number of past return's observations used for its computation. To illustrate how the condition number of the covariance matrix rises as not-perfectly-uncorrelated assets, i.e. correlated assets,

are added to the set of investable assets we want to consider to construct our portfolio, we have performed a simulation. We took the components of the S&P 500 index, which is an index that tracks the 500 biggest companies in the US by free floating market capitalisation (number of shares outstanding that can be traded times the current price of one share), as of October 2021. In total, we had 505 companies in the index (the index does not always contain exactly 500 companies). We computed the covariance and the corresponding correlation matrix of the assets by adding them one by one to the set of investable assets in random order. We performed this simulation using the return's data of the past 510, 765, 892 and 1020 days from the December 24, 2021. These correspond to approximately 2, 3.5, 4 and 4.5 years worth of trading data (one year contains around 255 days of trading, the rest are weekends and holidays). The results are plotted in figures 1, 2, 3 and 4.

In these figures, we can observe several interesting findings:

1. The number of assets in each simulation run decreases as the number of past daily-return's observations rises. Of the 505 assets in the S&P 500 index, there is enough data to compute covariance and correlation matrices with 255 past daily-return's observations for 497 assets. This number decreases to 488 for the run with 765 daily-return's observations, to 486 for the run with 892 daily-return's observations and to 484 assets for the run with 1020 daily-return's observations.

2. There are some big jumps in the condition number at some point in all figures. This is caused by the addition of a couple of assets that are highly correlated to the others (jumps in the condition number of the correlation matrix) or that present very high variance of returns (jumps in the covariance matrix). This happens at different points in each of the figures, because the random order in which assets were added to the computation was not the same across all simulations, however they are caused by the same assets.

3. A higher number of past daily-return's observations used in the computation of covariance and correlation matrices resulted in lower condition numbers. This is not a surprise, because using a matrix R as in definition 2.4 of bigger size means more degrees of freedom for the computation of covariances and correlations. Hence, the invertibility of the covariance matrix is not guaranteed, but it is more probable, simply because the probability of falsely estimating that two assets are highly correlated due to low number of observations is lower. Nevertheless, using more past return's observations is not always possible. In the simulation plotted in figure 4, the ratio between past return's observations and number of assets is around 2.1. If an asset manager like the Norwegian Sovereign Wealth Fund had to use such a ratio, they would need more than 74 years of past return's data, because their portfolio contains holdings in more than 9000 companies (as claimed in the website of the Norges Bank Investment Management `https://www.nbim.no/` as of Dec. 28, 2021). Needless to say that this is impossible, because most companies have not existed for so long. Other asset managers, specially pension funds, do also hold portfolios with similar number of assets. Finally, using a higher number of past return's observations has its disadvantages, as we will see later in section 3.1.

4. There is a clear trend in the condition numbers, which rise exponentially or almost exponentially as the number of assets increases.

Now that we have defined some theory for the stability of Markowitz's minimum volatility portfolios, we can proceed to show the results of the practical implementation.

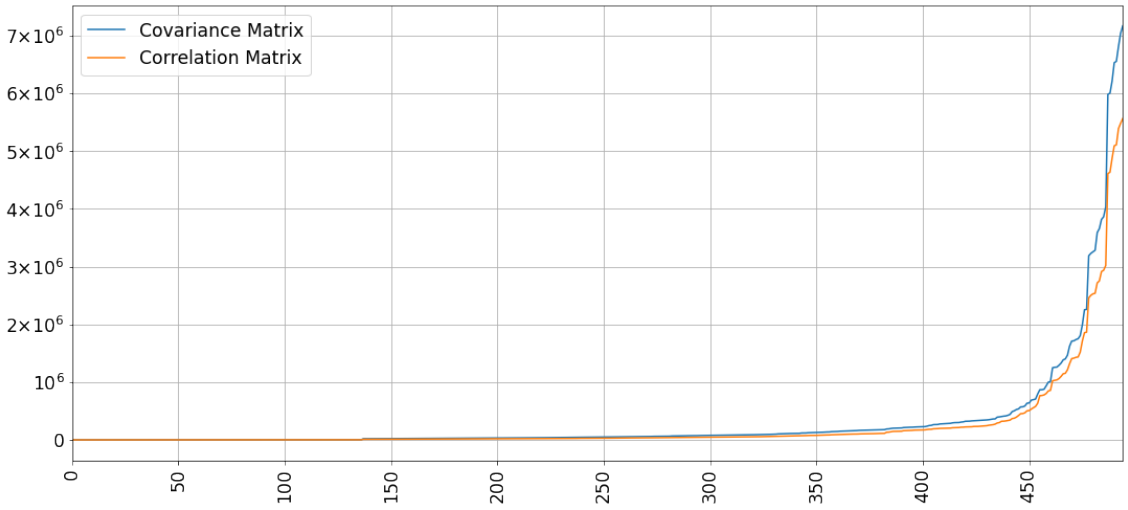

Figure 1: Condition Number of Covariance and Correlation Matrices Computed by Adding 497 Assets One by One to the Set of Investable Assets and Using 510 Days Past Return's Observations until 24/12/2021.

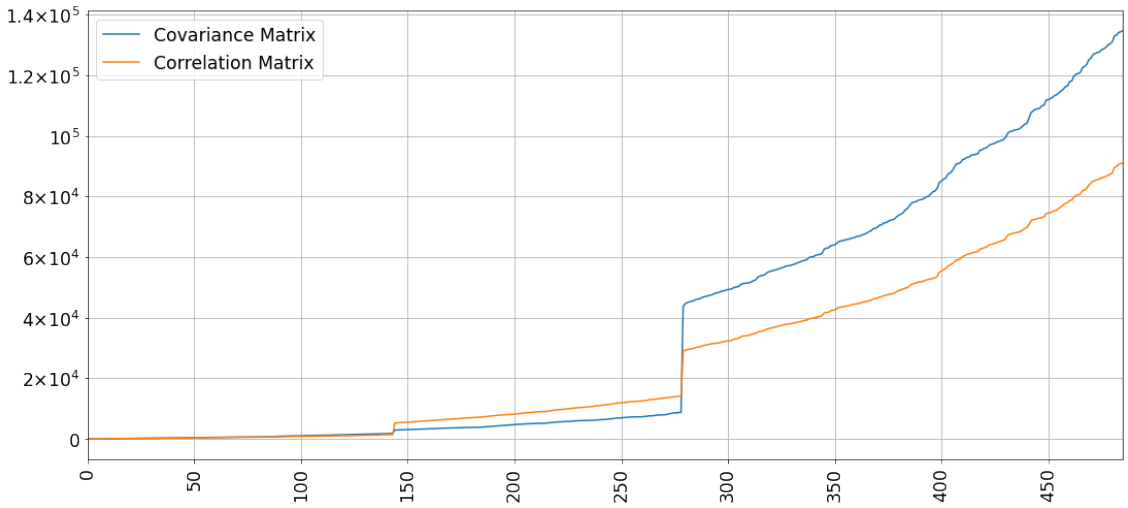

Figure 2: Condition Number of Covariance and Correlation Matrices Computed by Adding 488 Assets One by One to the Set of Investable Assets and Using 765 Days Past Return's Observations until 24/12/2021.

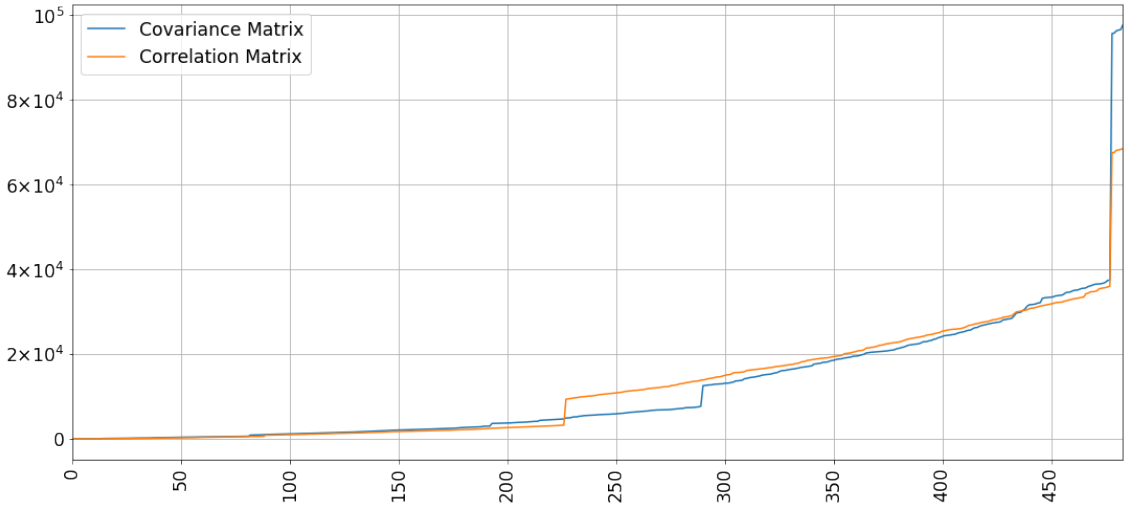

Figure 3: Condition Number of Covariance and Correlation Matrices Computed by Adding 486 Assets One by One to the Set of Investable Assets and Using 892 Days Past Return's Observations until 24/12/2021.

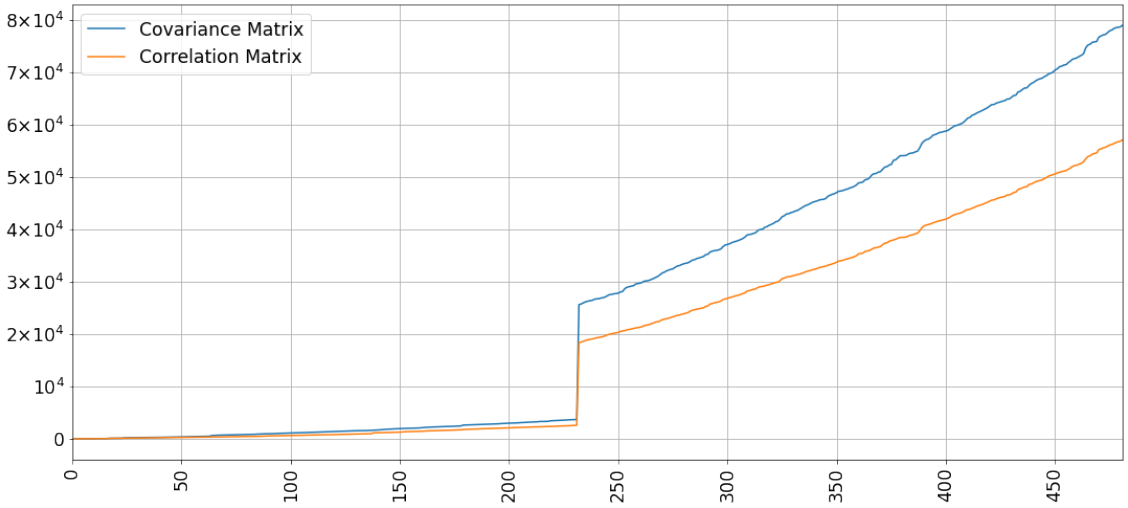

Figure 4: Condition Number of Covariance and Correlation Matrices Computed by Adding 484 Assets One by One to the Set of Investable Assets and Using 1020 Days Past Return's Observations until 24/12/2021.

2.3 Practical Implementation of Naive Portfolio Allocation Methods

In this subsection, we present the results of the practical implementation of the naive portfolio allocation methods described in definitions 2.2 and 2.3, which will serve as benchmarks for the Markowitz's minimum volatility portfolios.

In total, 12 portfolios are computed, corresponding to a backtesting period from January 1, 2020 to December 31, 2020 and monthly rebalancing-interval. The rebalancing occurs on the first trading day of each month at 10:00 AM New York time. An initial account balance of 10 million U.S. dollars is assumed.

For the computation of portfolio weights, the daily returns of the past 60 days are used as input. This last setting does not affect the equal weight portfolio construction method, because it does not use any past data as input. The reason why we use 60 days of past data is because we need at least 20 periods of data to fulfil a necessary condition for the invertibility of the sample covariance matrix, and 60 days corresponds to 3 full trading months of data.

In our practical implementation framework, we have selected $N = 20$ assets, therefore $T \geq 20$ must hold. This does however not guarantee that the covariance matrix has full rank and hence is invertible, because the condition $N \leq T$ is not a sufficient condition for that. For more details about the assets selected, see appendix A.

There is not a single true underlying covariance matrix, but rather one that changes over time according to human behaviour and psychology, else every investor could compute optimal portfolios that would remain optimal over time. Therefore, there is a trade-off between higher values of T, which lead to a more stable, with higher probability non-singular sample covariance matrix over time but at the same time make the portfolio optimisation less sensitive to changes in market trends and regimes, and lower values of T, which make the covariance matrix estimation process more sensitive to changes in market trends and regimes but make the portfolio optimisation process less stable. The asset manager should always take this into account, in order to reduce noise in the estimation while still using input parameters that allow to promptly react to changes in the market. In the case of our practical implementation framework, we could, of course, have chosen a different value for T, and we do not have any mathematical reason to prefer $T = 60$ to $T = 59$ or $T = 100$, for example. However, optimising the covariance matrix estimation process with respect to the number of observations used for its computation is not part of this work, and a 3-month past-data-window seems acceptable for the estimation of portfolios with a holding period of one month (optimal portfolios are computed and rebalanced each month).

The portfolios generated in backtesting by each of the naive optimisation methods and the other methods defined later are compared with respect to risk-adjusted return, concentration and stability. The Sharpe ratio is the chosen measure to quantify risk-adjusted returns. To assess concentration, we will use a normalised version of the Herfindahl-Hirsch-Index (HHI). The normalisation step is necessary, because we will allow for short positions in our portfolios, and the HHI is only defined for positive portfolio weights. We will measure stability via the turnover of the portfolio.

Definition 2.14 (Sharpe Ratio. Page 216, [11])**.** *Let $\mu_P \in \mathbb{R}$ be the return of the portfolio P over a given period of time and $\sigma_P \in \mathbb{R}_{\geq 0}$ its standard deviation over that same period of time. Moreover, let $R_f \in \mathbb{R}$ be the risk-free rate of that given period. Then the* ***Sharpe Ratio of portfolio P over the given period*** *$SR_P \in \mathbb{R}$ is given by*

$$SR_P := \frac{\mu_P - R_f}{\sigma_P}.$$

The risk-free rate is the return of the safest asset available, usually cash deposits at a commercial or central bank or the return of some high-quality short-term government bond like the U.S. treasury bill (T-Bill) or the German Federal Treasury notes (Schatz).

Remark 2.15. *We do not compute the Sharpe ratio of the backtest separately, and rely upon QuantConnect's internal overall portfolio statistics for its computation, i.e. all Sharpe ratios presented here are the Sharpe ratios that appear in QuantConnect's backtesting results. QuantConnect computes the Sharpe ratio using the same definition as above, but using the annualised return of the portfolio in the backtest and assuming a risk-free rate of* 0% *[5].*

Definition 2.16 (Herfindahl-Hirsch-Index. Page 71, [2])**.** *Let $P = [w_i] \in \mathbb{R}^N_{\geq 0}$ be a portfolio, such that $\sum_{i=1}^N w_i = 1$. We define the* ***Herfindahl-Hirsch-Index (HHI)*** *as*

$$HHI_P := \|P\|_2^2.$$

For the general case of a portfolio $P = [w_i] \in \mathbb{R}^N$, we define the ***normalised Herfindahl-Hirsch-Index (nHHI)*** *as*

$$nHHI_P := \left(\frac{\|P\|_2}{\|P\|_1}\right)^2.$$

Obviously, $nHHI_P \leq HHI_P$ for all portfolios $P \in \mathbb{R}^N$.

Definition 2.17 (Turnover. Page 1929, [9])**.** *Consider a portfolio P with N assets. Let $w_{j,t+1} \in \mathbb{R}$ be the portfolio weight assigned to asset i at time t by the portfolio optimisation method, and let $w_{j,t^+} \in \mathbb{R}$ be the portfolio weight of asset i before rebalancing at time $t+1$ for $i = 1, \ldots, N$ and $t = 0, \ldots, T$. The turnover of the portfolio P is defined as*

$$TO_P := \frac{1}{T} \sum_{t=0}^{T} \sum_{i=1}^{N} \left(\left| w_{i,t+1} - w_{i,t^+} \right| \right),$$

where $T \in \mathbb{N}$ is the number of rebalancing points of the portfolio.

This turnover quantity can be interpreted as the average percentage of wealth traded at each rebalancing point.

Now that we have defined the 3 measures which we will use to compare the portfolios resulting from the different portfolio construction models across the backtesting period, we proceed to present the results of the 2 naive portfolio optimisation models from definitions 2.2 and 2.3. The vectors with the optimal portfolios computed using these 2 naive methods can be found in appendices C and D.

In figure 5, a visual representation of the target allocation of all 20 assets in the IVP over the backtesting period is presented. Most of the portfolio's wealth is allocated to ISTB (iShares Core 1-5 Year USD Bond ETF) and IAGG (iShares Core International Aggregate Bond ETF). The allocation to the first drops at the start of the COVID-19 crisis, while the allocation of the second rises. The allocation to IMTB (iShares 5-10 Year USD Bond ETF) rises after the start of the COVID-19 crisis. For the rest of assets the allocation remains relatively stable over the whole backtesting period. Finally, the high allocation to fixed income ETFs is not surprising, because they tend to show lower variance of returns, while the allocation to equity, including real-estate, and commodity ETPs remains low due to the higher variance of returns that they tend to present.

A similar plot for the EWP target allocation over time is not necessary, since the target allocation to each asset is equal and remains the same over the whole backtesting period.

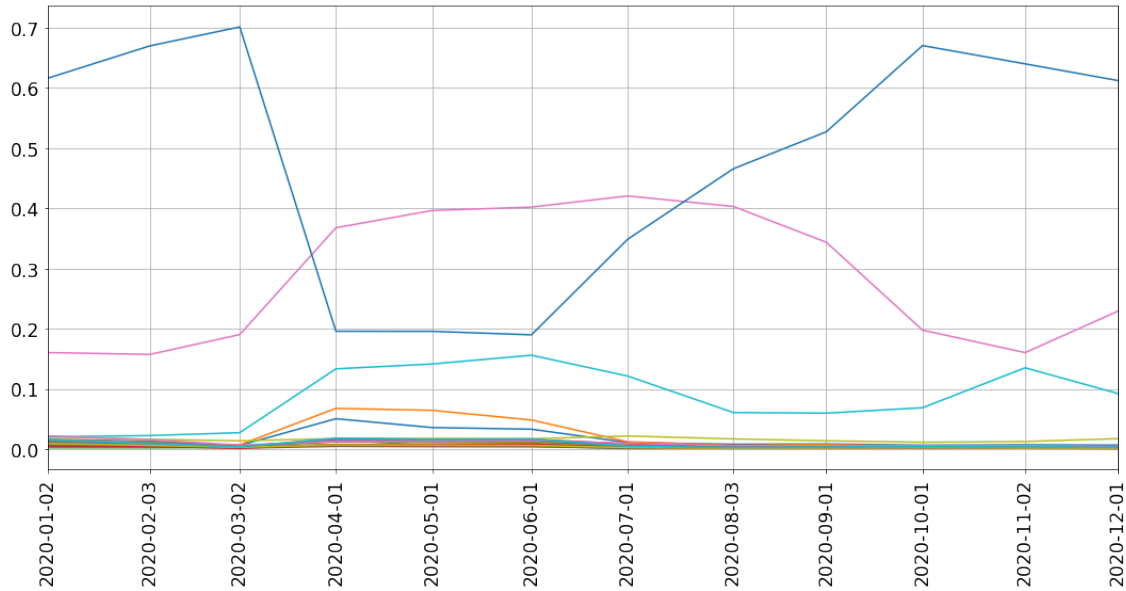

Figure 5: IVP Target Allocation of the Practical Implementation Portfolio Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

Figure 6 shows the evolution of the nHHI of the 2 naive portfolio allocation models described above. The nHHI of the equal weight portfolios is low and stays the same over time, as expected. In the case of IVP, the nHHI rises with the start of the COVID-19 crisis, and then halves, mostly because of a reduction in the allocation to ISTB as explained previously.

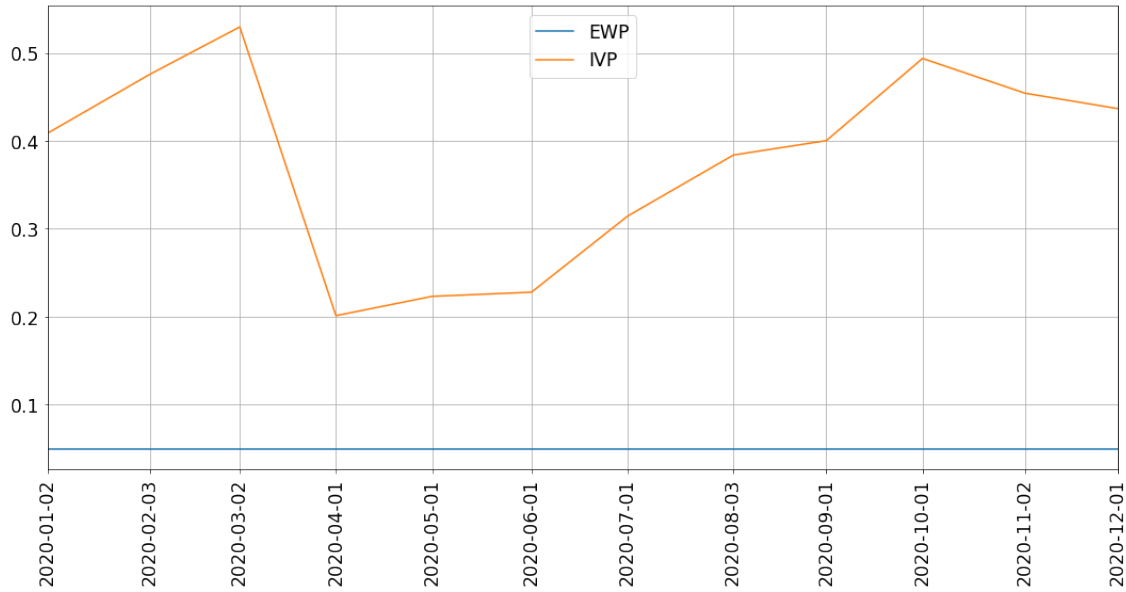

Figure 6: nHHI of the Equal Weight and Inverse Variance Portfolios Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

Table 1 presents the risk-adjusted returns and turnover of both naive strategies. Inverse variance portfolios achieve a Sharpe ratio almost three times higher than the Sharpe ratio of the equal weight portfolios. The turnover of the IVP is also higher than the turnover of the EWP by a factor of 2.

Portfolio Optimisation Method	**Sharpe Ratio**	**Turnover**
EWP	0.36	0.003634
IVP	1.008	0.007396

Table 1: Sharpe Ratio and Turnover Figures of the Equal Weight and Inverse Variance Portfolios for the Backtesting Period from 01/01/2020 to 31/12/2020.

2.4 Practical Implementation of Markowitz's Minimum Volatility Portfolios

Now, we continue with the results of the implementation of Markowitz's minimum volatility portfolios from definition 2.8. We refer to section 2.3 for details about the backtesting settings and preformance measures, as well as to appendix A for details about the 20 selected assets and to appendix E for the 12 computed Markowitz's minimum volatility portfolios. But first, we describe how the algorithm 1 was implemented in detail.

For the practical implementation, we use SciPy's linear system solver which can take certain structural properties of the quadratic matrix that defines the linear system, and uses that information to select the method with which to solve the linear system. We take advantage of the fact that the sample covariance matrix is always symmetric and positive-semidefinite. In case the sample covariance matrix has full rank, then it is even positive-definite. Hence, we first assume the sample covariance matrix to be positive-definite, and use this structural property as input for the `scipy.linalg.solve` method. As a result, the solver will try to solve the linear equation system in definition 2.8 using the *LAPACK ?POSV linear equation routine* [4], which tries to solve linear equation systems using the Cholesky decomposition [12]. In case the method fails to compute a solution to the linear system, then we know that the sample covariance matrix does not have full rank, in which case the practical implementation turns to `numpy.linalg.lstsq`, which is a NumPy method to compute the least-squares solution to a linear matrix equation [3]. However, all the covariance matrices computed during the backtesting period have full rank, and the least-squares solver was not used to compute any of the minimum variance portfolios presented here.

Theorem 2.18 (Cholesky Decomposition. Pages 307-308, [16])**.** *Let $A \in \mathbb{C}^{N\times N}$ be a symmetric positive-definite matrix, then there exists a uniquely determined lower triangular matrix $L \in \mathbb{C}^{N\times N}$ with positive diagonal elements, such that*

$$A = LL^H. \tag{2}$$

Figure 7 shows the evolution of allocations in Markowitz minimum volatility portfolios over time. The changes in allocation are clearly more pronounced than in the case of IVPs (see figure 5), specially around the start of the COVID-19 (March 2020) crisis when correlations as well as the variances of returns rose and all asset classes suffered negative returns.

The concentration of MMVPs measured by the nHHI remains however lower than in IVPs over the whole backtesting period as figure 8 shows. Nevertheless, a drop in concentration can be observed following the start of the COVID-19 crisis.

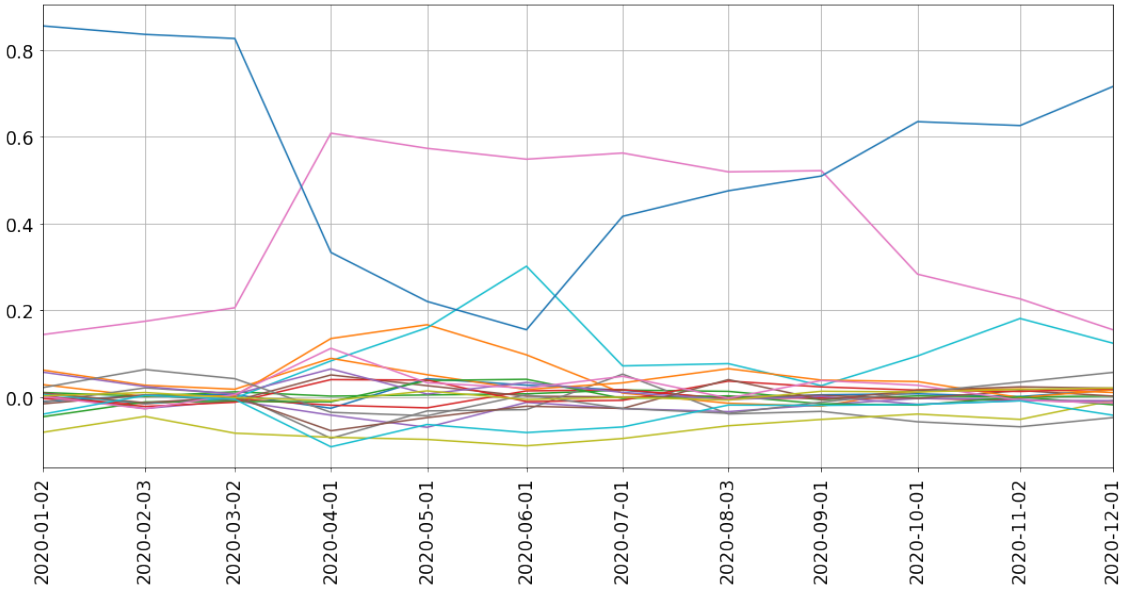

Figure 7: MMVP Target Allocation of the Practical Implementation Portfolio Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

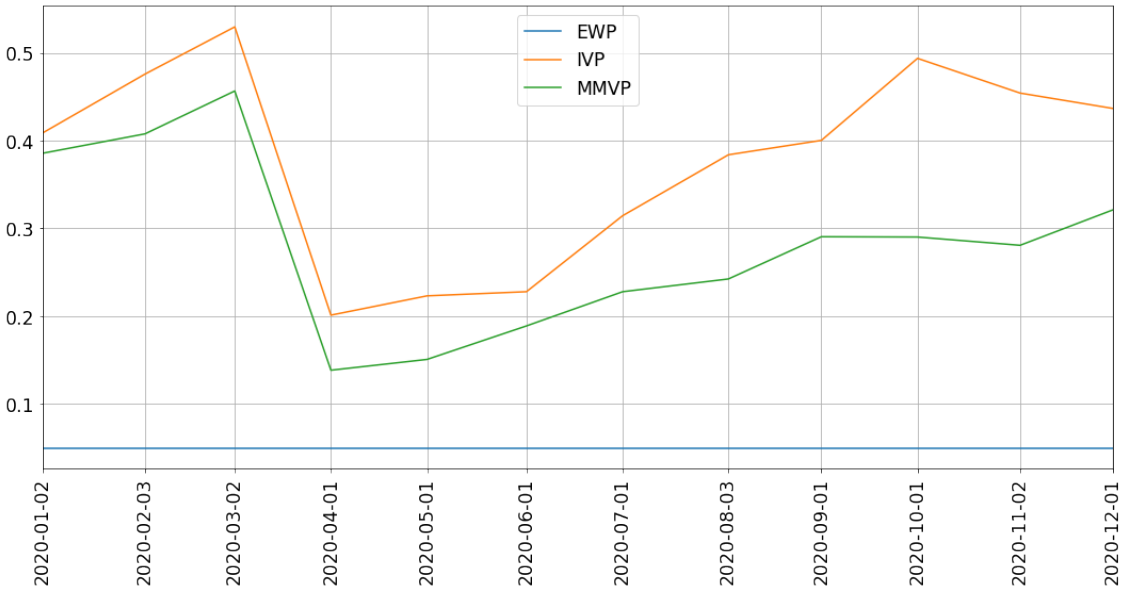

Figure 8: nHHIn of the Equal Weight, Inverse Variance and Markowitz Minimum Volatility Portfolios Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

With respect to risk-adjusted returns, MMVPs deliver significantly worse results than both naive allocation methods. The Sharpe ratio of Markowitz's strategy is even negative, i.e. MMVPs fail to generate positive returns during the backtesting period. Moreover, the turnover is also significantly higher than in the case of both naive portfolio optimisation models, as seen in table 2.

Portfolio Optimisation Method	**Sharpe Ratio**	**Turnover**
EWP	0.36	0.003634
IVP	1.008	0.007396
MMVP	-0.017	0.018856

Table 2: Sharpe Ratio and Turnover Figures of the Equal Weight, Inverse Variance Portfolios and Markowitz Minimum Volatility for the Backtesting Period from 01/01/2020 to 31/12/2020.

In section 2.2.1 we have laid out numerical linear algebra error theory applied to the analysis of Markowitz's model stability. We now quantify the sources of instability and instability itself in the 12 portfolios computed in our backtest.

To quantify how correlated assets are, we define the following distance measure for the distance between each of the computed correlation matrices corresponding to each sample covariance matrix of our practical implementation and the *ideal* correlation matrix[4], i.e. the identity matrix.

Definition 2.19. *Let $\Omega \in \mathbb{R}^{N\times N}$ be a correlation matrix as in definition 2.4. We define the* ***distance of Ω to the ideal correlation matrix*** *as*

$$\|\Omega - I_N\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm and I_N is the N-dimensional identity matrix. Obviously,

$$\|\Omega - I_N\|_F \geq 0$$

holds for all correlation matrices $\Omega \in \mathbb{R}^{N\times N}$.

The distance of a correlation matrix to the ideal correlation matrix that we have just defined is a measure of how much different assets are correlated (positively as well as negatively). The more correlated assets are, the higher the distance to the ideal correlation matrix, which corresponds to all assets being completely uncorrelated.

Figure 9 shows the change in the distance of the correlation matrices corresponding to the computed covariance matrices over the backtesting period. We can appreciate a steep rise in the distance coinciding with the start of the COVID-19 crisis, showing how strong the correlations between the 20 assets in our universe rose. Correlations remained high until October, and later failed to get back down to the levels observed at the beginning of the backtesting period. In this figure, we can also observe that the rises can be very sudden and steep.

We quantify the condition number with respect to the spectral norm of the covariance matrices used for the computation of Markowitz minimum volatility portfolios in

[4]We consider the identity matrix as the ideal correlation matrix because it maximises the stability of the optimisation method, as we saw in section 2.2.1. Furthermore, of all possible correlation matrices, a set of perfectly uncorrelated assets used as input for Markowitz's portfolio optimisation method results in the portfolio with the lowest variance. For other purposes, a correlation matrix equal to the identity matrix might not be ideal.

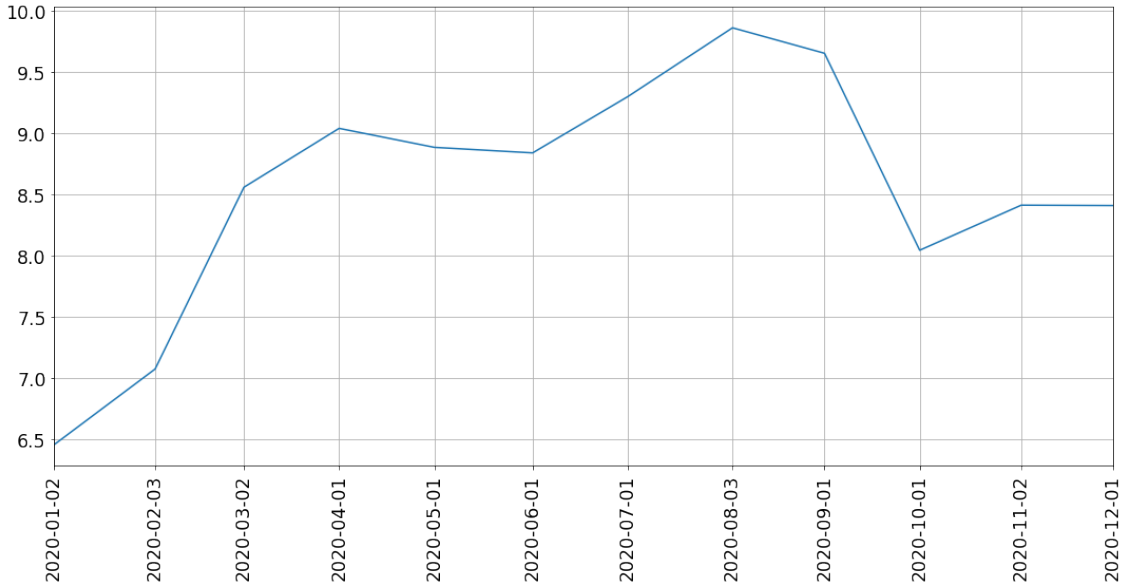

Figure 9: 20 Assets Correlation Matrix Distance to the Ideal Correlation Matrix at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

our practical implementation, as well as the ones of the correlation and variance matrices corresponding to those covariance matrices. The results are plotted in figure 10. A clear sudden sharp rise in the condition number of the sample covariance matrix can be observed at the beginning of the COVID-19 crisis, when lock-downs and other measures that slowed down economies worldwide were enacted. This rise in the condition number of the sample covariance matrix was initially driven by a steep rise in correlations, as the condition number of the correlation matrix and figure 9 show. However, as the variances of returns across the assets in our portfolio universe rose, they also got to similar levels, which in turn led to a sharp drop in the condition number of the variance, and hence also of the covariance, matrix. Economically speaking, as the COVID-19 crisis started, investors initially decided to liquidate their holdings across all assets classes, and increase their cash allocation. This led to a reduction in the relatively wide difference in returns' variance between more conservative asset classes, like fixed-income, and more aggressive asset classes, like equities and commodities, that prevails during normal times. In second half of the year, correlations remained high, and variances returned to more normal levels, leading to a new increase in the condition number of the sample covariance matrix to even higher levels than in the first half of the year, as the advantageous (for the condition number) effects of higher, more similar variances across the assets disappeared.

The effects of the changes in the condition number of the sample covariance matrix can be observed in figure 8, where the changes in concentration track the changes in the condition number of the sample covariance matrix, and in figure 11, where the turnover of the MMVP is almost twice as high as the turnover of the IVP[5].

We conclude that even though the condition number of the covariance matrices used for the computation of Markowitz's minimum volatility portfolios in our practical implementation is relatively low to produce high residual-based forward error bounds, the stability of the portfolio allocation over time suffered from sudden sharp rises in corre-

[5]The inverse variance portfolio (IVP) is actually Markowitz's minimum volatility portfolio (MMVP) computed using the ideal correlation matrix. See remark 2.10.

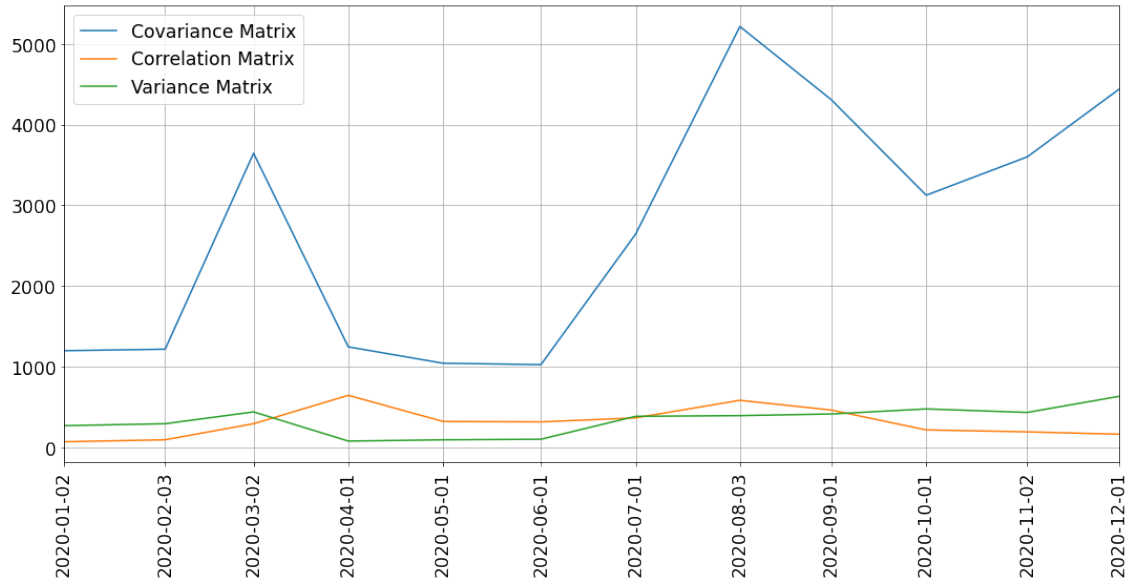

Figure 10: Condition Number of the 20 Assets Covariance and its Corresponding Correlation and Variance Matrices at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

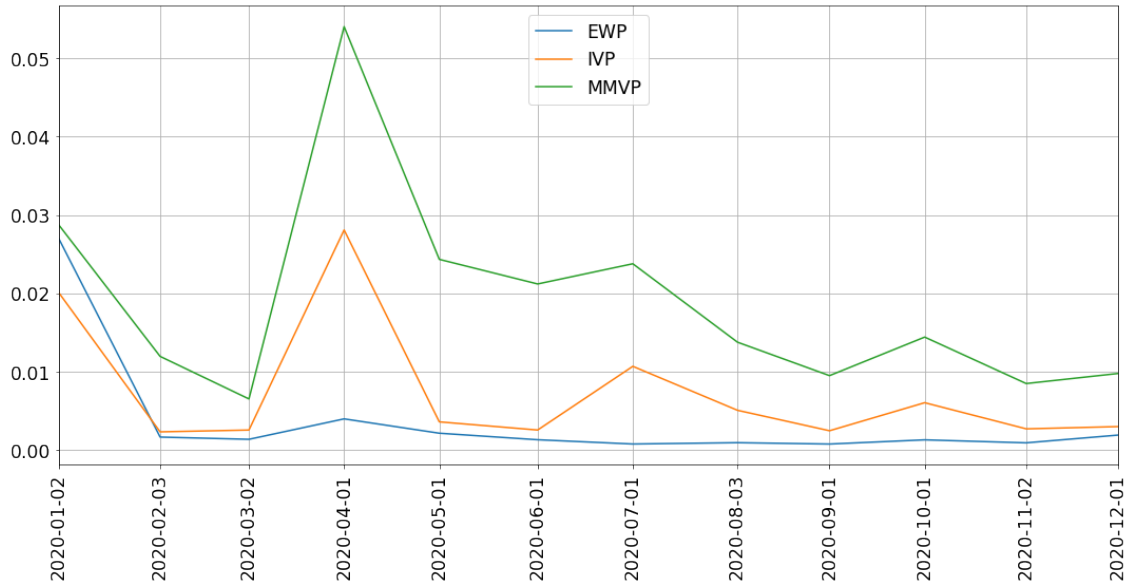

Figure 11: Turnover of the Equal Weight, Inverse Variance and Markowitz Minimum Volatility Portfolios at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

lations between assets, as well as from sudden sharp changes in the relative difference between the variance of returns of rather *safe* assets like fixed-income ETFs and "riskier" assets like equity ETFs. The low condition numbers might also be partly explained by the fact that our set of investable assets is so small, as we have shown in the simulations at the end of section 2.2.1. Another interesting finding is the fact that the stronger rise in returns' variance of "safer" assets, which exhibit lower variances in normal times, than that of "riskier" assets had a cushion effect, sinking the condition number of the covariance matrix at the start of the COVID-19 crisis and reducing portfolio turnover in the last half of the year. Finally, Markowitz's minimum volatility portfolios fail to perform better than the two naive portfolio allocation models defined above, producing higher portfolio turnover and even negative risk-adjusted returns.

3 Improving Markowitz's Portfolio Optimisation Model

In this section, we present two methods that try to improve the stability of Markowitz's minimum volatility portfolios. The first method, called covariance matrix shrinking, tries to tackle instability by reducing noise in the sample covariance matrix: if the input covariance matrix does not change dramatically across time, then the resulting optimal portfolio won't change dramatically across time either. The second method, called hierarchical risk parity, is an intermediate optimisation method between Markowitz's minimum variance portfolio and the inverse variance portfolio, and was developed by López de Prado: instead of considering stochastic dependency between the returns of different assets via their covariances directly, assets are first classified in clusters according to their correlation, and then wealth is allocated across the mentioned clusters so that they contribute to the total portfolio risk in equal proportions.

3.1 Covariance Matrix Shrinking

In Markowitz's minimum volatility portfolio construction method, the only input variable that we have is the covariance matrix of asset returns. Hence, the covariance matrix is the only source of instability and errors in the process of portfolio optimisation for this method. We have already shown this in our practical implementation in section 2.2.1.

In section 2.4, we have used the sample covariance matrix computed using NumPy's `.cov()` method applied to the matrix containing the time-series of past returns of each of the N investable assets. This method computes the unbiased sample covariance matrix of the past returns' time-series. Hence, the most important parameter for the covariance matrix estimation is the number of past periods returns in our time-series.

In order to get a covariance matrix with full-rank, we need more past returns' observations than investable assets (even though this does not guarantee invertibility). This is not a problem in our practical implementation, because we consider a relatively small set of investable assets. Nevertheless, one can think of asset managers constructing portfolios using a way bigger set of investable assets, for example equal to the S&P 500 index, that contains almost 500 companies, or the Norges Bank Investment Management, which manages a portfolio with holdings in over 9000 companies, as we mentioned before. For them, having enough past return's observations is not necessarily a simple condition to fulfil for all investable assets. Table 3 summarises the advantages and disadvantages of choosing a low and a high number of past returns' observations to estimate the covariance matrix of asset returns irrespective of the value of N.

	Low T	**High** T
Advantages	React to market changes quicker/sooner.	More stable portfolios, lower turnover.
Disadvantages	Less stable portfolios, higher turnover	React to market changes slower/later.

Table 3: T Value Trade-off

On the one hand, we can choose a low number of past return's observations to construct our covariance matrix. Doing this will result in covariance matrices that are more volatile, i.e. do change a lot across time. The estimation will be more sensitive to changes in market regimes and changes in asset returns' relationships, but also to noise, which will lead to more unstable portfolios and to higher portfolio turnover across time. On the other hand, we can also choose a high number of past returns' observations to construct our covariance matrix. This will help to get covariance matrices that are more stable

across time, producing optimal portfolios that are also more stable across time and hence reducing turnover. However, this portfolios will also be more inelastic to changes in market regimes and changes in asset returns' relationships. As a practical example, one can think about the changes in performance across the different equity sectors during a business cycle: some sectors like consumer discretionary (e.g. companies selling luxury goods or travel) tend to perform better than other sectors during periods of low unemployment and high growth, because people tend to be in a better financial situation to afford that kind of purchases, meanwhile other sectors like consumer staples (e.g. companies selling basic products like rice, bread, toilet paper, tooth paste, etc.) tend to perform better than other sectors during periods of high unemployment and low growth, when people cut off unnecessary spending and these companies profit from the fact that they sell products that are necessary for our daily life. Investors would like to get optimal portfolios that react to those changes in business cycles in order to profit from the out-performance of some asset classes and sectors, while protecting their portfolios from the under-performance of some other asset classes and sectors. Moreover, optimal portfolios constructed on the noise abundant in financial time-series should be avoided, so as to reduce turnover at each rebalancing point.

Optimising this trade-off is the objective of the covariance matrix shrinkage methods, even though they do not optimise the value of the parameter T directly.

3.1.1 Ledoit and Wolf

Ledoit and Wolf propose using a convex combination of the sample covariance matrix and a structured estimator, i.e. some matrix containing little noise and which is estimated with high bias. The idea is to reduce the noise contained in the sample covariance matrix by compensating positive and negative error that tends to be contained in its highest entries in absolute value. These high-absolute-valued entries of the covariance matrix tend to be high-absolute-valued mostly because of noise and not because the covariance between the corresponding asset return's time-series is truly high, as argued in [15] (page 2).

Definition 3.1 (Shrunk Covariance Matrix. Page 6, [15])**.** *Let $\Sigma \in \mathbb{R}^{N\times N}$ be a sample covariance matrix and $F \in \mathbb{R}^{N\times N}$ some other matrix, also called* ***shrinkage target****. The* ***shrunk covariance matrix*** *is defined as*

$$\Sigma_{shrunk} := \delta F + (1-\delta)\Sigma,$$

where $\delta \in [0,1]$ is called ***shrinkage constant****.*

In definition 3.1, we have 2 parameters apart from the sample covariance matrix Σ that we need to further specify: the shrinkage target and the shrinkage constant.

For the first, the shrinkage target, we need some estimator that can be computed using few parameters, i.e. a structured estimator, and that also contains important information about the measure that we are trying to estimate, i.e. the stochastic dependency between the random variables of the asset returns. In [14] (page 5), the authors propose using the single-factor matrix of Sharpe as the shrinkage target, which is a single-index asset pricing model (see [21] and [14] page 5 for more details). However, in [15] (page 6), the same authors propose a simpler estimator as shrinkage target.

Definition 3.2 (Constant Correlation Model. Page 12, [15])**.** *Let $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{N\times N}$ be the sample covariance matrix, then the* ***sample constant correlation matrix*** *$F_\Sigma = [f_{i,j}] \in \mathbb{R}^{N\times N}$* ***corresponding to the sample covariance matrix*** *Σ is defined as*

$$f_{i,j} := \begin{cases} \sigma_{i,i}, & \text{for } i = j, \\ \bar{r}\sqrt{\sigma_{i,i}\sigma_{j,j}}, & \text{for } i \neq j, \end{cases}$$

where

$$\bar{r} := \frac{2}{(N-1)N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\sigma_{i,j}}{\sqrt{\sigma_{i,i}\sigma_{j,j}}} \tag{3}$$

is the average sample correlation.

Ledoit and Wolf argue in [15] (page 6) that the constant correlation model gives comparable performance while being easier to implement than the single-factor matrix model of Sharpe. However, they also mention that the constant correlation model is not suited for portfolio construction if the set of investable assets contains assets from different classes, for example fixed-income and equities, because it assumes that all pairwise correlations are treated equal for the computation of the average sample correlation r. From an economic perspective, this makes sense because the returns of different asset classes can be driven by very different factors. For example, while the stock returns of technology companies are often driven by their margins and customer base growth, the bond returns of the same companies are rather driven by the liquidity of their balance sheet (that is, how much cash is and will be available to the pay the coupons and pay the principal back). Nevertheless, we consider this shrinkage target in our practical implementation, because our backtesting period is marked by the market-stress caused by the COVID-19 crisis, and the asset returns across the different asset classes during this kind of periods tend to be driven by macroeconomic factors, like lockdown measures, that affect all of them relatively equally.

The second parameter in definition 3.1, the shrinkage constant, represents the compromise between the sample covariance matrix and the structured estimator. It can take infinite many different values, even if constrained to be in the interval $[0, 1]$ so as to compute the shrunk covariance matrix as a convex combination of the structured estimator and the sample covariance matrix. Ledoit and Wolf propose defining the shrinkage constant as the optimal $\delta^* \in [0, 1]$ that minimises the distance between the shrunk covariance matrix and the true covariance matrix, i.e. the covariance matrix that contains the covariances of investable asset returns without noise.

Theorem 3.3 (Page 7, [14])**.** *Let $S = [s_{i,j}] \in \mathbb{R}^{N\times N}$ be a sample covariance matrix computed using $T \in \mathbb{N}$ past return's observations, $F_S = [f_{i,j}] \in \mathbb{R}^{N\times N}$ the sample constant correlation matrix corresponding to S, $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{N\times N}$ the true covariance matrix and $F_\Sigma = [\phi_{i,j}] \in \mathbb{R}^{N\times N}$ its constant correlation matrix. Moreover, we define the loss function*

$$L(\delta) := \|\delta F_S + (1-\delta)S - \Sigma\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm. The shrinkage constant δ^* that minimises the expected value of L is given by

$$\delta^* = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{V}[s_{i,j}] - Cov[f_{i,j}, s_{i,j}]}{\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{V}[f_{i,j} - s_{i,j}] + (\phi_{i,j} - \sigma_{i,j})^2}, \tag{4}$$

where the function $\mathbb{V}[\cdot]$ designates the variance and $Cov[\cdot,\cdot]$ the covariance.

The proof of this theorem is provided in [14] (page 8) and goes as follows.

Proof. The expected value of the loss function L can be rewritten as

$$\begin{aligned}\mathbb{E}[L(\delta)] &= \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}[\delta f_{i,j} + (1-\delta)s_{i,j} - \sigma_{i.j}]^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[\delta f_{i,j} + (1-\delta)s_{i,j}] + (\mathbb{E}[\delta f_{i,j} + (1-\delta)s_{i,j} - \sigma_{i,j}])^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\delta^2\mathbb{V}[f_{i,j}] + (1-\delta)^2\mathbb{V}[s_{i,j}] + 2\delta(1-\delta)Cov[f_{i,j}, s_{i,j}] + \delta^2(\phi_{i,j} - \sigma_{i,j})^2.\end{aligned}$$

The first and second derivatives of the expected value of the loss function L with respect to δ are given by

$$\begin{aligned}\frac{\partial\mathbb{E}[L]}{\partial\delta} &= 2\sum_{i=1}^{N}\sum_{j=1}^{N}\delta\mathbb{V}[f_{i,j}] - (1-\delta)\mathbb{V}[s_{i,j}] + (1-2\delta)Cov[f_{i,j}, s_{i,j}] + \delta(\phi_{i,j} - \sigma_{i,j})^2, \\ \frac{\partial^2\mathbb{E}[L]}{\partial\delta^2} &= 2\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[f_{i,j} - s_{i,j}] + (\phi_{i,j} - \sigma_{i,j})^2.\end{aligned}$$

Because $\frac{\partial^2\mathbb{E}[L]}{\partial\delta^2} > 0$ holds, the minimum of the expected value of the loss function L can be computed by setting $\frac{\partial\mathbb{E}[L]}{\partial\delta}$ equal to zero. Solving for δ^* we get

$$\delta^* = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[s_{i,j}] - Cov[f_{i,j}, s_{i,j}]}{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[f_{i,j} - s_{i,j}] + (\phi_{i,j} - \sigma_{i,j})^2}.$$

□

The only problem of the optimal shrinkage constant presented in theorem 3.3 is that we do not know the true covariance matrix and can only compute sample variances and covariances. Therefore, Ledoit and Wolf propose using a consistent estimator for δ^*, i.e. an estimator that converges to the true δ^* for $T \to \infty$.

Theorem 3.4 (Pages 12-15, [15])**.** *Assuming that asset returns' random variables a_i are independent and identically distributed with finite fourth moment, i.e. $\mathbb{E}[|a_i^4|] < \infty$ for $i = 1, \ldots, N$. Let $Y = [y_{i,j}] \in \mathbb{R}^{N\times T}$ the matrix with the past T periods returns, and $\bar{y}_i := \sum_{j=1}^{T} y_{i,j}$. Moreover, we define the matrices $S = [s_{i,j}] \in \mathbb{R}^{N\times N}$ as the sample covariance matrix of asset returns, $F_S = [f_{i,j}] \in \mathbb{R}^{N\times N}$ as its sample constant correlation matrix, $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{N\times N}$ as the true covariance matrix and $F_\Sigma = [\phi_{i,j}] \in \mathbb{R}^{N\times N}$ as its constant correlation matrix. Then the following holds.*

1. *A consistent estimator for $\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[\sqrt{(T)}s_{i,j}]$ is given by*

$$\hat{\pi} := \sum_{i=1}^{N}\sum_{j=1}^{N}\hat{\pi}_{i,j} \quad \textit{where} \quad \hat{\pi}_{i,j} := \frac{1}{T}\sum_{t=1}^{T}\left((y_{i,t} - \bar{y}_i)(y_{j,t} - \bar{y}_j)\right)^2.$$

2. *A consistent estimator for $\sum_{i=1}^{N}\sum_{j=1}^{N}Cov[\sqrt{T}f_{i,j}, \sqrt{T}, s_{i,j}]$ is given by*

$$\hat{\rho} := \sum_{i=1}^{N}\hat{\pi}_{i,i} + \sum_{i=1}^{N}\sum_{j=1,\, j\neq i}^{N}\frac{\bar{r}}{2}\left(\sqrt{\frac{s_{j,j}}{s_{i,i}}}\hat{\vartheta}_{i,i,j} + \sqrt{\frac{s_{i,i}}{s_{j,j}}}\hat{\vartheta}_{j,i,j}\right),$$

$$\textit{where} \quad \hat{\vartheta}_{k,i,j} := \frac{1}{T}\sum_{t=1}^{T}\left((y_{k,t} - \bar{y}_k)^2 - s_{k,k}\right)\left((y_{i,t} - \bar{y}_i)(y_{j,t} - \bar{y}_j) - s_{i,j}\right),$$

where $\bar{r}$ is defined as in equation 3.

3. *A consistent estimator for $\sum_{i=1}^{N}\sum_{j=1}^{N}(\phi_{i,j}-\sigma_{i,j})^2$ is given by*

$$\hat{\gamma} := \sum_{i=1}^{N}\sum_{j=1}^{N}(f_{i,j}-s_{i,j})^2.$$

4. *$\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{V}[f_{i,j}-s_{i,j}] = O\left(\frac{1}{T}\right)$ holds.*

5. *A consistent estimator for the optimal δ^* defined in equation 4 is given by*

$$\hat{\delta}^* = \frac{1}{T}\frac{\hat{\pi}-\hat{\rho}}{\hat{\gamma}}. \tag{5}$$

A proof of this theorem cannot be found in [15] but is given in [14] (pages 10-11). The assumption of asset returns' random variables being independent and identically distributed with finite fourth moments is necessary to prove convergence using the Central Limit Theorem. Ledoit and Wolf argue in [14] (page 4) that stock returns do not fulfil this assumption, but that covariance matrix estimators also use this assumption and that introducing extra degrees of freedom in the estimation process to account for dependence and conditional heteroskedastic does not necessarily lead to better out-of-sample performance. However, we do not want to get into those details, because proving or disproving them is very hard and often relies on empirical statistical tests. Hence, we will focus on the empirical results of Ledoit and Wolf's method in our practical implementation backtest, and compare them to the other models according to the risk-adjusted returns, nHHI and turnover measures defined at the beginning of section 2.3.

In [15] (page 15), Ledoit and Wolf point to the fact that $\hat{\delta}^* \notin [0,1]$ in equation 5 may hold. Hence, the final shrinkage intensity proposed by Ledoit and Wolf is defined as

$$\delta := \max\left\{0, \min\left\{\frac{1}{T}\frac{\hat{\pi}-\hat{\rho}}{\hat{\gamma}}, 1\right\}\right\}. \tag{6}$$

Algorithm 2: Markowitz Minimum Variance Portfolio Computation with Ledoit and Wolf's Covariance Matrix Shrinkage

Input : N-Asset return matrix $R \in \mathbb{R}^{N\times T}$
Output: Markowitz Minimum Variance Portfolio w_{opt} corresponding to Ledoit and Wolf's shrunk covariance matrix

Compute the sample covariance matrix $\Sigma \in \mathbb{R}^{N\times N}$;
Compute the sample constant correlation matrix $F_\Sigma \in \mathbb{R}^{N\times N}$;
Compute the shrinkage intensity $\delta = \max\left\{0, \min\left\{\frac{1}{T}\frac{\hat{\pi}-\hat{\rho}}{\hat{\gamma}}, 1\right\}\right\} \in [0,1]$;
Set $\Sigma_{shrunk} = \delta F_\Sigma + (1-\delta)\Sigma$;
if Σ_{shrunk} *is non-singular* **then**
    Set $\tilde{w}_{opt} \in \mathbb{R}^N$ to the solution to $\Sigma_{shrunk} w = \mathbb{1}$;
end
else
    Set $\tilde{w}_{opt} \in \mathbb{R}^N$ to one of the solutions to the least-squares problem $\Sigma_{shrunk} w = \mathbb{1}$;
end
Set $w_{opt} = \frac{1}{\mathbb{1}^T \tilde{w}_{opt}} \tilde{w}_{opt}$;

Now that we have introduced the topic of covariance matrix shrinking and defined the new algorithm to use the shrunk covariance matrix for the computation of Markowitz's minimum volatility portfolios, we can describe a more sophisticated approach to covariance shrinkage.

3.1.2 De Nard

In subsection 3.1.1, we have discussed a covariance matrix shrinkage method that uses the sample constant correlation matrix corresponding to the sample covariance matrix as shrinkage target. As already mentioned before, the disadvantage of this shrinkage target is that all pairwise correlations are weighted equally. However, the pairwise unconditional correlations of assets from different classes (for example of fixed-income vs equity assets) or even of assets from the same class but from different industries or regions are not equal, because the asset returns of different classes are driven by different economic factors.

In [7], De Nard proposed a different approach for the covariance shrinkage method, that first clusters the assets in the covariance matrix in homogeneous groups to then shrink the corresponding block matrices inside the covariance matrix separately, instead of using the "all-assets-are-equal" approach proposed by Ledoit and Wolf. This approach allows us to apply covariance shrinkage to portfolios composed of assets from different classes (as in our practical implementation) and also account for the different fundamental properties of assets from different industries and regions in the case of portfolios that are homogeneous with respect to asset class.

Definition 3.5 (Generalised Constant-Variance-Covariance Shrinkage Target. Page 13, [7])**.** *Let $K \in \mathbb{N}$ be the number of homogeneous groups in which we cluster the $N \in \mathbb{N}$ assets, such that $K \leq N$. Without loss of generality, assume that the covariance matrix of asset returns is given by*

$$\Sigma = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \dots & \Sigma_{1,K} \\ \Sigma_{2,1} & \Sigma_{2,2} & \dots & \Sigma_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{K,1} & \Sigma_{K,2} & \dots & \Sigma_{K,K} \end{bmatrix} \in \mathbb{R}^{N \times N},$$

where $\Sigma_{i,j} = \left[\sigma_{l,m}^{i,j}\right] \in \mathbb{R}^{N_i, N_j}$ is the covariance matrix between the assets in the i-th and j-th homogeneous groups.

*We define the **generalised constant-variance-covariance (GCVC) shrinkage target** as*

$$\Phi_{GCVC} = \begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} & \dots & \Phi_{1,K} \\ \Phi_{2,1} & \Phi_{2,2} & \dots & \Phi_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{K,1} & \Phi_{K,2} & \dots & \Phi_{K,K} \end{bmatrix} \in \mathbb{R}^{N \times N},$$

where $\Phi_{i,j} \in \mathbb{R}^{N_i, N_j}$ is the (i,j)-th cluster of Φ_{GCVC} and is defined as

$$\Phi_{i,j} := \begin{cases} \varphi_k I_{N_k} + \nu_k \left(\mathbb{1}_{N_k} \mathbb{1}_{N_k}^T - I_{N_k} \right), & \text{for } i = j = k \in \{1, \dots, K\}, \\ \nu_{i,j} \mathbb{1}_{N_i} \mathbb{1}_{N_j}^T, & \text{for } i \neq j \in \{1, \dots, K\}, \end{cases}$$

where N_k is the number of assets in group k, I_{N_k} is the N_k-dimensional identity matrix and $\mathbb{1}_{N_k}$ is the N_k-dimensional vector of ones for all $k \in \{1, \dots, K\}$. Moreover, $\varphi_k, \nu_k, \nu_{i,j}$

are the average variance of group k, the average covariance of group k and the average covariance of the off-diagonal (i,j)-th cluster respectively, i.e.

$$\varphi_k := \frac{1}{N_k}\sum_{l=1}^{N_k}\sigma_{l,l}^{k,k},$$
$$\nu_k := \frac{1}{N_k^2}\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}\sigma_{l,m}^{k,k},$$
$$\nu_{i,j} := \frac{1}{N_i N_j}\sum_{l=1}^{N_i}\sum_{m=1}^{N_j}\sigma_{l,m}^{i,j}.$$

Obviously, $\nu_{i,j} = \nu_{j,i}$ and hence $\Phi_{i,j} = \Phi_{j,i}$ for all $i,j = 1,\dots,K$.

De Nard's approach is to use the generalised constant-variance-covariance shrinkage target and the consistent estimator of the optimal shrinkage constant defined in equation 6, which in this case is computed using the matrix Φ_{GCVC} instead of the constant correlation matrix.

The most important parts of the computation of the GCVC shrinkage target are choosing the parameter K, i.e. deciding how many groups we want to cluster our N assets into, and also deciding how to cluster the N assets into K (relatively) homogeneous groups. The answer to the first question can be given intuitively, by considering as many groups as assets classes, regions or industries as there are represented in our N assets. De Nard also describes a clustering method in [7] (pages 16-19) that does not use K as input parameter. However, we will focus on the *Blockbuster Shrinkage Estimator* that clusters the N assets into a previously determined number of groups K using the Blockbuster clustering algorithm, because the number of assets considered in our practical implementation is rather low and we would like to group them by asset class.

Before we define the Blockbuster Clustering Algorithm, we need to define the K-means algorithm, which is an unsupervised learning algorithm that clusters a set of data points $x_1,\dots,x_L \in \mathbb{R}^N$, $L \in \mathbb{N}$ into a preset number $K \in \mathbb{N}$ of groups. This algorithm is responsible for the grouping step in the Blockbuster Clustering Algorithm, after the input data is preprocessed.

Algorithm 3: K-means

Input : Data points $X = \{x_1,\dots,x_L\} \subset \mathbb{R}^N$, $L \in \mathbb{N}$, number of groups $K \in \mathbb{N}$ and tolerance $tol \in \mathbb{R}_{\geq 0}$

Output: Groups $M_1,\dots,M_K \subset \{x_1,\dots,x_L\}$, such that $M_i \cap M_j = \emptyset$ for all $i \neq j$

Select K random points from X, denote them by $x_{i_1},\dots,x_{i_K}$;
Initialise $M_j = \{x_{i_j}\}$ and set $\mu_j = x_{i_j}$ and $\tilde{\mu}_j = 0$ for $j = 1,\dots,K$; **while** $\|\mu_j - \tilde{\mu}_j\| \geq tol$ *for some* $j = 1,\dots,K$ **do**
  foreach $j = 1,\dots,K$ **do**
    Set $\tilde{\mu}_j = \mu_j$;
    Set $M_j = \{x_l \in X : \mu_j = \arg\min_{j=1,\dots,K} \|\mu_j - x_l\|\}$;
    Set $\mu_j = \frac{1}{|M_j|}\sum_{x\in M_j} x$;
  end
end

The K-means algorithm has as objective the minimisation of within-cluster sum-of-

squares, i.e. the minimisation of

$$\sum_{l=0}^{L} \min_{\mu_j} \|x_l - \mu_j\|^2,$$

where μ_j is the sample mean of the data points in the group M_j for $j = 1, \ldots, K$.

Because the algorithm is initialised with random centroids, i.e. random points that act as initial sample mean of each group, the algorithm only guarantees convergence to a local minimum, but not necessarily to a global minimum of the sum above. The algorithm is hence sensitive to the initialisation step. Moreover, given a relatively small sample ($|X|$ is *small*), the algorithm is very sensitive to data points that lie *far away* for the others, i.e. the algorithm is very sensitive to outliers. Some other drawbacks from the algorithm are depicted in figure 12, sourced from scikit-learn's website.

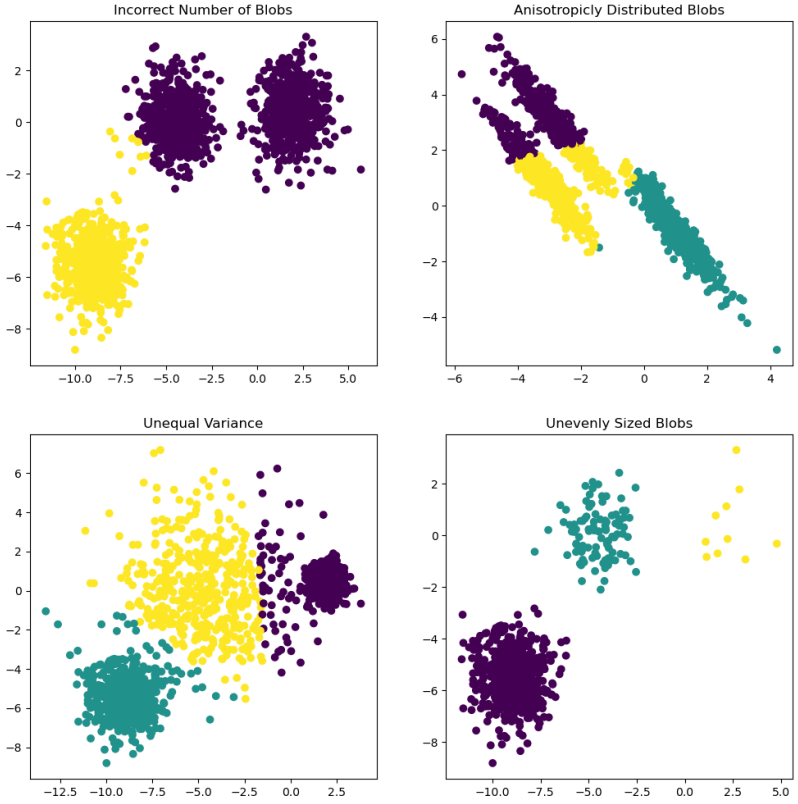

Figure 12: Visualisation of Some Drawbacks of the K-means Clustering Algorithm.

Algorithm 4: Blockbuster Clustering Algorithm. Page 16, [7]

Input : N-Asset return matrix $R \in \mathbb{R}^{N \times T}$ and number of groups $K \in \mathbb{N}$
Output: K-means partition $\hat{\Pi}^K = \{\hat{\Pi}_1, \ldots, \hat{\Pi}_K\}$ as the estimate of the asset groups

1 Compute the sample covariance matrix $S \in \mathbb{R}^{N \times N}$;
2 Construct the sample eigenvector matrix $\tilde{U} \in \mathbb{R}^{N \times K}$ such that its columns are the eigenvectors corresponding to the K largest sample eigenvalues;
3 Standardise each row of $\tilde{U}$ by its Euclidean norm and denote the row-normalised sample eigenvector matrix by X, so that $x_{n,k} = \tilde{u}_{n,k}/\|\tilde{u}_n\|$;
4 Apply the K-means algorithm to the rows of X;

The objective of this algorithm is to cluster the N assets into K groups such that the correlations between the assets in a given group is higher than the correlation between assets of different groups. The result is a partition into K (relatively) homogeneous sets. We use these sets to reorder the rows and columns of the sample covariance matrix in order to have the assets within a given group in contiguous rows and columns, and for the sample covariance matrix to have a block structure as the GCVC shrinkage target in definition 3.5.

After the theoretical introduction to Ledoit and Wolf's and De Nard's covariance shrinkage methods, we proceed to the practical implementation results of the two methods.

3.1.3 Practical Implementation

We implemented both covariance shrinkage methods and simulated the results of the Markowitz's minimum volatility portfolios computed using the shrunk covariance matrix as input. For De Nard's method, we used a fixed amount of groups, which the assets in our portfolio should be clustered into before computing the shrinkage target: 4 groups, because we have 4 different asset classes (see appendix A). Hence, the suffix "K4" to the abbreviated name of De Nard's method that we will use in this section. The computed portfolios as well as the clusters can be found in appendix E.

Because covariance shrinkage methods try to improve stability by modifying the input matrix used for the computation of the corresponding Markowitz's minimum volatility portfolio, we first have a look at the condition number of the input matrices used in the computation for Ledoit and Wolf's and De Nard's methods, and compare them to the condition number of the sample covariance matrices used for the computation of the portfolios in section 2.4. In figure 13, the condition number with respect to the Euclidian norm of the sample covariance matrix (MMVP), Ledoit and Wolf's shrunk covariance matrix (MMVP-LW) and De Nard's shrunk covariance matrix (MMVP-DN-K4) over time are plotted. Both covariance shrinkage methods clearly achieve to reduce the condition number of the input matrix used for the computation of Markowitz's minimum volatility portfolios, with condition numbers that consistently lie below those of the sample covariance matrix. De Nard's method does achieve a more dramatic reduction in the condition number of the input matrix, with condition numbers that remain pretty stable over time until August 2020, when a short-lived spike can be observed. However, the spike in the condition number of De Nard's August 2020 shrunk covariance matrix stays well below the spikes which are also observed for the sample covariance matrix and Ledoit and Wolf's shrunk covariance matrix in August 2020.

Figure 14 shows the distance of the correlation matrix to the ideal correlation matrix computed as in definition 2.19 for the correlation matrices corresponding to each of the 12 covariance matrices computed with each shrinkage method. Here we observe, that

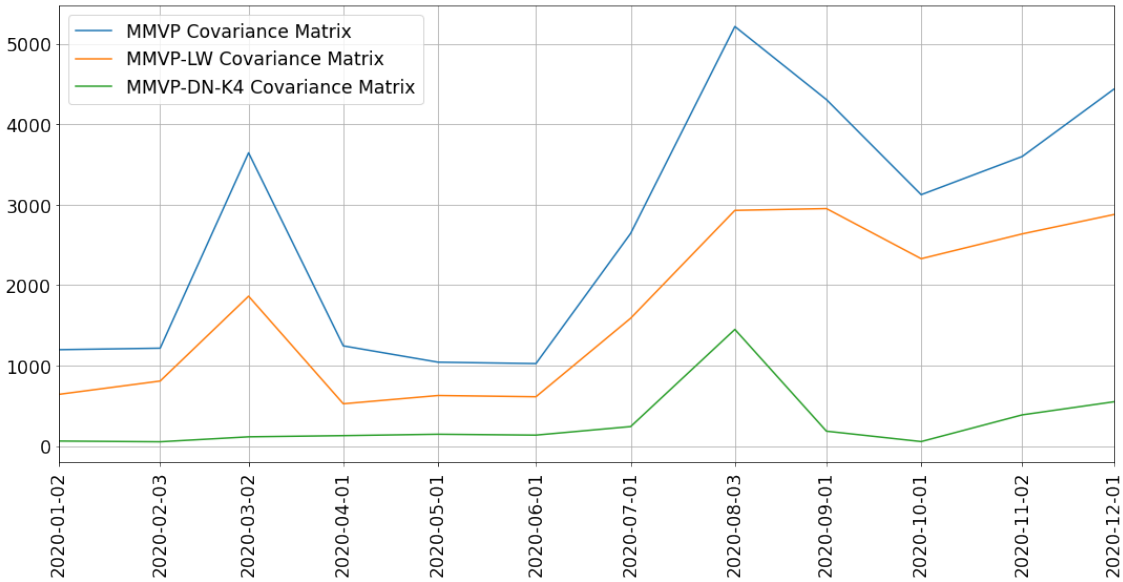

Figure 13: Condition Number of the 20 Assets Sample, Ledoit and Wolf Shrunk and De Nard $K = 4$ Shrunk Covariance Matrices at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

both shrinkage methods result in underlying correlation matrices that are nearer to the identity matrix with respect to the mentioned distance measure. It is interesting to see that Ledoit and Wolf's method reduces the distance further than De Nard's method, even though the condition number of the shrunk covariance matrices of the first method are higher than in the case of the second method. This lets us think that the low condition numbers of De Nard's shrunk covariance matrices are caused by a stronger reduction in the ratio of highest to lowest variance (or standard deviation) than in the case of Ledoit and Wolf's method. This makes sense, because Ledoit and Wolf's constant correlation model focuses on shrinking the covariance matrix by pulling together off diagonal elements of the covariance matrix, mathematically this can be verified by the fact that the shrinkage target matrix' diagonal elements are equal to those of the sample covariance matrix (the variances of asset returns), see definitions 2.4 and 3.2. On the other hand, De Nard's method uses an average of the variances of each group in the definition of the diagonal entries of the shrinkage target, see definition 3.5.

Figure 15 shows the evolution of target allocation across time of each asset in our portfolio for Markowitz's minimum volatility portfolios computed using Ledoit and Wolf's covariance shrinkage method described in section 3.1.1 (MMVP-LW). The plot shows lower variance in allocation than we saw in figure 7 for Markowitz's minimum volatility portfolios computed with the sample covariance matrix as input, but higher variance than for inverse variance portfolios in plot 5.

In figure 16, we see the same plot as in figure 15 but for Markowitz's minimum volatility portfolios generated using De Nard's covariance shrinkage method described in section 3.1.2 (MMVP-DN-K4). The changes in allocation to each asset in the portfolio over time are clearly more pronounced than for any of the other portfolio allocation methods analysed before. This is a rather unexpected result, because covariance shrinkage methods try to tackle the very source of instability in portfolio allocation over time, and the condition numbers of the shrunk covariance matrices were shown to be well below those of the

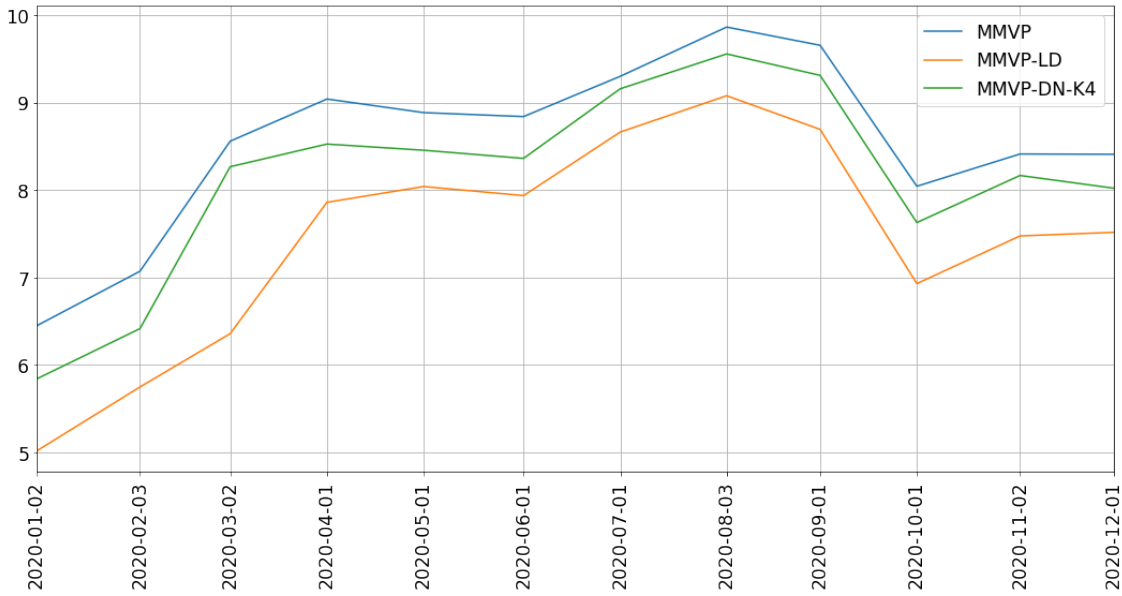

Figure 14: 20 Assets Correlation Matrix Distance to the Ideal Correlation Matrix at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020 for each of the MMVP Methods.

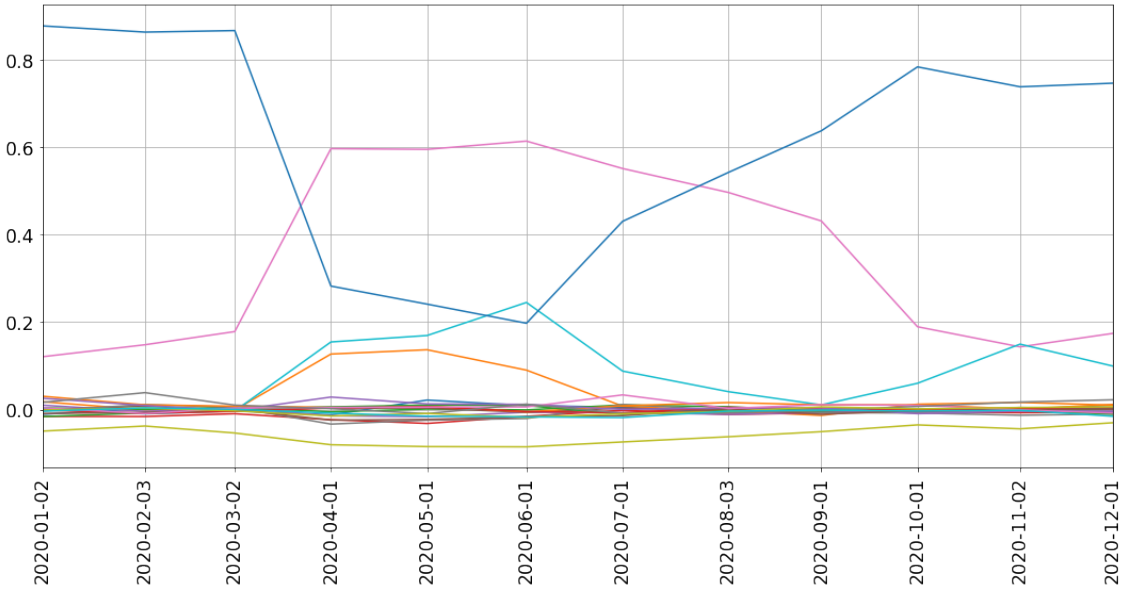

Figure 15: MMVP-LW of the Practical Implementation Portfolio Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

sample and Ledoit and Wolf's shrunk covariance matrices. However, when comparing their target allocation over time plots, De Nard's method does not show any improvement compared to the original Markowitz's minimum volatility portfolios computed with the sample covariance matrix as input (see figure 7), and is clearly the most unstable method among the ones analysed before.

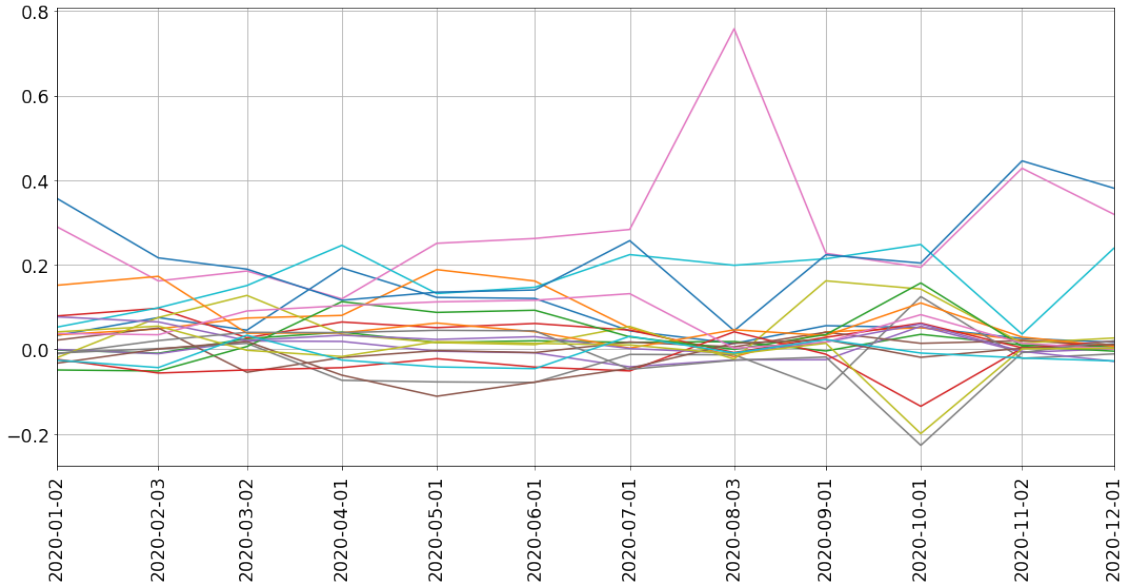

Figure 16: MMVP-DN-K4 of the Practical Implementation Portfolio Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

In table 4, we have summarised the risk-adjusted returns measures and turnover figures of all portfolio allocation methods analysed until now. Ledoit and Wolf's covariance shrinkage method (MMVP-LW) achieves a significant reduction in turnover while improving the Sharpe ratio compared to Markowitz's minimum volatility portfolios computed without shrinkage (MMVP). The Sharpe ratio of MMVP-LW is even positive and higher than Sharpe ratio of equal weight portfolios (EWP), but still lower than the Sharpe ratio of inverse variance portfolios (IVP). On the contrary, De Nard's covariance shrinkage method (MMVP-DN-K4) fails to improve the Sharpe ratio of Markowitz's minimum volatility portfolios, and even produces an important increase in portfolio turnover, which might have contributed to the bad risk-adjusted returns generated because of increased transaction costs.

Portfolio Optimisation Method	**Sharpe Ratio**	**Turnover**
EWP	0.36	0.003634
IVP	1.008	0.007396
MMVP	-0.017	0.018856
MMVP-LW	0.5	0.012527
MMVP-DN-K4	-0.134	0.027798

Table 4: Sharpe Ratio and Turnover Figures of the Equal Weight, Inverse Variance Portfolios, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit and Wolf's Shrinkage and Markowitz Minimum Volatility after De Nard's Shrinkage Portfolios for the Backtesting Period from 01/01/2020 to 31/12/2020.

In figure 17, we can see the portfolio turnover at each rebalancing point for all portfolio allocation models. We have already seen in table 4, that Ledoit and Wolf's method does indeed lead to a lower portfolio turnover compared to Markowitz's minimum volatility portfolios computed using the sample covariance matrix. In figure 17, this observation is

also confirmed for all rebalancing points, where Ledoit and Wolf's covariance shrinkage method (MMVP-LW) results in turnover figures that are consistently lower than those of the classical Markowitz's minimum volatility portfolios (MMVP) and higher than those of the inverse variance portfolio allocation model. On the contrary, De Nard's method fails to produce lower portfolio turnover at almost all rebalancing points compared to MMVP, and even results in turnover figures more than double as high as those of all other portfolio optimisation methods analysed from the August rebalancing point onwards.

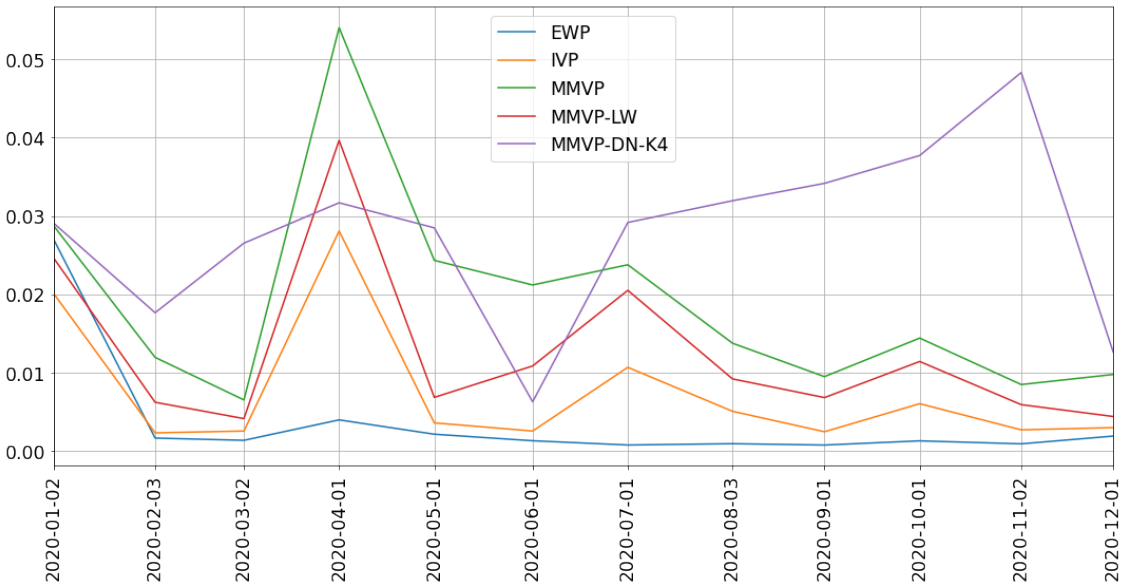

Figure 17: Turnover of Equal Weight, Inverse Variance Portfolios, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit and Wolf's Shrinkage and Markowitz Minimum Volatility after De Nard's Shrinkage Portfolios at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

Figure 18 shows the normalised Herfindahl-Hirsch-Index (nHHI) over time of the target portfolios computed by the different portfolio allocation models analysed until now. We can observe that Ledoit and Wolf's covariance shrinkage method (MMVP-LW portfolios) fail to reduce concentration with respect to nHHI compared to Markowitz's minimum volatility portfolios computed using the sample covariance matrix (MMVP). None of the MMVP-LW portfolios presents a lower nHHI than their corresponding MMVP. De Nard's covariance shrinkage method (MMVP-DN-K4) does indeed result in a significant reduction in concentration with respect to nHHI compared to MMVP, with only two exceptions: August and November portfolios.

We can conclude, that Ledoit and Wolf's covariance shrinkage method produces input matrices with lower condition numbers, and achieves better out-of-sample risk-adjusted returns and significantly lower portfolio turnover than Markowitz's minimum volatility portfolios computed using the sample covariance matrix as input. However, it fails to reduce concentration with respect to the normalised Herfindahl-Hirsch-Index. De Nard's method also produces input matrices with lower condition numbers that lead to lower concentration figures. However, it fails to reduce portfolio turnover and its out-of-sample risk-adjusted performance is the worst among all portfolio optimisation methods analysed until now. This bad results might be explained by the fact that our asset universe is small, with only 20 assets, which leads to high sensitivity to outliers in the k-means algorithm

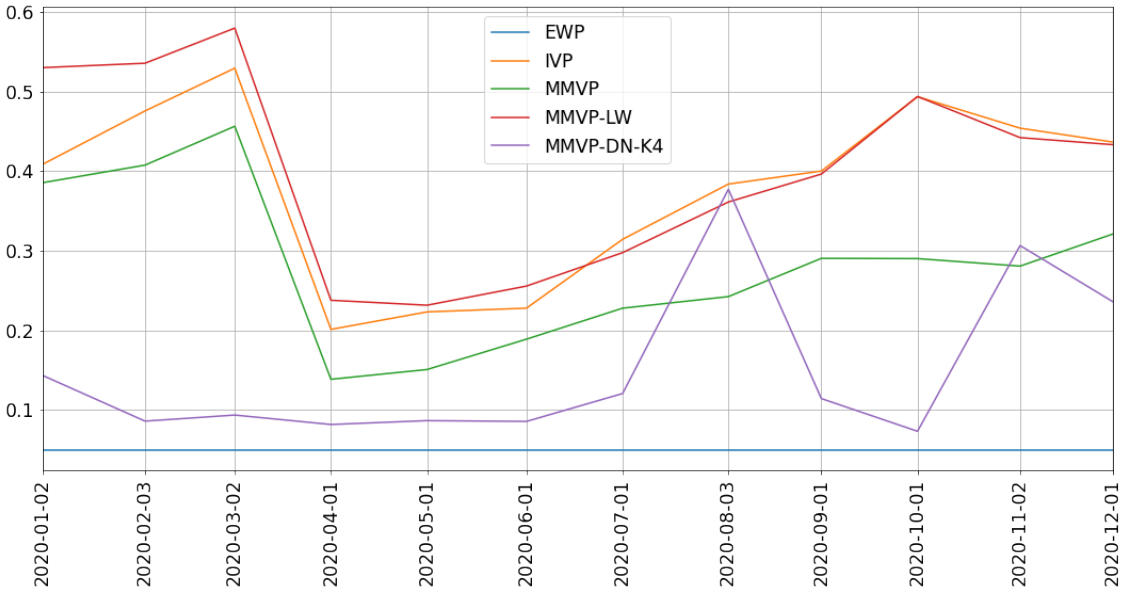

Figure 18: nHHIn of the Equal Weight, Inverse Variance, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit and Wolf's Shrinkage and Markowitz Minimum Volatility after De Nard's Shrinkage Portfolios Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

used for clustering them. Finally, it is worth to mention, that none of the methods to compute Markowitz's minimum volatility portfolios has resulted in better Sharpe ratios or turnover figures than those of the inverse variance portfolios.

3.2 Hierarchical Risk Parity

In this section, we present an alternative risk-based portfolio allocation model that lies between Markowitz minimum volatility portfolio and the inverse variance portfolio.

In [8] (page 4), López de Prado argues that the correlation matrix of asset returns is a linear algebra object that measures the cosines of the angles between any two vectors in the vector space formed by the return's time-series

Definition 3.6 (Page 181, [16]). *Let $x, y \in \mathbb{R}^N \setminus \{0\}$, then the **cosine of the angle between** x **and** y is defined as*

$$\cos\theta(x,y) := \frac{\langle x, y\rangle}{\|x\|_2\|y\|_2},$$

where $\langle\cdot,\cdot\rangle$ is the Euclidean scalar product in $\mathbb{R}^N$.

Two vectors are consider "close" when the angle $\theta(x,y)$ is "small", i.e. when $\cos\theta(x,y)$ is "close" to 1.

Theorem 3.7. *Let $R \in \mathbb{R}^{N\times T}$ be a matrix that contains the past $T \in \mathbb{N}$ periods returns of $N \in \mathbb{N}$ assets and let $P = [\rho_{i,j}] \in \mathbb{R}^{N\times N}$ be the corresponding sample correlation matrix. Then $\rho_{i,j}$ is equal to the cosine of the angle between the row vectors $\tilde{r}_i, \tilde{r}_j \in \mathbb{R}^T$ containing the centred T past period returns of asset $i, j \in \{1, \ldots, N\}$ up to some constant factor.*

Proof. Let $r_i, r_j \in \mathbb{R}^T$ be the vectors containing the past T periods returns of assets $i, j \in \{1, \dots, N\}$, then the corresponding centred vectors of past returns are given by

$$\tilde{r_k} = \begin{bmatrix} r_{k,1} - \mu_k \\ \vdots \\ r_{k,T} - \mu_k \end{bmatrix}, \quad \text{where} \quad \mu_k := \frac{1}{T} \sum_{t=1}^{T} r_{k,t} \quad \text{for} \quad k = i, j.$$

See definition 2.4.

Using $\tilde{r_i}, \tilde{r_j}$, we can compute the unbiased sample variance σ_i^2, σ_j^2 of assets i, j and their unbiased covariance $\sigma_{i,j}$.

$$\sigma_k := \frac{1}{T-1} \sum_{t=1}^{T} (r_{k,t} - \mu_k)^2 = \frac{1}{T-1} (\tilde{r_k}^T \tilde{r_k}) \qquad \text{for} \quad k = i, j,$$

$$\sigma_{i,j} := \frac{1}{T-1} \sum_{t=1}^{T} (r_{i,t} - \mu_i)(r_{j,t} - \mu_j) = \frac{1}{T-1} (\tilde{r_i}^T \tilde{r_j}).$$

Knowing the sample variances and sample covariance, we can compute the sample correlation $\rho_{i,j}$ between the past T returns of asset i and asset j.

$$\rho_{i,j} := \frac{\sigma_{i,j}}{\sqrt{\sigma_i^2}\sqrt{\sigma_j^2}} = \frac{\frac{1}{T-1}(\tilde{r_i}^T \tilde{r_j})}{\sqrt{\frac{1}{T-1}(\tilde{r_i}^T \tilde{r_i})}\sqrt{\frac{1}{T-1}(\tilde{r_j}^T \tilde{r_j})}} = \frac{1}{\sqrt{T}} \frac{\langle \tilde{r}_i, \tilde{r}_j \rangle}{\|r_i\|_2 \|r_j\|_2} = \frac{1}{\sqrt{T}} \cos\theta(\tilde{r}_i, \tilde{r}_j).$$

□

When computing Markowitz minimum volatility portfolios with algorithm 1, assets are considered as being all equal, and only their correlations/covariances are used to decide what weight an asset gets in the resulting portfolio. That means, that there is no hierarchy. This has as consequence that, in algorithmic terms, all assets are potential substitutes of each other if their covariances are low enough. López de Prado describes this by modelling the vector spaced formed by the returns series as a fully connected graph that contains a node for each asset and where each edge represents the correlation between the two assets whose nodes it connects (page 4, [8]). In economic terms, this means that, using our practical implementation framework as example, the agricultural commodities ETP is directly competing for portfolio allocation with the technology sector ETP, even though the asset classes that they represent and assets contained in each of the ETPs are very different: the returns of companies in the technology sector are driven by very different factors than the returns of wheat for example. Introducing hierarchy to the optimisation process could help to get rid of this problem, and also help to approach the asset allocation problem from a more sensible economic perspective, by allowing allocating wealth from a top-down approach. A top down approach is used by many asset managers, and consists in allocating portions of the portfolio to each asset class first, then within each asset class to each region, within each region to each industry, etc. In our practical implementation framework, this would imply that similar assets would compete for allocation against each other, and not against non-similar assets. López de Prado main contribution is to define the Hierarchical Risk Parity (HRP) model that clusters similar assets into groups recursively to later allocate portions of the portfolio to each of the groups starting from the biggest, and then further dividing allocation among the smaller groups contained in the bigger ones (top-down approach).

The HRP model is divided into 3 parts:

1. Tree Clustering.
2. Quasi-Diagonalisation.
3. Recursive Bisection.

In the next subsections, we will describe the first two, which deal with the problem of classifying assets into clusters that contain similar assets, and the third part, which deals with the allocation problem among and between the asset clusters built in the first 2 steps.

3.2.1 Tree Clustering and Quasi-Diagonalisation

In the first step of the HRP algorithm, a hierarchical structure in the form of a tree is built using the covariance matrix as input to define a metric space, in which the distance between assets measures the similarity of their returns over a certain period of time. In the second step the covariance matrix' columns and rows are reordered so that the largest covariances lie on the diagonal.

Definition 3.8 (Page 5, [8]). *Let $R \in \mathbb{R}^{N\times T}$ be a matrix that contains the past $T \in \mathbb{N}$ periods returns of $N \in \mathbb{N}$ assets and let $P = [\rho_{i,j}] \in \mathbb{R}^{N\times N}$ be the corresponding sample correlation matrix. We define the* ***matrix of distances between each pair of assets*** *$D \in \mathbb{R}^{N\times N}$ and the* ***matrix of distances between each asset and the rest of assets*** *$\tilde{D} \in \mathbb{R}^{N\times N}$ as*

$$D = [d_{i,j}], \quad \textit{where} \quad d_{i,j} := \sqrt{\frac{1}{2}(1-\rho_{i,j})} \qquad \textit{for} \quad i,j = 1,\dots,N,$$

$$\tilde{D} = [\tilde{d}_{i,j}], \quad \textit{where} \quad \tilde{d}_{i,j} := \sqrt{\sum_{n=1}^{N}(d_{n,i}-d_{n,j})} \qquad \textit{for} \quad i,j = 1,\dots,N.$$

Theorem 3.9 (Page 13, [8]). *Let $D = [d_{i,j}]$ be a matrix of distances between each pair of assets as in definition 3.8. Then $d : \mathbb{R}^T \times \mathbb{R}^T \to [0,1], d[r_i, r_j] := d_{i,j}$ is a metric for r_i, r_j vectors of T past returns of assets $i, j \in \{1,\dots,N\}$.*

López de Prado provides a simple proof for this theorem in [8] (page 13) by showing that the defined metric is linear multiple of the Euclidean distance between the z-standardised vectors of returns, i.e. after subtracting their mean and dividing by their standard deviation.

Once we have the matrix of distances between each asset and the rest of assets $\tilde{D}$, we can start clustering similar assets into groups. This is done iteratively using the linkage criterion. Initially there are N clusters $u[i] := \{i\}$ containing only one asset (asset i) for $i = 1,\dots,N$, and the matrix $\tilde{D}$ can be interpreted as the table that contains the distances between each pair of clusters.

$$\tilde{D} = \begin{matrix} \\ u[1] \\ u[2] \\ \vdots \\ u[N] \end{matrix} \begin{matrix} \begin{matrix} u[1] & u[2] & \dots & u[N] \end{matrix} \\ \begin{bmatrix} \tilde{d}_{1,1} & \tilde{d}_{1,2} & \dots & \tilde{d}_{1,N} \\ \tilde{d}_{2,1} & \tilde{d}_{2,2} & \dots & \tilde{d}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{d}_{N,1} & \tilde{d}_{N,2} & \dots & \tilde{d}_{N,N} \end{bmatrix} \end{matrix}$$

Algorithm 5: Tree Clustering Algorithm

Input : Matrix of distances between each asset and the rest of assets $\tilde{D} \in \mathbb{R}^{N \times N}$
Output: Set U containing the nested clusters of assets

Initialise $U = \{\{1\}, \ldots, \{N\}\}$;
for $n = 1$ *to* $N - 1$ **do**
  Set $u^* = \{v^*, w^*\} = \arg\min_{v,w \in U, v \neq w} \tilde{d}_{v,w}$; `// Find the 2 nearest clusters.`
  Set $U = (U \setminus \{v^*, w^*\}) \cup \{u^*\}$;
  Delete the v^*, w^* rows and v^*, w^* columns from $\tilde{D}$;
  Add row and column $\tilde{d}_{u^*} = \left[\min_{u \in u^*}\{\tilde{d}_{v,u}\}\right]_{v \in U}$ to $\tilde{D}$, where $\tilde{d}_{u^*,u^*} := 0$
end

López de Prado provides a short example of this algorithm in [8] (pages 5-6).

Example 3.10 (Pages 5-6, [8])**.** *For better understanding, we denote each of the elements in the algorithm that we mention in the example with a subscript indicating the step in the for-loop to which they correspond.*

Let the initial matrix of distances between each asset and the rest of assets $\tilde{D}_0 \in \mathbb{R}^{3\times 3}$ be given by

$$\tilde{D}_0 := \begin{array}{c} \begin{matrix} u[1] & u[2] & u[3] \end{matrix} \\ \begin{bmatrix} 0 & 0.5659 & 0.9747 \\ 0.5659 & 0 & 1.1225 \\ 0.9747 & 1.1225 & 0 \end{bmatrix} \end{array} \begin{matrix} u[1] \\ u[2] \\ u[3] \end{matrix},$$

where the initial clusters $u[i] := \{i\}$ only contain one asset for $i = 1, \ldots, 4$. Moreover, initialise the set $U_0 := \{u[i] : i = 1, \ldots, N\}$. Because we only have three clusters (assets) initially, we only need to run the steps inside the for-loop twice.

First Iteration. *From the entries of the matrix $\tilde{D}_0$, we can read that the nearest clusters are $u[1]$ and $u[2]$. We proceed to create a new cluster $u_1^* := \{u[1], u[2]\}$ that contains these two nearest clusters and to update the set of clusters to $U_1 := \{u[3], u_1^*\} = \{u[3], \{u[1], u[2]\}\}$.*

Now, we can update the matrix $\tilde{D}_0$. First, we compute the column $\tilde{d}_{u_1^*}$.

$$\tilde{d}_{u_1^*} := \begin{bmatrix} 0.9747 \\ 0 \end{bmatrix}.$$

We denote the updated matrix $\tilde{D}_0$ as $\tilde{D}_1$.

$$\tilde{D}_1 := \begin{array}{c} \begin{matrix} u[3] & u_1^* \end{matrix} \\ \begin{bmatrix} 0 & 0.9747 \\ 0.9747 & 0 \end{bmatrix} \end{array} \begin{matrix} u[3] \\ u_1^* \end{matrix}.$$

Second Iteration. *Because we only have 2 clusters left, these are also the closest clusters. We proceed to create a new cluster $u_2^* := \{u[3], u_1^*\}$ that contains these two nearest clusters and to update the set of clusters to $U_2 := \{u_2^*\} := \{\{u[3], \{u[1], u[2]\}\}\}$.*

Now, we can update the matrix $\tilde{D}_1$. First, we compute the column $\tilde{d}_{u_2^*}$.

$$\tilde{d}_{u_2^*} := \begin{bmatrix} 0 \end{bmatrix}.$$

We denote the updated matrix $\tilde{D}_1$ as $\tilde{D}_2$.

$$\tilde{D}_2 := \begin{matrix} u_2^* \\ [\,0\,] \end{matrix} \quad u_2^* \ .$$

The algorithm then terminates and returns the clusters' set $U_2 := \{\{u[3], \{u[1], u[2]\}\}\}$.

After grouping the assets in clusters iteratively, the HRP algorithm uses the hierarchical structure defined by the clusters to reorganise the rows and columns of the covariance matrix, so that similar assets are placed together. To do this, rows and columns are resorted so that the rows and columns corresponding to assets that are contained in the same cluster are contiguous. This results in the covariances of asset returns pairs corresponding to the highest correlations to lie along the diagonal of the covariance matrix.

Example 3.11. *Let the covariance matrix of asset returns used to compute the matrix of distances between each asset and the rest of assets $\tilde{D}_0$ given in example 3.10 be defined as follows*

$$\Sigma = \begin{matrix} \\ u[1] \\ u[2] \\ u[3] \end{matrix} \begin{matrix} \begin{matrix} u[1] & u[2] & u[3] \end{matrix} \\ \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} \end{bmatrix} \end{matrix} \in \mathbb{R}^{3\times 3}.$$

Using the result of the tree clustering algorithm computed in example 3.10, we can compute the corresponding quasi-diagonalised covariance matrix, which is given by

$$\Sigma_{Quasi-diag} = \begin{matrix} \\ u[3] \\ u[1] \\ u[2] \end{matrix} \begin{matrix} \begin{matrix} u[3] & u[1] & u[2] \end{matrix} \\ \begin{bmatrix} \sigma_{3,3} & \sigma_{3,1} & \sigma_{3,2} \\ \sigma_{1,3} & \sigma_{1,1} & \sigma_{1,2} \\ \sigma_{2,3} & \sigma_{2,1} & \sigma_{2,2} \end{bmatrix} \end{matrix} \in \mathbb{R}^{3\times 3}.$$

Now that we know how assets are clustered into groups, we can describe the allocation process.

3.2.2 Recursive Bisection

Once we have established a hierarchical structure of the assets by computing the set of clusters U and the quasi-diagonal covariance matrix of asset returns $\Sigma_{Quasi-diag}$ as described in the previous section, we can use them to compute a target portfolio.

Because the quasi-diagonalised covariance matrix is similar to a diagonal matrix, we can take advantage of the fact that the inverse-variance allocation corresponds to Markowitz minimum volatility allocation given a diagonal matrix to compute a target portfolio by defining the variance of an asset cluster as the variance of the inverse variance allocation of the portfolio containing only the assets in the cluster (bottom-up allocation approach), and by splitting allocations between adjacent clusters in inverse proportion to

their aggregated variances (top-down allocation approach).

Algorithm 6: Recursive Bisection

Input : Set U containing the nested clusters of assets and quasi-diagonal covariance matrix of asset returns $\Sigma_{Quasi-diag} \in \mathbb{R}^{N \times N}$

Output: Hierarchical Risk Parity Portfolio $w \in \mathbb{R}^N$

1 Initialise $w_i = 1$ for all $i = 1, \ldots, N$;
2 Initialise $L = U$;
3 **if** $|L_i| = 1$ *for all* $L_i \in L$ **then**
4 Stop;
5 **end**
6 **else**
7 **foreach** $L_i \in L$ *such that* $|L_i| > 1$ **do**
8 Split the cluster L_i into the two sub-clusters $L_i^{(1)}, L_i^{(2)}$ that compose it;
9 **for** $j = 1, 2$ **do**
10 Define the variance of each sub-cluster as $\tilde{V}_i^{(j)} = \tilde{w}_j^T \Sigma_i^{(j)} \tilde{w}_j^T$, where $\Sigma_i^{(j)}$ is the covariance matrix between the assets contained in sub-cluster L_i^j, and $\tilde{w}_j^T$ is the inverse variance portfolio as in definition 2.3 using $\Sigma_i^{(j)}$ as input;
11 **end**
12 Set $\alpha_i = 1 - \frac{\tilde{V}_i^{(1)}}{\tilde{V}_i^{(1)} + \tilde{V}_i^{(2)}}$;
13 **foreach** *asset* k *in cluster* $L_i^{(1)}$ **do**
14 Set $w_k = \alpha_i w_k$;
15 **end**
16 **foreach** *asset* k *in cluster* $L_i^{(2)}$ **do**
17 Set $w_k = (1 - \alpha_i) w_k$;
18 **end**
19 **end**
20 Go to 3;
21 **end**

The output of algorithm 6 above given the set of nested clusters and the quasi-diagonal covariance matrix from algorithm 5 defines the **Hierarchical Risk Parity Portfolio**, which does not have an analytical definition as is the case for all other portfolios previously defined (EWP, IVP, MMVP, etc.).

The bottom-up allocation approach can be observed in line 9 of the algorithm, where the variance of each sub-cluster is defined by the variance of the inverse allocation portfolio that contains only the assets included in the given sub-cluster.

The top-down allocation approach can be observed in line 10, where the allocation to each sub-cluster is defined by the inverse variance portfolio considering a set of only two assets, in this case the two sub-clusters. This can be easily seen using the definition 2.3. Let $w \in \mathbb{R}^2$ be the IVP for the set of investable assets $A := \{L_i^{(1)}, L_i^{(2)}\}$, then w is given

by

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad \text{where} \quad w_1 := \frac{\frac{1}{\tilde{V}_i^{(1)}}}{\sum_{k=1}^{2} \frac{1}{\tilde{V}_i^{(k)}}} = 1 - \frac{\tilde{V}_i^{(1)}}{\tilde{V}_i^{(1)} + \tilde{V}_i^{(2)}} = \alpha_i,$$

$$w_2 := \frac{\frac{1}{\tilde{V}_i^{(2)}}}{\sum_{k=1}^{2} \frac{1}{\tilde{V}_i^{(k)}}} = 1 - \frac{\tilde{V}_i^{(2)}}{\tilde{V}_i^{(1)} + \tilde{V}_i^{(2)}} = 1 - \alpha_i.$$

Finally, for the portfolio weights, $0 \leq w_k \leq 1$ for all $k = 1, \ldots, N$ holds throughout the algorithm, because they are initialised to 1 and only changed by a multiplication at steps 12 and 15, where $0 \leq \alpha_i \leq 1$ holds for all multiplication factors. $\sum_{k=1}^{N} w_k = 1$ also holds throughout the algorithm, because the weights of higher hierarchical levels are split at each iteration, i.e. the multiplication factors at steps 12 and 15 add up to 1.

Before we present the results of the hierarchical risk-parity method's practical implementation, we would like to flag the flexibility of the method up. Here, we have described the hierarchical risk-parity method as López de Prado does in [8]. However, this method does allow the asset manager to adjust certain parts to their mandate and constrains, as well as to the idiosyncracies of the assets in the portfolio.

1. The distance matrix $\tilde{D} \in \mathbb{R}^{N \times N}$ defined in definition 3.8 can be chosen differently. The asset manager can either define it using the correlations between assets but with a different metric or using a completely different input other than correlations. An example of an alternative metric given by López de Prado in [8] (page 13) is

$$d_{i,j} := \sqrt{1 - |\rho_{i,j}|}.$$

2. Asset can be clustered using a linkage criterion other than the single link criterion (see algorithm 5, line 3). An example of an alternative linkage criterion is Ward's criterion.

3. The inverse variance allocation between clusters used in algorithm 6 can also be replaced by a different method to split allocation between clusters.

López de Prado method should be understood as a framework that advocates for the use of hierarchical structures in the asset allocation process, and not as method for which the input parameters should be optimised in order to get the best results. Asset managers are required to use the framework and adapt it to their assets, goals and constrains.

We have described in the past two sections, how the hierarchical risk parity allocation model clusters assets into hierarchical groups and allocates wealth to each of the clusters recursively. Now we can continue to the results of its practical implementation.

3.2.3 Practical Implementation

In this final subsection, we present the results of López de Prado's hierarchical risk parity practical implementation, and compare the results to all the previously defined portfolio construction models. The resulting portfolios can be found in appendix F.

First, we see the dendrogram plots of the clustered assets in figures 19, 20 and 21, which correspond to the initial clustering and the clustering at the 2 peaks in portfolio turnover in figure 23. The vertical distances in the plots represent the distances between each asset/asset group that is represented by horizontal lines linking two vertical lines of the tree. Here we can observe that the dendrogram or tree in April and July 2020 are

deeper than in the January rebalancing point, indicating that the distances between assets are greater. On the contrary, in January 2020, assets are better distributed in high-order groups containing more assets each.

In the January example, assets that tend to exhibit less variance of returns and that are considered *safer* investments are clustered together like fixed income ETPs and the precious metals ETP (cluster containing IMTB, DBP, IAGG, ILTB, ISTB). Other similar ETPs like the energy producers ETF (FILL) and the base metals (DBB) and energy funds (DBE) are also clustered together. ETFs of sectors that tend to follow the economic cycle like financials, industrials, consumer discretionary, etc. also form a cluster (RXI, IXN, IXP, IXG, EXI, MXI) marked in red. Economic cycle following sectors are those that profit from higher inflation, higher growth and higher unemployment. The distance between the top-level group (the one that contains the last two clusters) and the bottom groups (the ones that contain only one asset) is less than 1, as we can see with the help of the left vertical axis labels.

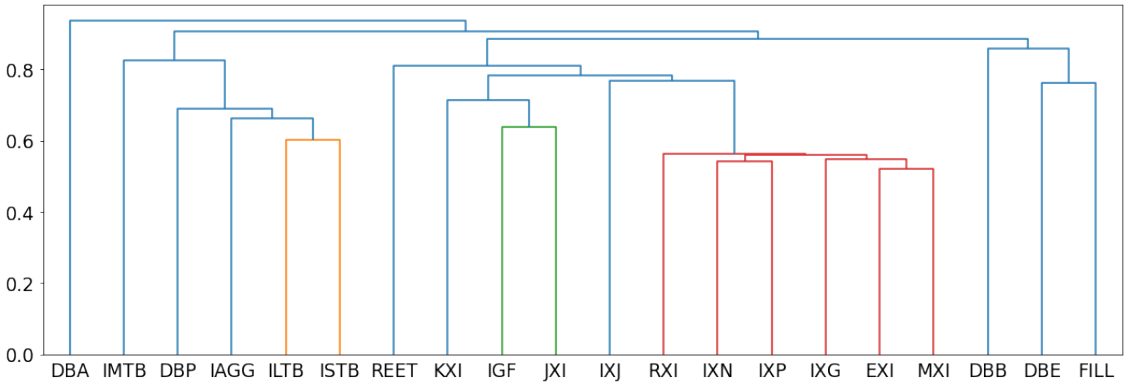

Figure 19: Tree Clustering Algorithm Result for a Portfolio of 20 Assets at the Rebalancing Point in January 2020.

In April and July, the separation between *safer* and *riskier* investments into different clusters is more pronounced. All equity ETFs are part of a big cluster, while fixed income ETFs are part of another separate big cluster, with the edges that connect the 2 sub-trees at a relatively high level. This is a visualisation of the fact that the returns' time-series of the ETPs in these two asset classes were less similar than in January 2020 (see theorem 3.7 and the interpretation that followed). The distance between the top-level group (the one that contains the last two clusters) and the bottom groups (the ones that contain only one asset) is greater in April and slightly smaller in July than 1, as we can see with the help of the left vertical axis. The first clusters are built at the 0.3 level, while in the January plot, we see that the first clustered assets appear near the 0.6 level, indicating greater distances between assets in January than in April and July. Nevertheless, higher-order clusters are further apart in the latter months than in the former one.

Second, we have a look at the target allocation over time. Figure 22 shows that the changes in target allocation over time are very similar to those of the inverse variance allocation method (see figure 5). This is not surprising, because the hierarchical risk parity method uses inverse variance allocation to split portfolio weights among the different clusters constructed by the tree clustering algorithm (algorithms 5 and 6).

Table 5 contains a summary of the Sharpe ratio and turnover figures of all portfolio construction models analysed in this bachelor thesis. Here, we can see that the hierarchical risk parity method (HRP) is the method with the highest risk-adjusted returns, and its

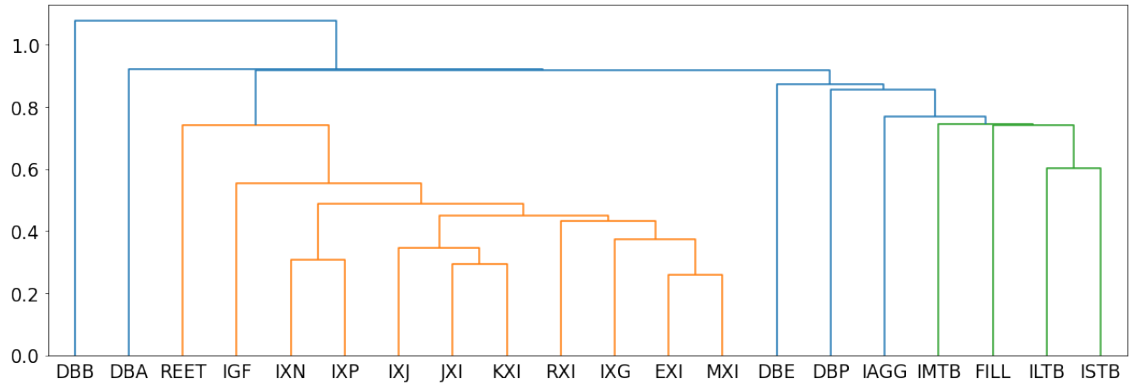

Figure 20: Tree Clustering Algorithm Result for a Portfolio of 20 Assets at the Rebalancing Point in April 2020

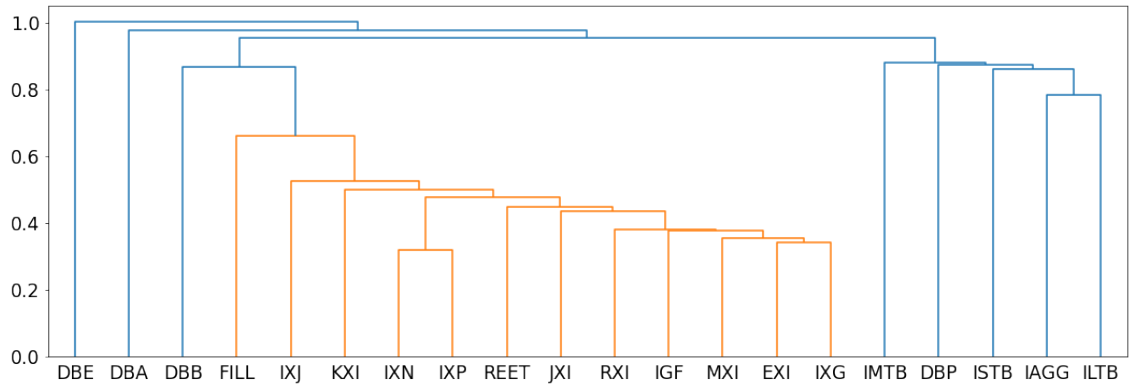

Figure 21: Tree Clustering Algorithm Result for a Portfolio of 20 Assets at the Rebalancing Point in July 2020

turnover is only slightly above the turnover of the inverse variance allocation method.

Portfolio Optimisation Method	**Sharpe Ratio**	**Turnover**
EWP	0.36	0.003634
IVP	1.008	0.007396
MMVP	-0.017	0.018856
MMVP-LW	0.5	0.012527
MMVP-DN-K4	-0.134	0.027798
HRP	1.097	0.008984

Table 5: Sharpe Ratio and Turnover Figures of the Equal Weight, Inverse Variance Portfolios, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit, Wolf's Shrinkage and Markowitz Minimum Volatility after De Nard's Shrinkage and Hierarchical Risk Parity Portfolios for the Backtesting Period from 01/01/2020 to 31/12/2020.

Figure 23 gives a more detailed representation of turnover. Here, we can see that the hierarchical risk parity method (HRP) closely tracks the turnover figures of the inverse variance method (IVP), which are consistently lower than those of the different Markowitz's minimum volatility portfolio construction methods (MMVP, MMVP-LW and

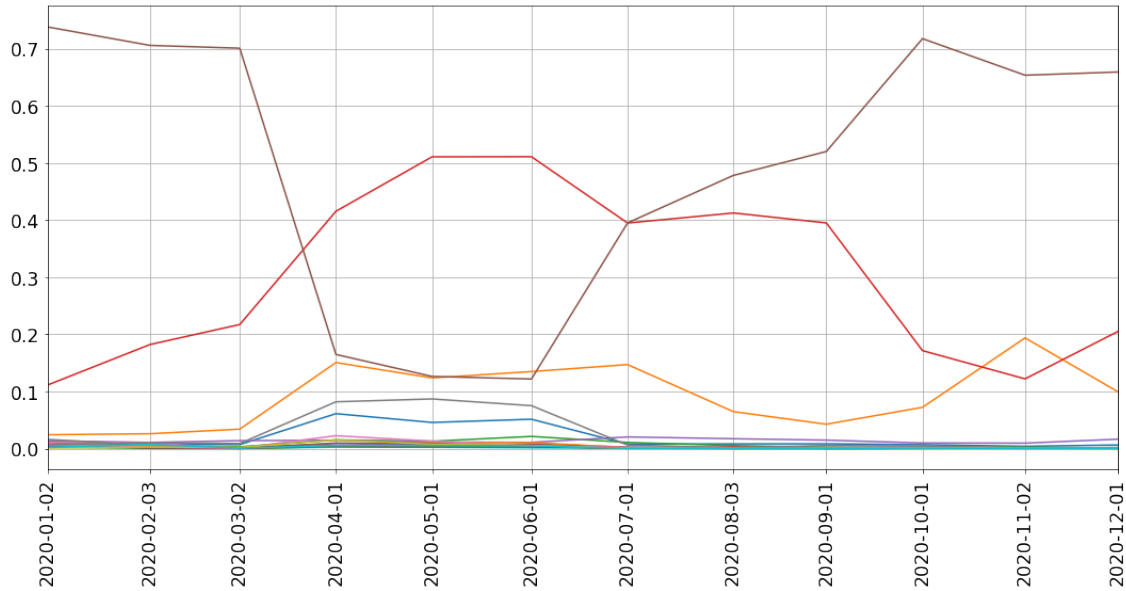

Figure 22: HRP Target Allocation of the Practical Implementation Portfolio Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

MMVP-DN-K4).

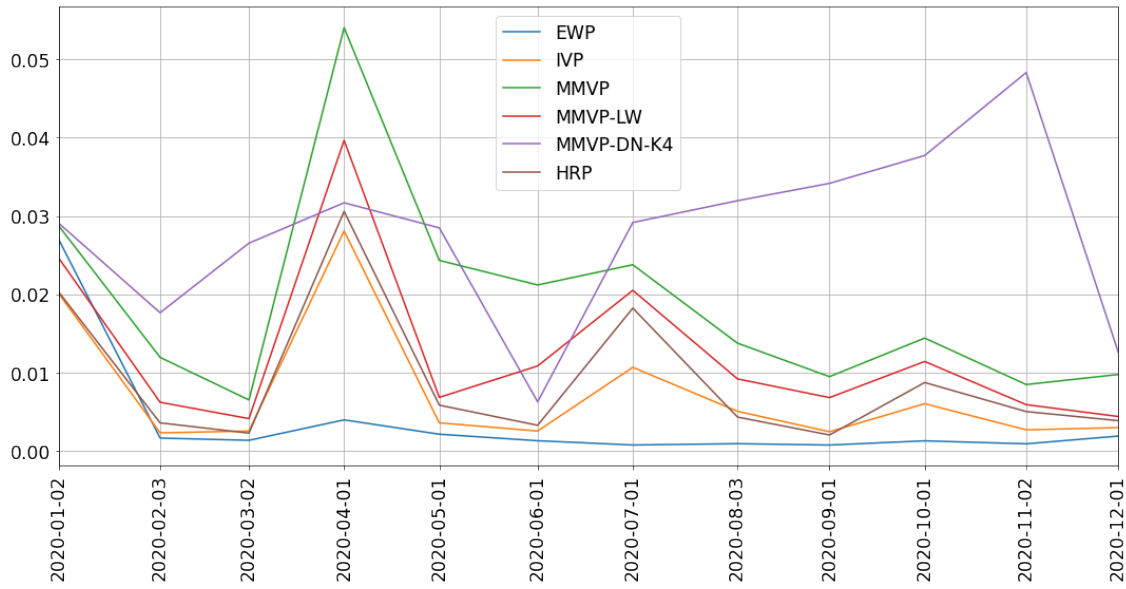

Figure 23: Turnover of Equal Weight, Inverse Variance Portfolios, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit and Wolf's Shrinkage, Markowitz Minimum Volatility after De Nard's Shrinkage and Hierarchical Risk Parity Portfolios at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

Finally, figure 24 presents the concentration figures of each monthly portfolio for all methods analysed. The hierarchical risk parity method (HRP) generates portfolios that

are consistently more concentrated with respect to the Herfindahl-Hirsch-Index than any of the other methods, even higher than the inverse variance (IVP) and Markowitz's minimum volatility portfolios computed using Ledoit and Wolf's covariance shrinkage method (MMVP-LW). This is a rather unexpected result, because López de Prado praises the hierarchical risk parity method as a way to reduce the high concentration present in Markowitz's minimum volatility portfolios (page 2, [8]).

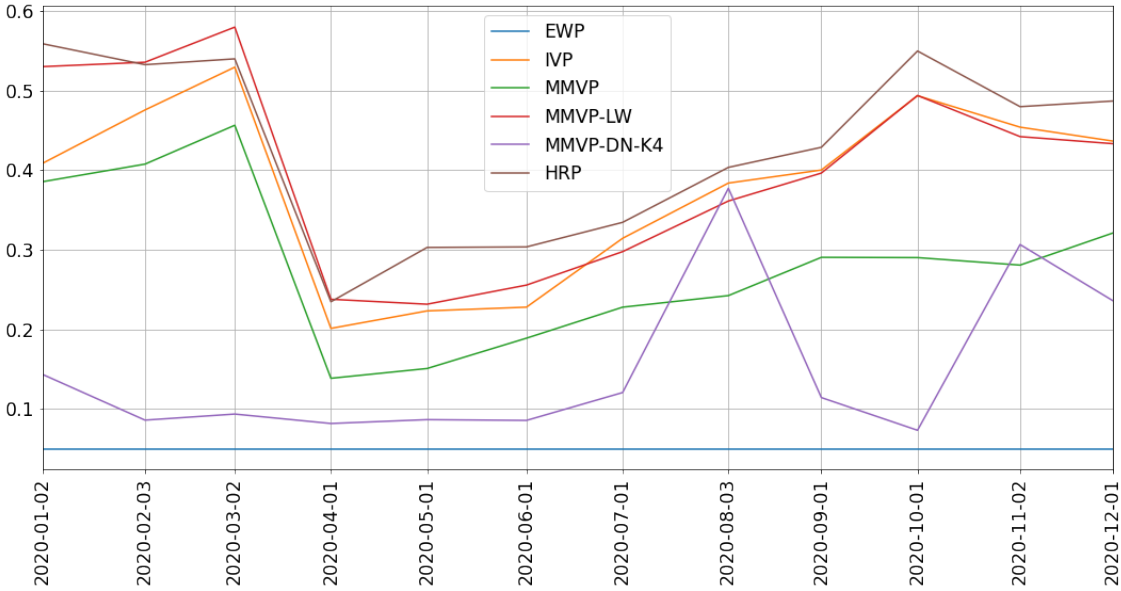

Figure 24: nHHIn of the Equal Weight, Inverse Variance, Markowitz Minimum Volatility, Markowitz Minimum Volatility after Ledoit and Wolf's Shrinkage, Markowitz Minimum Volatility after De Nard's Shrinkage and Hierarchical Risk Parity Portfolios Composed of 20 Assets at Each of the 12 Rebalancing Points in the Backtesting Period from 01/01/2020 to 31/12/2020.

We conclude that the hierarchical risk parity method produces the best out-of-sample performance among all portfolio optimisation methods analysed, while keeping portfolio turnover low. However, the method achieves that by generating portfolios that are also the most concentrated.

4 Conclusion

Asset managers confronted with the asset allocation problem in section 2 can choose among a huge number of asset allocation models. In this thesis, we have described optimal portfolios computed using Markowitz's portfolio allocation model, and focused on analysing its risk-based version, i.e. Markowitz's minimum volatility portfolios. Markowitz's minimum volatility portfolios are the portfolios that showed the lowest variance during the time window from which past data was sampled to compute it (past T periods returns used to compute the sample covariance matrix of asset returns). These portfolios are however highly sensitive to the condition number of the sample covariance matrix, which rises when correlations rise and the differences in return's variances of the assets become smaller. We have shown that this can lead to a significant under-performance in terms of out-of-sample risk-adjusted returns, concentration and turn-over of Markowitz's minimum volatility portfolios compared to naive portfolio allocation models like the equal weight and inverse variance allocation models.

To deal with instability in Markowitz's minimum volatility portfolios, we have described two methods that act on the covariance matrix used as input in the optimisation problem and one method which is a hybrid between the inverse variance allocation and Markowitz's minimum volatility allocation. The first two are two covariance shrinkage methods developed by Ledoit and Wolf and another developed by De Nard, while the last is a method developed by López de Prado. We have also tested these 3 methods in the one-year period of market stress given by the outbreak of the COVID-19 crisis in 2020, and compared them to the original Markowitz minimum volatility model and the two naive allocation models mentioned above (equal weight and inverse variance allocation models).

Covariance shrinkage methods try to tackle instability by defining a structured predictor for the covariance matrix, and using a convex combination of the structured predictor and the sample covariance matrix as input for the computation of Markowitz's minimum volatility portfolios. The different covariance shrinkage methods are given by different definitions of the structured predictor and of the shrinkage constant that defines the factor with which the structured predictor is weighted in the convex combination.

Ledoit and Wolf propose using the constant correlation matrix as structured predictor and the shrinkage constant given by the consistent estimator of the shrinkage constant that minimises the loss function given by the Frobenius norm of the difference between the shrunk and true covariance matrix and trimmed to the interval $[0, 1]$. The resulting shrunk covariance matrices exhibited a lower condition number in our practical implementation. Moreover, using the resulting shrunk covariance matrix for the computation of Markowitz's minimum volatility portfolios also led to sharply improved out-of-sample risk-adjusted returns while reducing portfolio turnover. However, this covariance shrinkage method failed to reduce portfolio concentration in our practical implementation. Ledoit and Wolf's method comes with some limitations, which are given by the fact that all assets are considered to be equal, and no differentiation between asset classes, sectors or regions are made in the computation of the structured predictor. Moreover, the consistent estimator of the optimal shrinkage constant is only consistent if we assume that asset return's random variables are independent and identically distributed with finite fourth moments, which is not true in general.

De Nard's covariance shrinkage method tries to tackle the limitations of Ledoit and Wolf's method by clustering the assets into homogeneous groups to later define a structured predictor of the covariance matrix for each group separately. To do this, the K-means algorithm is used, and a new structured predictor called the generalised constant-variance-covariance shrinkage target is defined. The shrinkage constant in De Nard's method is

the same as in Ledoit and Wolf's method, but using the generalised constant-variance-covariance shrinkage target as input for its computation. In our practical implementation, this method led to the lowest input covariance matrix' condition numbers, which were well below those of the sample covariance matrices and those of Ledoit and Wolf's shrunk covariance matrices. The condition numbers were also very stable across time, and a significant reduction in concentration could be observed. However, this method resulted in the highest turnover figures and was the worst performing in terms of out-of-sample risk-adjusted returns among all methods simulated. This might be explained by the fact that our practical implementation only used 20 assets, which were clustered into 4 homogeneous groups. Such a reduced number of assets and the fact that the K-means algorithm is very sensitive to outliers, could be the reason for the bad results in terms of Sharpe ration and portfolio turnover.

Finally, we described and implemented López de Prado hierarchical risk parity model, which can be considered a hybrid method between the inverse variance allocation and Markowitz's minimum volatility models. It was developed with the objective of improving the stability and out-of-sample performance of Markowitz's minimum volatility portfolios. It, first, hierarchically classifies the assets in the allocation problem using the sample covariance and its corresponding correlation matrix to give the problem a structure. Then, it allocates capital along the hierarchy using the inverse variance allocation model between the clusters. This method is not only an asset allocation model, but can also be understood as a framework that allows for some flexibility in terms of defining the tree structure and how allocations are then split between clusters in the structure. In our practical implementation, López de Prado hierarchical risk parity method preformed the best with respect to out-of-sample risk-adjusted returns. With respect to portfolio turnover, this method was only second to the equal weight allocation method. However, hierarchical risk parity portfolios were consistently among the most concentrated portfolios across time.

We conclude that asset managers should not discard naive portfolio allocation models like the equal weight or inverse variance allocation models in favour of more complex methods like Markowitz's. If they are to consider using Markowitz's minimum volatility portfolios, they must keep in mind the sensitivity to the sample covariance matrix' condition number in their computation. Covariance shrinkage methods can indeed reduce the condition number of the input covariance matrix and lead to better performance in terms of of risk-adjusted returns, turnover and concentration. Finally, alternative asset allocation methods like the hierarchical risk parity method can be good substitutes for Markowitz's model, even if the portfolios generated are not optimal in-sample.

As a final note, we would like to remind again that the empirical results of the practical implementation are limited by the fact that we have only analysed one past period of market stress. As it is often quoted in financial literature and marketing materials, past returns are not a guarantee of future performance. Moreover, better performance and optimal portfolios in periods of market stress might not result in better performance and optimal portfolios outside periods of market stress, and the overall (periods composed of market stress and calm markets phases) performance, as well as optimal portfolios, might differ from those in market stress phases.

References

[1] Folkmar Bornemann. *Numerical Linear Algebra*. Springer, Cham, 2018. ISBN 978-3-319-74222-9. doi: 10.1007/978-3-319-74222-9.

[2] Ghassan A. Chammas. *Portfolio Concentration*. PhD thesis, Erasmus University Rotterdam, January 2017. URL http://hdl.handle.net/1765/94975.

[3] The NumPy community. NumPy v1.21 API Reference, numpy.linalg.lstsq, 2021. URL https://numpy.org/doc/stable/reference/generated/numpy.linalg.lstsq.html.

[4] The SciPy community. SciPy v1.7.1 API Reference, scipy.linalg.solve, 2021. URL https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve.html.

[5] QuantConnect Corporation. QuantConnect Sharpe Ratio Code, 2021. URL https://github.com/QuantConnect/Lean/blob/master/Common/Statistics/Statistics.cs#L505.

[6] QuantConnect Corporation. QuantConnect Website, 2021. URL https://www.quantconnect.com/.

[7] Gianluca De Nard. Oops! i shrunk the sample covariance matrix again: Blockbuster meets shrinkage. *Journal of Financial Econometrics (2020, forthcoming)*, 06 2019. doi: 10.2139/ssrn.3400062. https://ssrn.com/abstract=3400062.

[8] Marcos López de Prado. Building diversified portfolios that outperform out-of-sample. *Journal of Portfolio Management*, 42, 2016. https://ssrn.com/abstract=2708678.

[9] Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *Review of Financial Studies*, 22, 2009. http://faculty.london.edu/avmiguel/DeMiguel-Garlappi-Uppal-RFS.pdf.

[10] The Nobel Foundation. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1990, 1990. URL https://www.nobelprize.org/prizes/economic-sciences/1990/summary/.

[11] Dietmar Franzen and Klaus Schäfer. *Assetmanagement. Portfolio Bewertung, Investmentstrategien und Risikoanalyse*. Schäffer Poeschel, 2018. ISBN 978-3-7910-3829-2.

[12] Intel®. Developer Reference for Intel® oneAPI Math Kernel Library - C Version 2021.3, ?posv, 2021. URL https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-driver-routines/posv.html.

[13] iShares by BlackRock Inc. iShares Online ETF Screener, 2021. URL https://www.ishares.com/us/products/etf-investments.

[14] Olivier Ledoit and Michael Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5):603–621, 2003. ISSN 0927-5398. doi: https://doi.org/10.1016/S0927-5398(03)00007-0. URL https://www.sciencedirect.com/science/article/pii/S0927539803000070.

[15] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004. ISSN 0095-4918. doi: 10.3905/jpm.2004.110. URL https://jpm.pm-research.com/content/30/4/110.

[16] Jörg Liesen and Volker Mehrmann. *Lineare Algebra*. Springer Spektrum, 2014. ISBN 978-3-658-06610-9.

[17] Harry M. Markowitz. Portfolio selection, efficient diversification of investments. *Cowles Foundation for Research in Economics at Yale University*, 16, 1959. https://cowles.yale.edu/sites/default/files/files/pub/mon/m16-all.pdf.

[18] Richard Michaud and Robert Michaud. *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation 2nd Edition*. Oxford University Press, 01 2008.

[19] Holger Nobach. Practical realization of bessel's correction for a bias-free estimation of the auto-covariance and the cross-covariance functions. 05 2017. http://nambis.bplaced.net/download/text/BC4corr.pdf.

[20] Bernd Scherer. *Portfolio Construction and Risk Budgeting*. Risk Books, 2015. ISBN 978-178272-100-0.

[21] William F. Sharpe. A simplified model for portfolio analysis. *Management Science*, 9(2):277–293, 1963. doi: 10.1287/mnsc.9.2.277. URL https://doi.org/10.1287/mnsc.9.2.277.

[22] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997. ISBN 0-89871-361-7.

Appendices

A Practical Implementation Framework

To show the problems of Markowitz's minimum volatility portfolios in real applications, and compare these portfolios to the portfolios computed using other methods and techniques, each of the techniques described here is implemented for a portfolio of exchange traded products (ETPs) representing 4 assets classes: equities, fixed income, real estate and commodities. We use the QuantConnect platform [6] for the implementation and as the data source. This platform and its API allow us to perform backtests with one-minute resolution data. A backtest is nothing else than a simulation of a portfolio with past data to see how it would have behaved, if we had hold it during that past period considered in the backtest. With one-minute resolution data, we have price data that is sampled once per minute. This allows us to model the changes in portfolio value as well as the entry and closing price of orders accurately considering the fact that the holding period is at least one month.

A.1 Selected ETPs

Due to the lack of Euro denominated ETP data in the QuantConnect data bank, we decided to use US dollar denominated ETPs which are intended for US based traders. For the selection of ETPs, we use iShares website [13], because it provides a screener for its ETPs[6].

For the equity ETPs, we selected equity index exchange traded funds (ETFs) with regional category "Global" by sectors and industries. From all of the ETPs fulfilling those conditions shown in the online screener, we chose the following ETFs (alphabetically ordered by ticker):

- iShares Global Industrials ETF with ticker "EXI" which seeks to track the investment results of an index composed of global equities in the industrials sector. More specifically, the index tracked is the S&P Global 1200 Industrials Sector Index (TM). The top sub-sectors represented in the ETF holdings are capital goods, transportation and commercial and professional services. Source: iShares Global Industrials ETF Fact Sheet as of June 30, 2021.

- iShares MSCI Global Energy Producers ETF with ticker "FILL" which seeks to track the investment results of an index composed of global equities of companies primarily engaged in the business of energy exploration and production. More specifically, the index tracked is the MSCI ACWI Select Energy Producers Investable Market Index. The top sub-sectors represented in the ETF holdings are integrated oil and gas, oil and gas exploration and production, oil and gas refining and marketing and transportation, coal and consumable fuels and oil and gas drilling. Source: iShares MSCI Global Energy Producers ETF Fact Sheet as of June 30, 2021.

- iShares Global Infrastructure ETF with ticker "IGF" which seeks to track the investment results of an index composed of developed market equities in the infrastructure industry. More specifically, the index tracked is the S&P Global Infrastructure Index. The top sub-sectors represented in the ETF holdings are utilities, transportation and energy. Source: iShares Global Infrastructure ETF Fact Sheet as of June 30, 2021.

[6] Disclosure: the author of this thesis is not associated with iShares, BlackRock or any of its companies.

- iShares Global Financials ETF with ticker "IXG" which seeks to track the investment results of an index composed of global equities in the financials sector. More specifically, the index tracked is the S&P Global 1200 Financials Sector Index (TM). The top sub-sectors represented in the ETF holdings are banks, diversified financials and insurance. Source: iShares Global Financials ETF Fact Sheet as of June 30, 2021.

- iShares Global Healthcare ETF with ticker "IXJ" which seeks to track the investment results of an index composed of global equities in the healthcare sector. More specifically, the index tracked is the S&P Global 1200 Healthcare Sector Index (TM). The top sub-sectors represented in the ETF holdings are pharmaceuticals, biotech and life sciences and health care equipment and services. Source: iShares Global Healthcare ETF Fact Sheet as of June 30, 2021.

- iShares Global Tech ETF with ticker "IXN" which seeks to track the investment results of an index composed of global equities in the technology sector. More specifically, the index tracked is the S&P Global 1200 Information Technology Sector Index (TM). The top sub-sectors represented in the ETF holdings are software and services, tech hardware and equipment, semiconductors and semiconductor equipment. Source: iShares Global Tech ETF Fact Sheet as of June 30, 2021.

- iShares Global Comm Services ETF with ticker "IXP" which seeks to track the investment results of an index composed of global equities in the communication services sector. More specifically, the index tracked is the S&P Global 1200 Communication Services 4.5/22.5/45 Capped Index. The top sub-sectors represented in the ETF holdings are interactive media and services, integrated telecommunication services, movies and entertainment, cable and satellite and wireless telecommunication services. Source: iShares Global Comm Services ETF Fact Sheet as of June 30, 2021.

- iShares Global Utilities ETF with ticker "JXI" which seeks to track the investment results of an index composed of global equities in the utilities sector. More specifically, the index tracked is the S&P Global 1200 Utilities (Sector) Capped Index. The top sub-sectors represented in the ETF holdings are electric utilities, multi-utilities, gas utilities and water utilities. Source: iShares Global Utilities ETF Fact Sheet as of June 30, 2021.

- iShares Global Consumer Staples ETF with ticker "KXI" which seeks to track the investment results of an index composed of global equities in the consumer staples sector. More specifically, the index tracked is the S&P Global 1200 Consumer Staples Sector Capped Index. The top sub-sectors represented in the ETF holdings are food, beverages and tobacco, household and personal products and food and staples retailing. Source: iShares Global Consumer Staples ETF Fact Sheet as of June 30, 2021.

- iShares Global Materials ETF with ticker "MXI" which seeks to track the investment results of an index composed of global equities in the materials sector. More specifically, the index tracked is the S&P Global 1200 Materials Sector Index (TM). The top sub-sectors represented in the ETF holdings are chemicals, metals and mining, construction materials and containers and packaging. Source: iShares Global Materials ETF Fact Sheet as of June 30, 2021.

- iShares Global REIT ETF with ticker "REET" which seeks to track the investment results of an index composed of global real estate equities in developed and emerging markets. More specifically, the index tracked is the FTSE EPRA Nareit Global REITS Net Total Return Index. The top sub-sectors represented in the ETF holdings are retail REIT's, industrial REIT's, residential REIT's and specialised REIT's. Source: iShares Global REIT ETF Fact Sheet as of June 30, 2021. This single equity ETF is used as a proxy to represent the real estate asset class.

- iShares Global Consumer Discretionary ETF with ticker "RXI" which seeks to track the investment results of an index composed of global equities in the consumer discretionary sector. More specifically, the index tracked is the S&P Global 1200 Consumer Discretionary Sector Capped Index. The top sub-sectors represented in the ETF holdings are retailing, autos and components, consumer durables and consumer services. Source: iShares Global Consumer Discretionary ETF Fact Sheet as of June 30, 2021.

For the fixed income ETPs, we selected fixed income index exchange traded funds (ETFs) of the iShares *Core* category. From all of the ETPs fulfilling those conditions shown in the online screener, we chose the following ETFs (alphabetically ordered by ticker):

- iShares Core International Aggregate Bond ETF with ticker "IAGG" which seeks to track the investment results of an index composed of global non-U.S. dollar-denominated investment-grade bonds that mitigates exposure to fluctuations between the value of the component currencies and the U.S. dollar. More specifically, the index tracked is the Bloomberg Barclays Global Aggregate ex USD 10% Issuer Capped (Hedged) Index. Most of the ETF holdings are rated single A or higher, and have maturities between 3 and 10 years. Source: iShares Core International Aggregate Bond ETF Fact Sheet as of June 30, 2021.

- iShares Core 10+ Year USD Bond ETF with ticker "ILTB" which seeks to track the investment results of an index composed of U.S. dollar-denominated bonds that are rated either investment-grade or high-yield with remaining maturities greater than 10 years. More specifically, the index tracked is the Bloomberg Barclays U.S. Universal 10+ Year Index. Most of the ETF holdings are rated AAA or BBB. Source: iShares Core 10+ Year USD Bond ETF Fact Sheet as of June 30, 2021.

- iShares Core 5-10 Year USD Bond ETF with ticker "IMTB" which seeks to track the investment results of an index composed of U.S. dollar-denominated bonds that are rated either investment-grade or high-yield with remaining effective maturities between 5 and 10 years. More specifically, the index tracked is the Bloomberg US Universal 5-10 Years Index. Most of the ETF holdings are rated AAA. Source: iShares Core 5-10 Year USD Bond ETF Fact Sheet as of June 30, 2021.

- iShares Core 1-5 Year USD Bond ETF with ticker "ISTB" which seeks to track the investment results of an index composed of U.S. dollar-denominated bonds that are rated either investment-grade or high-yield with remaining maturities between 1 and 5 years. Most of the ETF holdings are rated AAA. Source: iShares Core 1-5 Year USD Bond ETF Fact Sheet as of June 30, 2021.

Because iShares commodities ETPs only offer exposure to gold and silver specifically and to certain commodity factor strategies, we turn to Invesco DB Funds for the selection of ETPs that will cover the commodities exposure of our portfolios. We chose Invesco

DB Funds because they offer exposure to commodities aggregated by sector[7]. From all commodity-linked funds offered by Invesco, we chose the following ones:

- Invesco DB Agriculture Fund with ticker DBA which seeks to track changes in the level of the DBIQ Diversified Agriculture Index Excess Return over time. The 11 agricultural commodity futures contracts included in the holdings are corn, soybeans, sugar, coffee, live cattle, lean hogs, cocoa, wheat, Kansas wheat, feeder cattle and cotton. Source: Invesco Agriculture Fund Fact Sheet as of June 30, 2021.

- Invesco DB Base Metals Fund with ticker DBB which seeks to track changes in the level of the DBIQ Optimum Yield Industrial Metals Index Excess Return over time. The 3 industrial metals commodity futures contracts included in the holdings are copper, aluminium and zinc. Source: Invesco DB Base Metals Fund Fact Sheet as of June 30, 2021.

- Invesco DB Energy Fund with ticker DBE which seeks to track changes in the level of the DBIQ Optimum Yield Energy Index Excess Return over time. The 5 energy commodity futures contracts included in the holdings are WTI crude, gasoline, NY Harbor ULSD, Brent crude and natural gas. Source: Invesco DB Energy Fund Fact Sheet as of June 30, 2021.

- Invesco DB Precious Metals Fund with ticker DBP which seeks to track changes in the level of the DBIQ Optimum Yield Precious Metals Index Excess Return over time. The 2 precious metals commodity futures contracts included in the holdings are gold and silver. Source: Invesco DB Precious Metals Fund Fact Sheet as of June 30, 2021.

We acknowledge that the selecting of assets could have been more granular as well as coarser. The ETPs selected might also not be the cheapest available to represent their corresponding index. However, this selection is intended for the illustration of the stability of the different portfolio construction methods presented here, and not as asset selection for the portfolio construction of a particular investor.

In figures 25 and 26, we can see the end-of-day absolute return on an investment of 1 U.S. dollar in each of the selected equity ETPs from end 2019 until end 2020. The zero return level and the start of the backtesting period are marked by a horizontal and a vertical black dashed line respectively. The plot to the left of the vertical black dashed line represents the performances in the period used to warm-up the algorithm with data to be able to compute the first portfolio at the beginning of January 2020. The steep drop in both charts around March 2020 marks the results of the COVID-19 crisis.

In figure 26, we observe the great run and out-performance of the iShares Global Tech ETF with ticker IXN.

In figure 27, we can see the end-of-day absolute return on an investment of 1 U.S. dollar in the ETP selected as a proxy to represent the real estate asset class.

In figure 28, we can see the end-of-day absolute return on an investment of 1 U.S. dollar in each of the selected fixed income ETPs from end 2019 until end 2020. The zero return level and the start of the backtesting period are marked by a horizontal and a vertical black dashed line respectively. The plot to the left of the vertical black dashed line represents the performances in the period used to warm-up the algorithm with data to be able to compute the first portfolio at the beginning of January 2020. The steep drop in both charts around March 2020 marks the results of the COVID-19 crisis, from which all fixed income ETFs recovered quickier than equity ETFs.

[7]Disclosure: the author of this thesis is not associated with any of Invesco's companies.

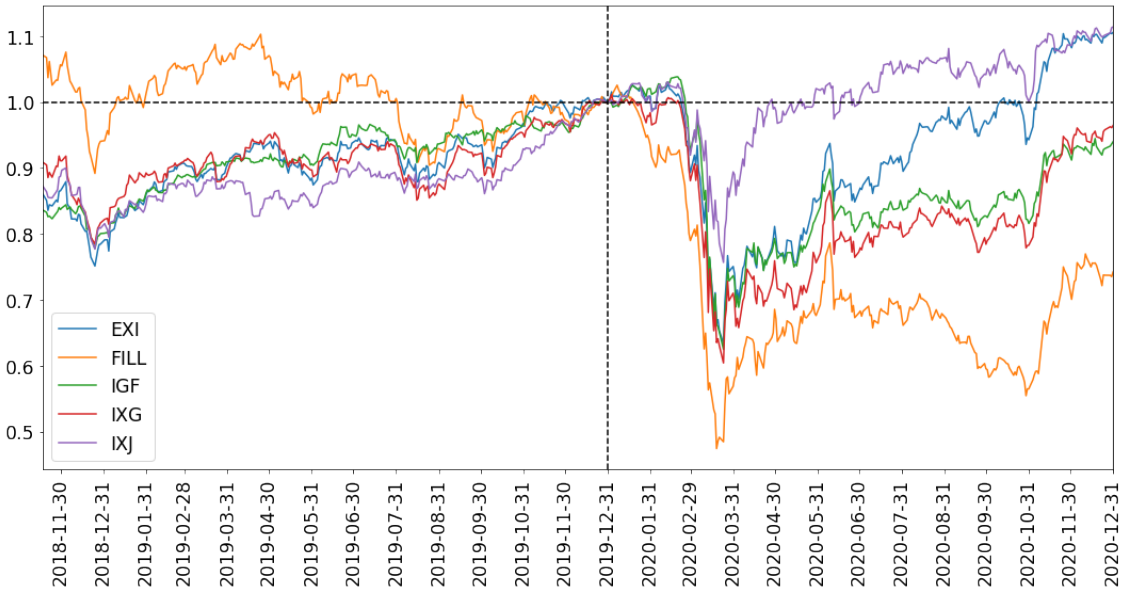

Figure 25: Equity ETPs' Returns on a 1 U.S. Dollar Investment from End 2019 until End 2020.

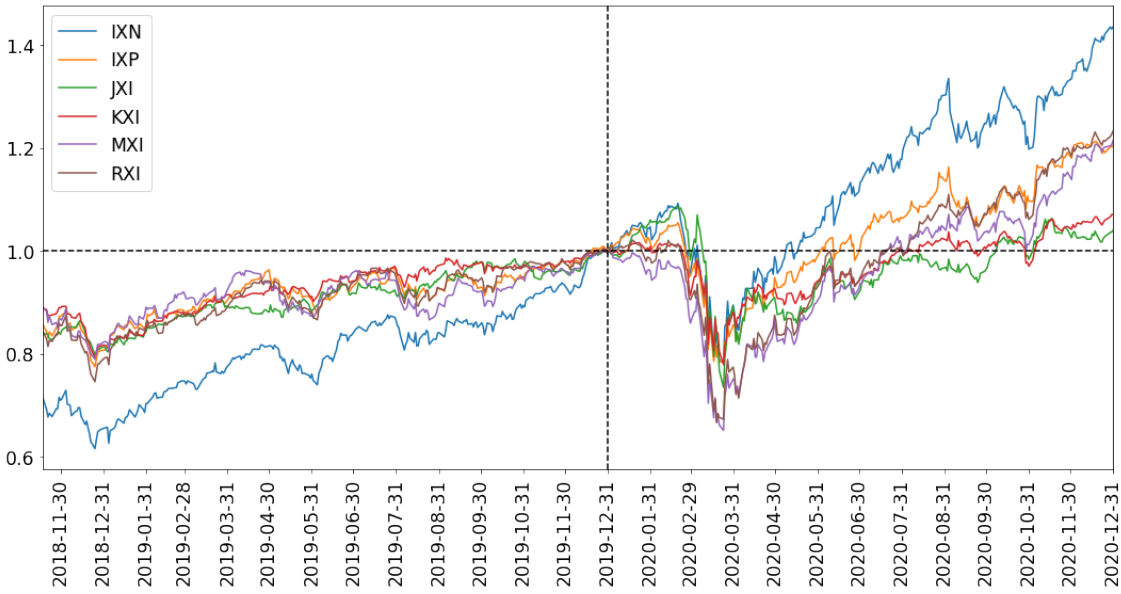

Figure 26: Equity ETPs' Returns on a 1 U.S. Dollar Investment from End 2019 until End 2020.

In figure 29, we can see the end-of-day absolute return on an investment of 1 U.S. dollar in each of the selected commodity ETPs from end 2019 until end 2020. The zero return level and the start of the backtesting period are marked by a horizontal and a vertical black dashed line respectively. The plot to the left of the vertical black dashed line represents the performances in the period used to warm-up the algorithm with data to be able to compute the first portfolio at the beginning of January 2020. The steep drop in both charts around March 2020 marks the results of the COVID-19 crisis. The

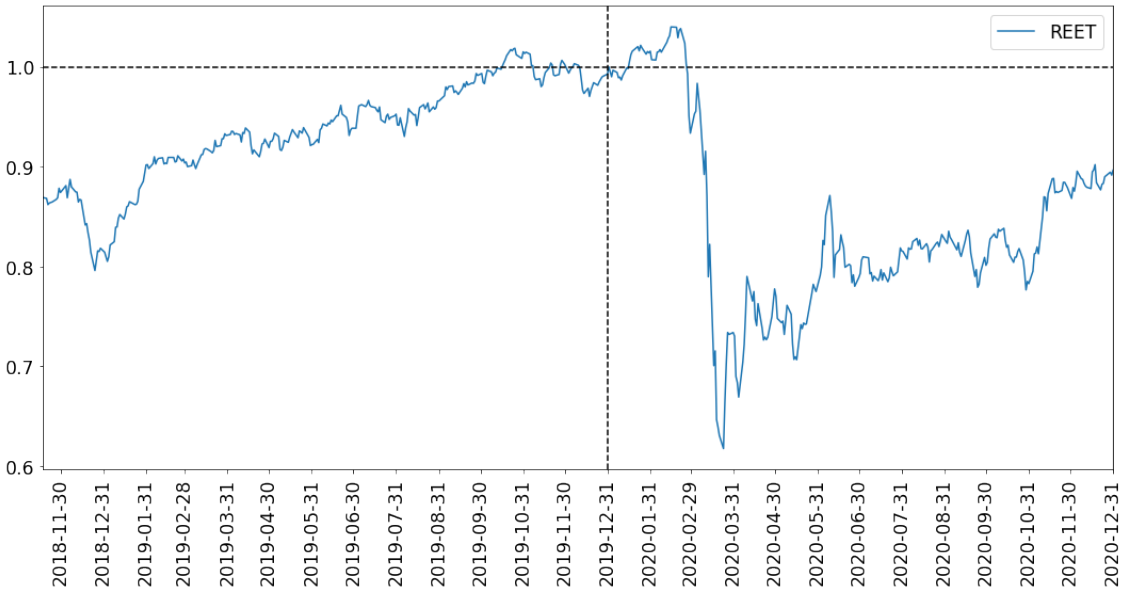

Figure 27: Real Estate ETP Returns on a 1 U.S. Dollar Investment from End 2019 until End 2020.

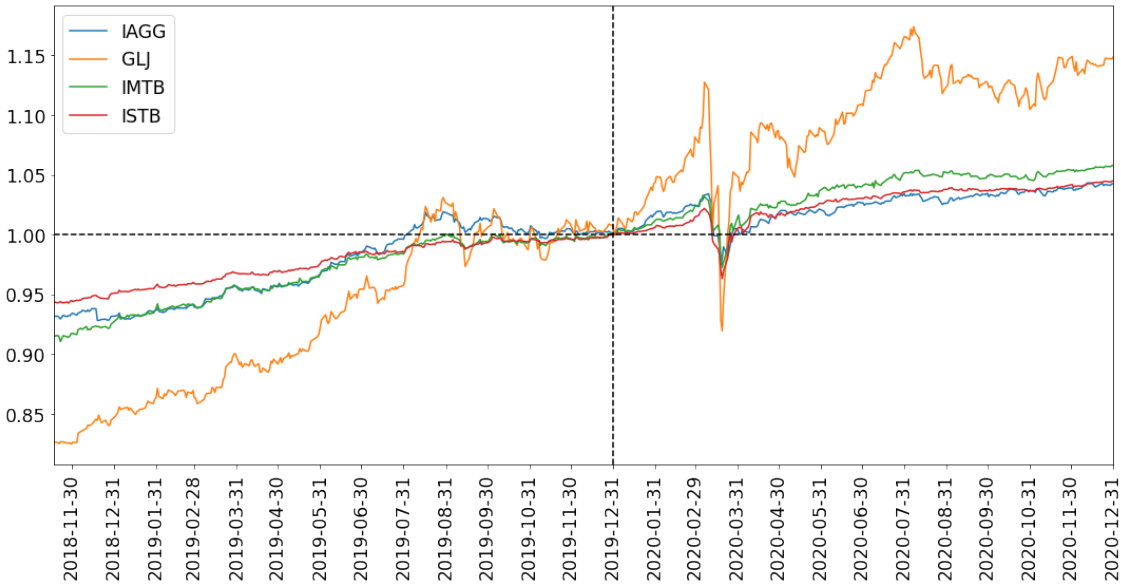

Figure 28: Fixed Income ETPs' Returns on a 1 U.S. Dollar Investment from End 2019 until End 2020.

red line represents the returns of the precious metals fund, which recovered quicker than the energy fund in green.

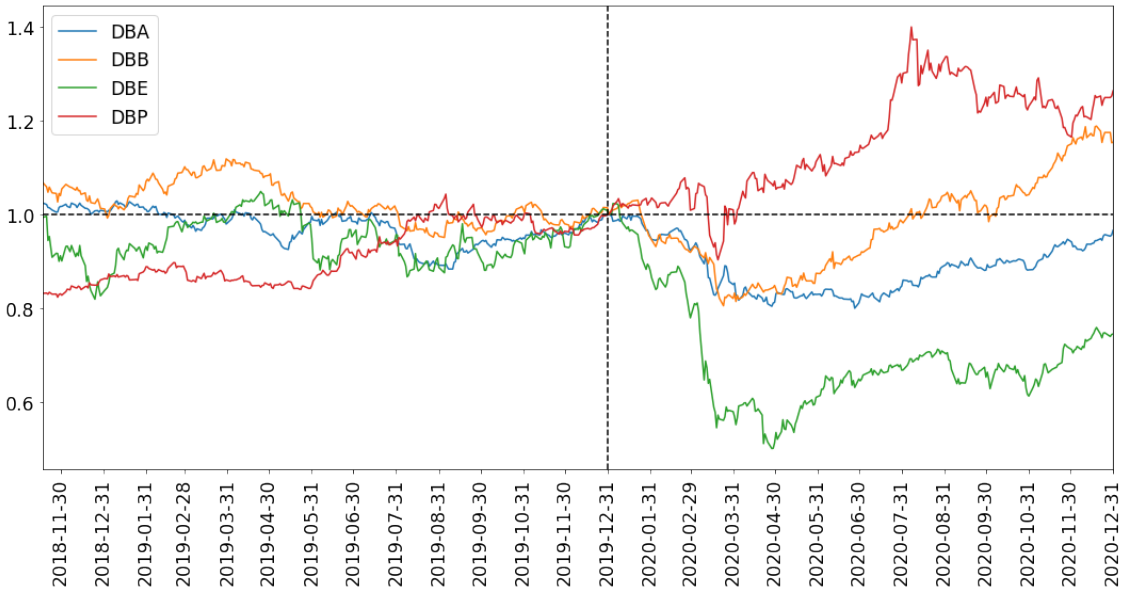

Figure 29: Commodity ETPs' Returns on a 1 U.S. Dollar Investment from End 2019 until End 2020.

B QuantConnect Code

```
# ------------------------------------------------------------------
# This code was written by Francisco Jose Manjon Cabeza Garcia to
# ilustrate the behaviour of different portfolio allocation methods
# for his bachelor thesis at the TU Berlin.
# ------------------------------------------------------------------

import numpy as np
import pandas as pd
import scipy.cluster.hierarchy as sch
from scipy.linalg import solve as linsolver
from sklearn.cluster import KMeans

class MarkowitzMinimumVolatilityInstabilityAnalyses(QCAlgorithm):

    def Initialize(self):

        # -----------------
        # Input Parameters.
        # -----------------

        # Start date of the backtesting.
        self.SetStartDate(2020, 1, 1)
        # End date of the backtesting.
        self.SetEndDate(2020, 12, 31)
        # Initial account balance.
        self.SetCash(1000000)
        # Benchmark for backtesting.
        self.SetBenchmark("SPY")

        # Optimisation model for the portfolio construction. Available options:
        #   - "Equal Weight"
        #   - "Inverse Variance"
        #   - "Markowitz Minimum Volatility"
        #   - "Hierarchical Risk Parity"
        self.OptimisationModel = "Hierarchical Risk Parity"

        # Covariance shrinkage method used in the optimisation model.
        # This parameter only affects the "Markowitz Minimum Volatility" optimisation model.
```

```
# Available options:
#   - None              ->  Do not shrink covariance matrix.
#   - "Ledoit-Wolf"     ->  Use Ledoit and Wolf constant correlation model.
#   - "De Nard"         ->  Use De Nard's generalised constant-variance-covariance model.
self.CovarianceShrinkageMethod = None

# Number of groups in which assets are clustered
# This parameter only affects "De Nard" covariance shrinkage method.
self.AssetGroupCount = 4

# Normalise the allocation vectors so that the portfolio weights sum up to 1.
NormaliseAllocationVector = True

# Number of past days' returns data to consider for the portfolio construction.
LookbackPeriods = 60

# New York time of the first day in the month at which the portfolio is rebalanced.
RebalancingTime = "10:00"

# Set the first month to start trading.
# If set to 0, enter the first portfolio on the first day of the backtesting.
FirstTradingMonth = 0

# List with the assets to consider in the portfolio.
SelectedAssets = [
            # Equity ETFs
            "EXI",      # iShares Global Industrials ETF
            "FILL",     # iShares MSCI Global Energy Producers ETF
            "IGF",      # iShares Global Infrastructure ETF
            "IXG",      # iShares Global Financials ETF
            "IXJ",      # iShares Global Healthcare ETF
            "IXN",      # iShares Global Tech ETF
            "IXP",      # iShares Global Comm Services ETF
            "JXI",      # iShares Global Utilities ETF
            "KXI",      # iShares Global Consumer Staples ETF
            "MXI",      # iShares Global Materials ETF
            "RXI",      # iShares Global Consumer Discretionary ETF
            # Fixed Income ETFs
            "IAGG",     # iShares Core International Aggregate Bond ETF
            "ILTB",     # iShares Core 10+ Year USD Bond ETF
            "IMTB",     # iShares Core 5-10 Year USD Bond ETF
            "ISTB",     # iShares Core 1-5 Year USD Bond ETF
            # Real Estate
            "REET",     # iShares Global REIT ETF
            # Commodities
            "DBA",      # Invesco DB Agriculture Fund
            "DBB",      # Invesco DB Base Metals Fund
            "DBE",      # Invesco DB Energy Fund
            "DBP"       # Invesco DB Precious Metals Fund
            ]
# ------------------------
# End of Input Parameters.
# ------------------------

# Set the internal clock of the algorithm to the New York timezone.
self.SetTimeZone("America/New_York")

# Create a list with QC Symbol objects of the selected assets.
QCSymbols = [Symbol.Create(asset, SecurityType.Equity, Market.USA) for asset in SelectedAssets]
# Set the data resolution for all instruments in the algorithm to 1-minute resolution.
# The portfolio construction model will so be triggered at the exact time chosen.
self.UniverseSettings.Resolution = Resolution.Minute
# Pass a custom initialiser for the instruments in the algorithm.
# This allows to adjust the leverage and the normalisation mode of each one.
self.SetSecurityInitializer(self.CustomSecurityInitializer)
# Set the universe selection model to manual, and add the symbols of the selected assets.
self.AddUniverseSelection(ManualUniverseSelectionModel(QCSymbols))
# Turn off margin calls.
# This is needed to be able to simulate the theoretical Markowitz' minimum volatility portfolios.
self.Portfolio.MarginCallModel = MarginCallModel.Null
```

```
        # Set the alpha model to generate constant insights the first day of each month.
        self.SetAlpha(AtMonthStartAlphaModel(RebalancingTime, FirstTradingMonth))

        # Set the portfolio construction model to the chosen optimisation method.
        if self.OptimisationModel == "Equal Weight":
            self.SetPortfolioConstruction(EqualWeightPortfolio())

        elif self.OptimisationModel == "Inverse Variance":
            self.SetPortfolioConstruction(InverseVariancePortfolio(LookbackPeriods,
                                                                   NormaliseAllocationVector
                                                                   ))

        elif self.OptimisationModel == "Markowitz Minimum Volatility":
            self.SetPortfolioConstruction(MarkowitzMinVolPortfolio(self.CovarianceShrinkageMethod,
                                                                   self.AssetGroupCount,
                                                                   LookbackPeriods,
                                                                   NormaliseAllocationVector
                                                                   ))

        elif self.OptimisationModel == "Hierarchical Risk Parity":
            self.SetPortfolioConstruction(HierarchicalRiskParityPortfolio(LookbackPeriods))

        # Set the risk management model to the null model that does nothing.
        self.SetRiskManagement(NullRiskManagementModel())

        # Set the execution model to a custom model that optimises the margin used when rebalancing.
        self.SetExecution(MarginOptimisedExecutionModel())

        # Define global variables to save data generated by the portfolio optimisation method.
        # All but the turnover figures and asset clusterings are saved indexed by symbol
        # alphabetically sorted.
        global CovarianceMatricesExport
        CovarianceMatricesExport = dict()
        global AllocationVectorsExport
        AllocationVectorsExport = dict()
        global TurnoverFiguresExport
        TurnoverFiguresExport = dict()
        global ShrinkageConstantsExport
        ShrinkageConstantsExport = dict()
        global ShrinkageTargetsExport
        ShrinkageTargetsExport = dict()
        global AssetClusteringsExport
        AssetClusteringsExport = dict()

    def OnData(self, data):
        ''' OnData event is the primary entry point for your algorithm.
        Each new data point will be pumped in here.
            Arguments:
                data: Slice object keyed by symbol containing the stock data
        '''
        pass

    def CustomSecurityInitializer(self, security):
        '''Custom security initialiser to set the leverage of each security higher
        to allow portfolios with individual allocations higher than 2 to be simulated.
        Moreover, the data normalisation mode is set to total return to account for
        ETP distributions.
        Arguments:
            security: security object of the instrument to initialise.
        '''
        security.SetLeverage(3)
        security.SetDataNormalizationMode(DataNormalizationMode.TotalReturn)

    def OnOrderEvent(self, orderEvent):
        '''This method is executed every time an order is sent in order to log its details.
        Arguments:
            orderEvent: OrderEvent object of the new order sent.
        '''

        order = self.Transactions.GetOrderById(orderEvent.OrderId)
        if orderEvent.Status == OrderStatus.Filled:
```

```
            self.Log("{0} - {1}: {2}".format(self.Time, order.Type, orderEvent))

    def OnEndOfAlgorithm(self):
        '''This method is executed at the very end of the backtesting period
        in order to liquidate all open positions and save the data generated
        by the optimisation methods in the ObjectStore.
        '''

        # Close all open positions
        self.Liquidate()

        modelname = ""

        if self.OptimisationModel == "Equal Weight":
            modelname = "EWP"

        elif self.OptimisationModel == "Inverse Variance":
            modelname = "IVP"

        elif self.OptimisationModel == "Markowitz Minimum Volatility":
            modelname = "MMVP"

            if self.CovarianceShrinkageMethod == "Ledoit-Wolf":
                modelname = modelname + "-LW"

            if self.CovarianceShrinkageMethod == "De Nard":
                modelname = modelname + "-DN-K" + str(self.AssetGroupCount)

        elif self.OptimisationModel == "Hierarchical Risk Parity":
            modelname = "HRP"

        # If covariance matrices were generated, save them.
        if CovarianceMatricesExport:
            covmatrices = pd.DataFrame.from_dict(CovarianceMatricesExport, orient="index")
            self.ObjectStore.Save(modelname+" Covariance Matrices", covmatrices.to_json())

        # If allocation vectors were generated, save them.
        if AllocationVectorsExport:
            allovectors = pd.DataFrame.from_dict(AllocationVectorsExport, orient="index")
            self.ObjectStore.Save(modelname+" Allocation Vectors", allovectors.to_json())

        # If turnover figures were generated, save them.
        if TurnoverFiguresExport:
            tovector = pd.DataFrame.from_dict(TurnoverFiguresExport, orient="index")
            self.ObjectStore.Save(modelname+" Turnover", tovector.to_json())

        # If shrinkage constants were generated, save them.
        if ShrinkageConstantsExport:
            scvector = pd.DataFrame.from_dict(ShrinkageConstantsExport, orient="index")
            self.ObjectStore.Save(modelname+" Shrinkage Constants", scvector.to_json())

        # If shrinkage targets were generated, save them.
        if ShrinkageTargetsExport:
            stvectors = pd.DataFrame.from_dict(ShrinkageTargetsExport, orient="index")
            self.ObjectStore.Save(modelname+" Shrinkage Targets", stvectors.to_json())

        # If clusterings were generated, save them.
        if AssetClusteringsExport:
            clusterings = pd.DataFrame.from_dict(AssetClusteringsExport, orient="index")
            self.ObjectStore.Save(modelname+" Clusterings", clusterings.to_json())

# -------------------------------
# Alpha insights generator model.
# -------------------------------

class AtMonthStartAlphaModel(AlphaModel):
    '''Custom AlphaModel class that emits constant insights of type price, upwards direction
    and 1 month duration on the first day of each month at the chosen rebalancing time.
    '''

    def __init__(self, rebalancingTime, currentMonth=0):
```

```
        # List of symbols for which insights are to be emitted.
        self.symbols = []
        # Last month for which insights were emitted.
        self.lastInsightMonth = currentMonth
        # Hour of the day at which to emit insights.
        self.rebalancingHour = int(rebalancingTime.split(":")[0])
        # Minute of the hour at which to emit insights.
        self.rebalancingMinute = int(rebalancingTime.split(":")[1])

    def OnSecuritiesChanged(self, algorithm, changes):
        '''This method is executed whenever securities are added or removed from the
        algorithm's universe. Update "self.symbols" accordingly if this happens.
        Arguments:
            algorithm: algorithm object that calls this method.
            changes: list of changed securities.
        '''

        for security in changes.AddedSecurities:
            if security.Symbol not in self.symbols:
                self.symbols.append(security.Symbol)

        for security in changes.RemovedSecurities:
            if security.Symbol in self.symbols:
                self.symbols.remove(security.Symbol)

    def Update(self, algorithm, data):
        '''This method is executed on every new piece of data that arrives. If it's
        rebalancing time, emit insights for each of the symbols in "self.symbols".
        Arguments:
            algorithm: algorithm object that calls this method.
            data: new piece of data that arrived.
        '''

        if len(self.symbols) > 1:

            currentTime = algorithm.Time

            if currentTime.month != self.lastInsightMonth:
                # It is the first day of a new month.
                if (currentTime.hour == self.rebalancingHour
                    and currentTime.minute == self.rebalancingMinute):

                    # New rebalancing point.
                    algorithm.Log("New rebalancing point.")
                    self.lastInsightMonth = currentTime.month
                    # Emit "up" insights.
                    return Insight.Group([Insight.Price(symbol,
                                                        timedelta(minutes=1),
                                                        InsightDirection.Up)
                                          for symbol in self.symbols])

        return []

# ------------------------------
# Portfolio construction models.
# ------------------------------

class EqualWeightPortfolio(PortfolioConstructionModel):
    '''PortfolioConstructionModel object that computes the equal weight portfolio.
    '''

    def CreateTargets(self, algorithm, insights):
        '''This method is executed every time new insights are emitted by the
        "AtMonthStartAlphaModel" object. If new insights are emitted, compute
        equal weight portfolio for the symbols of the insights.
        Arguments:
            algorithm: algorithm object that calls this method.
            insights: list of insights emitted by the "AtMonthStartAlphaModel" object.
        '''

        if len(insights) > 0:
```

```
            algorithm.Log("Computing equal weight portfolio allocation...")

            # Compute equal weight portfolio.
            numAssets = len(insights)
            targets = [PortfolioTarget.Percent(algorithm, insight.Symbol, 1./numAssets)
                       for insight in insights]

            # Save allocation vector.
            AllocationVectorsExport[algorithm.Time] = np.full(numAssets, 1./numAssets)

            # Return set of targets for the MarginOptimisedExecutionModel object.
            return targets

        return []

class InverseVariancePortfolio(PortfolioConstructionModel):
    '''PortfolioConstructionModel object that computes the inverse variance portfolio.
    '''

    def __init__(self, lookbackPeriods, normalise=True):
        # Number of past days' returns data to consider for the portfolio construction.
        self.LookbackPeriods = lookbackPeriods
        # Normalise the allocation vectors so that the portfolio weights sum up to 1.
        self.NormaliseAllocationVector = normalise

    def CreateTargets(self, algorithm, insights):
        '''This method is executed every time new insights are emitted by the
        "AtMonthStartAlphaModel" object. If new insights are emitted, compute
        inverse variance portfolio for the symbols of the insights.
        Arguments:
            algorithm: algorithm object that calls this method.
            insights: list of insights emitted by the "AtMonthStartAlphaModel" object.
        '''

        if len(insights) > 0:
            algorithm.Log("Computing inverse variance portfolio allocation...")

            # Get the past returns of the symbols for which insights were received.
            returnsData = dict()

            for symbol in [insight.Symbol for insight in insights]:
                symbolReturns = algorithm.History(symbol,
                                                  self.LookbackPeriods,
                                                  Resolution.Daily).loc[symbol, ["close", "open"]]
                symbolReturns = symbolReturns.apply(lambda x: x.loc["close"] / x.loc["open"] - 1,
                                                    axis=1)
                returnsData[str(symbol)] = symbolReturns
            # Create pandas DataFrame with the symbol names as index sorted alphabetically
            # and the returns of each of the "self.LookbackPeriods" days as columns.
            returnsData = pd.DataFrame.from_dict(returnsData, orient="index").sort_index()

            # Compute optimal portfolio according to the model self.OptimisationModel
            varmatrix = returnsData.var(axis=1)
            weights = 1 / varmatrix
            if self.NormaliseAllocationVector:
                weights = weights / weights.sum()

            targets = [PortfolioTarget.Percent(algorithm, symbol, weights[symbol])
                       for symbol in weights.index]

            # Save allocation vector.
            AllocationVectorsExport[algorithm.Time] = weights
            # Save variance matrix as covariance matrix.
            # All assets assumed to be uncorrelated.
            CovarianceMatricesExport[algorithm.Time] = varmatrix.to_numpy()

            # Return set of targets for the MarginOptimisedExecutionModel object.
            return targets

        return []
```

```
class MarkowitzMinVolPortfolio(PortfolioConstructionModel):
    '''PortfolioConstructionModel object that computes Markowitz Minimum Volatility portfolio.
    '''

    def __init__(self, covmatrixshrinkagemode, assetgroupcount, lookbackPeriods, normalise=True):
        # Covariance shrinkage method used in the optimisation model.
        # This parameter only affects the "Markowitz Minimum Volatility" optimisation model.
        # Available options:
        #   - None              ->  Do not shrink covariance matrix.
        #   - "Ledoit-Wolf"     ->  Use Ledoit and Wolf constant correlation model.
        #   - "De Nard"         ->  Use De Nard's generalised constant-variance-covariance model.
        self.CovarianceShrinkageMethod = covmatrixshrinkagemode
        # Number of groups in which assets are clustered
        # This parameter only affects "De Nard" covariance shrinkage method.
        self.AssetGroupCount = assetgroupcount
        # Number of past days' returns data to consider for the portfolio construction.
        self.LookbackPeriods = lookbackPeriods
        # Normalise the allocation vectors so that the portfolio weights sum up to 1.
        self.NormaliseAllocationVector = normalise

    def CreateTargets(self, algorithm, insights):
        '''This method is executed every time new insights are emitted by the
        "AtMonthStartAlphaModel" object. If new insights are emitted, compute
        Markowitz Minimum Volatility portfolio for the symbols of the insights.
        Arguments:
            algorithm: algorithm object that calls this method.
            insights: list of insights emitted by the "AtMonthStartAlphaModel" object.
        '''

        if len(insights) > 0:
            algorithm.Log("Computing Markowitz minimum volatility portfolio allocation...")

            # Get the past returns of the symbols for which insights were received.
            returnsData = dict()

            for symbol in [insight.Symbol for insight in insights]:
                symbolReturns = algorithm.History(symbol,
                                                  self.LookbackPeriods,
                                                  Resolution.Daily).loc[symbol, ["close", "open"]]
                symbolReturns = symbolReturns.apply(lambda x: x.loc["close"] / x.loc["open"] - 1,
                                                    axis=1)
                returnsData[str(symbol)] = symbolReturns
            # Create pandas DataFrame with the symbol names as index sorted alphabetically
            # and the returns of each of the "self.LookbackPeriods" days as columns.
            returnsData = pd.DataFrame.from_dict(returnsData, orient="index").sort_index()

            # Compute Markowitz minimum variance portfolio.

            # Compute (shrunk) covariance matrix.
            if self.CovarianceShrinkageMethod == "Ledoit-Wolf":
                shrinkageconstant, shrinkagetarget, covmatrix = self.LedoitWolfCovShrink(algorithm, returnsData)

            elif self.CovarianceShrinkageMethod == "De Nard":
                shrinkageconstant, shrinkagetarget, covmatrix = self.DeNardCovShrink(algorithm, returnsData)

            else:
                # We compute the sample covariance matrix after Bessel's correction factor.
                covmatrix = np.cov(returnsData, rowvar=True)

            numAssets = covmatrix.shape[0]

            # Try to solve the linear system using the Cholesky decomposition of the covariance matrix.
            try:
                weights = linsolver(covmatrix, np.ones(numAssets), assume_a="pos")

            # If the Cholesky decomposition fails, the covariance matrix does not have full rank.
            # Solve the linear system with the least-squares method.
            except:
                algorithm.Log("The covariance matrix is singular.")
                algorithm.Log("Solving linear system with the least-squares method...")
                weights, res, rnk, s = np.linalg.lstsq(covmatrix, np.ones(numAssets))
```

```
            if self.NormaliseAllocationVector:
                weights = weights / weights.sum()

            # Generate Target objects.
            targets = []
            i = 0
            for symbol in returnsData.index:
                targets.append(PortfolioTarget.Percent(algorithm, symbol, weights[i]))
                i = i + 1

            # Save allocation vector.
            AllocationVectorsExport[algorithm.Time] = weights
            # Save covariance matrix as vector.
            CovarianceMatricesExport[algorithm.Time] = covmatrix.reshape(numAssets**2)
            # Save shrinkage constant and shrinkage target as vectors.
            if self.CovarianceShrinkageMethod != None:
                ShrinkageConstantsExport[algorithm.Time] = shrinkageconstant
                ShrinkageTargetsExport[algorithm.Time] = shrinkagetarget.reshape(numAssets**2)

            # Return set of targets for the MarginOptimisedExecutionModel object.
            return targets

        return []

    def LedoitWolfCovShrink(self, algorithm, returnsData):
        '''This method is computes the constant correlation matrix and the optimal
        shrinkage constant as Ledoit and Wolf describe in their paper
        Honey, I Shrunk the Sample Covariance Matrix. It returns, the shrinkage
        constant, the constant correlation matrix and the srhunk sample covariance
        matrix.
        Arguments:
            algorithm: algorithm object that calls this method.
            returnsData: pandas DataFrame with the past period observations of each
                         asset as rows.
        '''

        returnsData = returnsData.to_numpy()

        # "returnsData" contains the returns of each symbol in rows.
        numAssets, T = returnsData.shape

        # Compute centered returns.
        x = (returnsData.T - returnsData.mean(axis=1)).T

        # Compute sample covariance, variance and standard deviation matrices.
        samplecovmatrix = np.cov(returnsData, rowvar=True)
        varmatrix = np.diag(np.diag(samplecovmatrix))
        stdmatrix = np.sqrt(varmatrix)
        stdmatrixinv = np.diag(1/np.diag(stdmatrix))
        corrmatrix = np.dot(stdmatrixinv, np.dot(samplecovmatrix, stdmatrixinv))

        # Compute the constant correlation matrix corresponding to the sample cov. matrix.
        rbar = (corrmatrix.sum() - numAssets) / ((numAssets-1) * numAssets)
        constcorrmatrix = np.full((numAssets, numAssets), rbar)
        constcorrmatrix = np.dot(stdmatrix, np.dot(constcorrmatrix, stdmatrix))
        np.fill_diagonal(constcorrmatrix, np.diag(varmatrix))

        # Compute pi hat.
        y = np.power(x, 2)
        piMat = np.dot(y, y.T) / T - np.power(samplecovmatrix, 2)
        pihat = piMat.sum()

        # Compute rho hat.
        term1 = np.dot(np.power(returnsData, 3), returnsData.T) / T
        term2 = np.dot(varmatrix, samplecovmatrix)
        term3 = np.dot(samplecovmatrix, varmatrix)
        term4 = np.dot(varmatrix, samplecovmatrix)
        thetaMat = term1 - term2 - term3 + term4
        np.fill_diagonal(thetaMat, 0)
        rhohat = np.diag(piMat).sum() + rbar * np.multiply(np.outer(np.diag(stdmatrixinv),
```

```
                                                        np.diag(stdmatrix)),
                                                thetaMat).sum()

        # Compute gamma hat.
        gammahat = np.linalg.norm(constcorrmatrix - samplecovmatrix, "fro") ** 2

        # Compute shrinkage constant.
        delta = 1/T * (pihat - rhohat) / gammahat

        if delta <= 0:
            algorithm.Log("Shrinkage constant is negative or zero.")
            return 0, constcorrmatrix, samplecovmatrix

        elif delta >= 1:
            algorithm.Log("Shrinkage constant is greater than 1.")
            return 1, constcorrmatrix, constcorrmatrix

        return delta, constcorrmatrix, (delta * constcorrmatrix + (1-delta) * samplecovmatrix)

    def DeNardCovShrink(self, algorithm, returnsData):
        '''This method is computes the generalised constant-variance-covariance
        matrix and the optimal shrinkage constant as De Nard describes in his paper
        Opps! I Shrunk the Sample Covariance Matrix Again: Blockbuster Meets Shrinkage,
        using a fixed number of groups for the Blockbuster clustering algorithm. It
        returns, the shrinkage constant, the generalised constant-variance-covariance
        matrix and the srhunk sample covariance matrix.
        Arguments:
            algorithm: algorithm object that calls this method.
            returnsData: pandas DataFrame with the past period observations of each
                         asset as rows.
        '''

        returnsDataIndex = returnsData.index
        returnsData = returnsData.to_numpy()

        # "returnsData" contains the returns of each symbol in rows.
        numAssets, T = returnsData.shape

        # Compute centered returns.
        x = (returnsData.T - returnsData.mean(axis=1)).T

        # Compute sample covariance, variance and standard deviation matrices.
        samplecovmatrix = np.cov(returnsData, rowvar=True)
        varmatrix = np.diag(np.diag(samplecovmatrix))
        stdmatrix = np.sqrt(varmatrix)
        stdmatrixinv = np.diag(1/np.diag(stdmatrix))
        corrmatrix = np.dot(stdmatrixinv, np.dot(samplecovmatrix, stdmatrixinv))

        '''
        Blockbuster Clustering Algorithm.
        '''
        # Compute the eigenvalues and normalised eigenvectors of the sample covariance matrix.
        eigenvals, eigenvects = np.linalg.eig(samplecovmatrix)
        # Select the normalised eigenvectors corresponding to the K highest eigenvalues.
        ind = np.argpartition(eigenvals, -self.AssetGroupCount)[-self.AssetGroupCount:]
        selectedEV = eigenvects[:, ind]
        # Normalise the matrix with the K highest eigenvectors as columns row-wise.
        for i in range(selectedEV.shape[0]):
            selectedEV[i,:] = selectedEV[i,:] / np.linalg.norm(selectedEV[i,:], ord=2)
        # Apply K-means to the rows of the matrix that contains the K selected eigenvectors as rows.
        kmeans = KMeans(n_clusters=self.AssetGroupCount, random_state=100).fit(selectedEV)

        # Sort the rows and columns of the sample covariance matrix,
        # so that the block covariance matrix between the assets in the same group lies on the diagonal.
        groupedAssets = pd.DataFrame(index=returnsDataIndex,
                                     data=kmeans.labels_,
                                     columns=["Group"]).sort_values(by="Group")
        # Save the asset clustering for later export.
        AssetClusteringsExport[algorithm.Time] = groupedAssets["Group"]
        pdSampleCovMatrix = pd.DataFrame(data=samplecovmatrix,
                                         index=returnsDataIndex,
```

```
                                    columns=returnsDataIndex)
pdSampleCovMatrix = pdSampleCovMatrix.reindex(index=groupedAssets.index,
                                              columns=groupedAssets.index)

'''
Generalised Constant-Variance-Covariance shrinkage target.
'''

PhiGCVC = []
for i in range(self.AssetGroupCount):
    ithBlockRow = []
    for j in range(self.AssetGroupCount):
        currentCovMatrixBlock = pdSampleCovMatrix.loc[(groupedAssets.iloc[:,0] == i),
                                                      (groupedAssets.iloc[:,0] == j)]
        if i == j:
            ijPhiGCVCBlock = np.full(shape=currentCovMatrixBlock.shape,
                                     fill_value=currentCovMatrixBlock.mean())
            np.fill_diagonal(ijPhiGCVCBlock, np.diagonal(currentCovMatrixBlock).mean())

        else:
            ijPhiGCVCBlock = np.full(shape=currentCovMatrixBlock.shape,
                                     fill_value=currentCovMatrixBlock.mean())

        ithBlockRow.append(ijPhiGCVCBlock)

    PhiGCVC.append(ithBlockRow)

PhiGCVC = np.block(PhiGCVC)
PhiGCVC = pd.DataFrame(data=PhiGCVC,
                       index=pdSampleCovMatrix.index,
                       columns=pdSampleCovMatrix.index)
PhiGCVC = PhiGCVC.reindex(index=returnsDataIndex,
                          columns=returnsDataIndex).to_numpy()

'''
Compute delta shrinkage intensity.
'''

# Compute r bar.
rbar = (corrmatrix.sum() - numAssets) / ((numAssets-1) * numAssets)

# Compute pi hat.
y = np.power(x, 2)
piMat = np.dot(y, y.T) / T - np.power(samplecovmatrix, 2)
pihat = piMat.sum()

# Compute rho hat.
term1 = np.dot(np.power(returnsData, 3), returnsData.T) / T
term2 = np.dot(varmatrix, samplecovmatrix)
term3 = np.dot(samplecovmatrix, varmatrix)
term4 = np.dot(varmatrix, samplecovmatrix)
thetaMat = term1 - term2 - term3 + term4
np.fill_diagonal(thetaMat, 0)
rhohat = np.diag(piMat).sum() + rbar * np.multiply(np.outer(np.diag(stdmatrixinv),
                                                            np.diag(stdmatrix)),
                                                   thetaMat).sum()

# Compute gamma hat.
gammahat = np.linalg.norm(PhiGCVC - samplecovmatrix, "fro") ** 2

# Compute shrinkage constant.
delta = 1/T * (pihat - rhohat) / gammahat

if delta <= 0:
    algorithm.Log("Shrinkage constant is negative or zero.")
    return 0, PhiGCVC, samplecovmatrix

elif delta >= 1:
    algorithm.Log("Shrinkage constant is greater than 1.")
    return 1, PhiGCVC, PhiGCVC
```

```
        return delta, PhiGCVC, (delta * PhiGCVC + (1-delta) * samplecovmatrix)

class HierarchicalRiskParityPortfolio(PortfolioConstructionModel):
    '''PortfolioConstructionModel object that computes the Hierarchical Risk Parity portfolio.
    '''

    def __init__(self, lookbackPeriods):
        # Number of past days' returns data to consider for the portfolio construction.
        self.LookbackPeriods = lookbackPeriods

    def CreateTargets(self, algorithm, insights):
        '''This method is executed every time new insights are emitted by the
        "AtMonthStartAlphaModel" object. If new insights are emitted, compute
        Hierarchical Risk Parity portfolio for the symbols of the insights.
        Arguments:
            algorithm: algorithm object that calls this method.
            insights: list of insights emitted by the "AtMonthStartAlphaModel" object.
        '''

        if len(insights) > 0:
            algorithm.Log("Computing Hierarchical Risk Parity portfolio allocation...")

            # Get the past returns of the symbols for which insights were received.
            returnsData = dict()

            for symbol in [insight.Symbol for insight in insights]:
                symbolReturns = algorithm.History(symbol,
                                                  self.LookbackPeriods,
                                                  Resolution.Daily).loc[symbol, ["close", "open"]]
                symbolReturns = symbolReturns.apply(lambda x: x.loc["close"] / x.loc["open"] - 1,
                                                    axis=1)
                returnsData[str(symbol)] = symbolReturns
            # Create pandas DataFrame with the symbol names as index sorted alphabetically
            # and the returns of each of the "self.LookbackPeriods" days as columns.
            returnsData = pd.DataFrame.from_dict(returnsData, orient="index").sort_index()

            # Compute hierarchical risk parity portfolio.
            # Code from Advances in Financial Machine Learning, Marcos López de Prado.
            # Small changes to the code from pages 240-242 applied to fit QuantConnect's API
            # and the general implementation structure of the program.

            # pandas cov() and corr() method compute the covariance and correlation of the columns.
            cov, corr = returnsData.T.cov(), returnsData.T.corr()

            # 3) cluster
            # Compute distance matrix based on correlation, where 0 <= d[i,j] <= 1.
            dist = ((1 - corr) / 2.)**.5
            link = sch.linkage(dist, "single")
            # Save the hierarchical structure to later export.
            AssetClusteringsExport[algorithm.Time] = link.reshape(link.size)
            # Sort clustered items by distance.
            link = link.astype(int)
            sortIx = pd.Series([link[-1,0], link[-1,1]])
            numItems= link[-1, 3] # Number of original items.

            while sortIx.max() >= numItems:
                sortIx.index = range(0, sortIx.shape[0]*2, 2) # Make space.
                df0 = sortIx[sortIx >= numItems] # Find clusters.
                i = df0.index
                j = df0.values - numItems
                sortIx[i] = link[j, 0] # Item 1.
                df0 = pd.Series(link[j, 1], index=i+1)
                sortIx = sortIx.append(df0) # Item 2.
                sortIx = sortIx.sort_index() # Re-sort.
                sortIx.index = range(sortIx.shape[0]) # Re-index
            sortIx = sortIx.tolist()
            sortIx = corr.index[sortIx].tolist() # Recover labels.
            #df0 = corr.loc[sortIx, sortIx] # Reorder.

            #4) Capital allocation
            # Compute HRP alloc
```

```
            w = pd.Series(1, index=sortIx)
            cItems = [sortIx] # Initialise all items in one cluster
            while len(cItems) > 0:
                cItems = [i[int(j):int(k)] for i in cItems for j, k in ((0, len(i) / 2),
                                                        (len(i) / 2, len(i))) if len(i) > 1] # Bi-section.

                for i in range(0, len(cItems), 2): # Parse in pairs.
                    cItems0 = cItems[i] # Cluster 1
                    cItems1 = cItems[i+1] # Cluster 2
                    # Compute variance per cluster.
                    cov0_ = cov.loc[cItems0, cItems0] # Matrix slice
                    w0_ = 1. / np.diag(cov0_)
                    w0_ = w0_ / w0_.sum()
                    w0_ = w0_.reshape(-1, 1)
                    cVar0 = np.dot(np.dot(w0_.T, cov0_), w0_)[0, 0]

                    cov1_ = cov.loc[cItems1, cItems1] # Matrix slice
                    w1_ = 1. / np.diag(cov1_)
                    w1_ = w1_ / w1_.sum()
                    w1_ = w1_.reshape(-1, 1)
                    cVar1 = np.dot(np.dot(w1_.T, cov1_), w1_)[0, 0]

                    alpha = 1 - cVar0 / (cVar0+cVar1)
                    w[cItems0] *= alpha # Weight 1
                    w[cItems1] *= 1 - alpha

            # Save allocation vector.
            AllocationVectorsExport[algorithm.Time] = w

            return [PortfolioTarget.Percent(algorithm, symbol, w[symbol]) for symbol in w.index]

        return []

# ------------------------
# Target execution model.
# ------------------------

class MarginOptimisedExecutionModel(ExecutionModel):
    '''Custom ExecutionModel class that optimises used margin on rebalancing by first
    rebalancing positions which will be reduced and then rebalancing positions which
    will increased.
    '''

    def Execute(self, algorithm, targets):

        # If there are targets, start rebalancing.
        if len(targets) > 0:

            algorithm.Log("Rebalancing portfolio...")

            # Compute the change in allocation to each symbol according to the new targets.
            orderQuantities = []

            for target in targets:
                open_quantity = sum([x.Quantity for x in algorithm.Transactions.GetOpenOrders(target.Symbol)])
                existing = algorithm.Securities[target.Symbol].Holdings.Quantity + open_quantity
                if target.Quantity - existing != 0:
                    orderQuantities.append((target.Symbol, target.Quantity - existing))

            # Sort the changes in allocation.
            sortedOrderQuantities = sorted(orderQuantities, key=lambda target: target[1])

            # Save the absolute change in allocation to compute the period's turnover.
            periodTurnover = 0

            # Rebalance positions by first tackling the ones that are reduced and the the ones that need to be increased.
            for (symbol, quantity) in sortedOrderQuantities:
                algorithm.MarketOrder(symbol, quantity)
                # Save the absolute change in allocation.
                periodTurnover = periodTurnover + np.abs(quantity)
```

```
        # Compute and save the relative change in allocation (turnover).
        TurnoverFiguresExport[algorithm.Time] = periodTurnover / algorithm.Portfolio.TotalPortfolioValue

    pass
```

C Equal Weighted Portfolios

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
DBB	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
DBE	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
DBP	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
EXI	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
FILL	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IAGG	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IGF	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
ILTB	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IMTB	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
ISTB	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IXG	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IXJ	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IXN	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IXP	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
JXI	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
KXI	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
MXI	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
REET	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
RXI	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Table 6: Equal Weight Portfolio Allocation over Time.

D Inverse Variance Portfolios

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	0.008213	0.005744	0.006305	0.050581	0.035906	0.033183	0.011031	0.008205	0.008647	0.006605	0.007275	0.006758
DBB	0.009619	0.006957	0.007222	0.067742	0.064526	0.048473	0.012317	0.006411	0.007779	0.004829	0.004512	0.003763
DBE	0.002294	0.002288	0.002756	0.015817	0.005898	0.005104	0.001094	0.001620	0.002129	0.001652	0.001582	0.001234
DBP	0.012815	0.009605	0.003375	0.015726	0.016118	0.016197	0.009010	0.004698	0.002207	0.001592	0.001947	0.003508
EXI	0.012999	0.008151	0.004100	0.013423	0.012562	0.012378	0.003427	0.001850	0.002032	0.002748	0.002616	0.002321
FILL	0.005073	0.004804	0.001761	0.004738	0.004254	0.003996	0.001750	0.001186	0.001276	0.001418	0.001480	0.000970
IAGG	0.160477	0.157436	0.190366	0.367811	0.396495	0.402141	0.420680	0.403180	0.343728	0.197724	0.160375	0.229325
IGF	0.016445	0.013497	0.005062	0.007286	0.006766	0.006841	0.004201	0.002609	0.002614	0.003206	0.003414	0.003013
ILTB	0.020121	0.016432	0.013869	0.017612	0.017907	0.017337	0.021894	0.017030	0.013929	0.011500	0.012732	0.017463
IMTB	0.020562	0.022605	0.027394	0.133447	0.141488	0.156049	0.121500	0.060810	0.059868	0.068785	0.135189	0.092439
ISTB	0.616085	0.670006	0.701451	0.195889	0.195755	0.189923	0.348641	0.465899	0.527300	0.670472	0.640125	0.612358
IXG	0.015048	0.009836	0.004676	0.012068	0.010705	0.010230	0.003107	0.001800	0.002220	0.003326	0.003096	0.002775
IXJ	0.010049	0.007009	0.003917	0.012890	0.012055	0.013235	0.005864	0.003885	0.004437	0.004279	0.003866	0.003809
IXN	0.005634	0.004617	0.001597	0.007944	0.007161	0.008537	0.003171	0.002040	0.002219	0.001409	0.001510	0.001554
IXP	0.012678	0.007692	0.003259	0.015024	0.014245	0.016257	0.005638	0.002822	0.002652	0.002621	0.002581	0.002611
JXI	0.011809	0.010938	0.004671	0.006517	0.005956	0.006330	0.003980	0.003009	0.002837	0.003022	0.003337	0.003723
KXI	0.022056	0.015826	0.006860	0.012751	0.012842	0.014028	0.009850	0.005648	0.005813	0.005600	0.005799	0.004865
MXI	0.012977	0.011040	0.004405	0.017257	0.016373	0.016400	0.005461	0.002848	0.003017	0.003205	0.002947	0.002594
REET	0.009345	0.006168	0.003194	0.007123	0.006566	0.006216	0.002705	0.001684	0.002017	0.002170	0.001974	0.001749
RXI	0.015701	0.009351	0.003760	0.018355	0.016424	0.017145	0.004678	0.002767	0.003281	0.003839	0.003642	0.003168

Table 7: Inverse Variance Portfolio Allocation over Time.

E Markowitz Minimum Variance Portfolios

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	-0.010412	0.004127	0.005605	-0.025722	0.042397	0.027575	0.015961	-0.004147	0.005381	0.007559	0.001158	0.016859
DBB	0.028830	0.001349	0.002945	0.134976	0.166745	0.097344	0.009246	-0.014220	-0.020863	0.016467	0.018739	0.010583
DBE	0.010651	0.001874	0.012185	0.002177	0.004891	0.007550	0.016277	0.013160	-0.005914	0.002709	0.000290	0.002467
DBP	-0.003560	-0.015141	-0.004190	-0.018601	-0.024824	0.014299	0.016981	-0.003537	-0.001715	-0.002947	-0.007918	-0.009781
EXI	0.004185	-0.026375	-0.008179	-0.041597	-0.069568	-0.012156	-0.026504	-0.033481	-0.017642	-0.003141	-0.004910	-0.015243
FILL	-0.015498	0.004971	-0.008568	0.051082	0.026318	0.002142	0.000356	0.000749	0.003916	-0.002827	0.014960	0.002338
IAGG	0.143893	0.174706	0.205880	0.608069	0.573183	0.548158	0.562370	0.518946	0.521679	0.283210	0.226281	0.155022
IGF	-0.011677	0.021202	0.008161	-0.095099	-0.031942	-0.028851	0.052859	-0.038799	-0.032925	-0.056866	-0.068394	-0.047214
ILTB	-0.081290	-0.044517	-0.083109	-0.092544	-0.097551	-0.112250	-0.095372	-0.066026	-0.051458	-0.039092	-0.051375	-0.007722
IMTB	0.006962	-0.014508	-0.002515	0.083481	0.159789	0.301502	0.071919	0.077053	0.026364	0.094756	0.181133	0.124145
ISTB	0.855319	0.835723	0.826256	0.333283	0.220472	0.155188	0.416459	0.475384	0.509384	0.634608	0.625665	0.715744
IXG	0.062458	0.027310	0.018127	0.089345	0.051558	0.017431	0.032835	0.065455	0.039182	0.035775	-0.001182	0.016928
IXJ	-0.045437	-0.011241	-0.009869	-0.010897	0.038226	0.041023	-0.003227	0.002894	-0.015501	-0.018411	0.000747	-0.017751
IXN	0.001566	-0.021573	-0.011836	0.040287	0.039194	-0.010413	-0.007367	0.036894	0.023534	0.015932	0.012923	0.019120
IXP	0.057603	0.024404	0.006374	0.064847	0.007105	0.033705	0.010826	-0.005482	0.001501	-0.016500	-0.008260	-0.008112
JXI	0.009794	-0.013053	0.000644	-0.077601	-0.047317	-0.020973	-0.026124	0.040234	-0.005817	0.015617	0.023658	0.020429
KXI	0.001547	-0.027034	0.005755	0.112382	0.033109	0.020787	0.047839	-0.002493	0.038234	0.027269	-0.007203	-0.013729
MXI	0.021799	0.063534	0.042449	-0.035141	-0.042995	0.005443	-0.026241	-0.037459	-0.011654	0.011179	0.034468	0.056919
REET	0.002849	0.010492	-0.001255	-0.008140	0.014349	-0.005770	-0.000309	-0.006812	0.013432	0.011765	0.017782	0.020780
RXI	-0.039584	0.003749	-0.004860	-0.114588	-0.063141	-0.081734	-0.068786	-0.018314	-0.019117	-0.017061	-0.008562	-0.041781

Table 8: Markowitz Minimum Variance Portfolio Allocation over Time.

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	-0.010709	0.002346	-0.002969	-0.011960	0.021781	0.009190	0.005428	-0.006125	-0.006518	-0.004528	0.000694	0.010189
DBB	0.017974	-0.000504	-0.001001	0.126893	0.136736	0.090116	0.007922	-0.003349	-0.013315	0.012325	0.016283	0.010093
DBE	0.003822	0.000562	0.005335	-0.005075	0.001240	-0.000686	0.010529	0.006506	-0.004395	-0.001525	-0.001731	0.000272
DBP	-0.015572	-0.015801	-0.009438	-0.022999	-0.031713	-0.015374	-0.001637	-0.011125	-0.007558	-0.005289	-0.007113	-0.005693
EXI	0.010019	-0.004159	0.004283	-0.014246	-0.016265	-0.006056	-0.007111	-0.010863	-0.006583	-0.002512	-0.003080	-0.004187
FILL	-0.000481	0.007902	-0.002044	0.003713	0.003398	-0.006149	0.003401	-0.002368	-0.001793	-0.003401	0.004687	0.000563
IAGG	0.120720	0.148162	0.178266	0.596211	0.594583	0.613461	0.550854	0.496108	0.430990	0.189264	0.143340	0.174194
IGF	-0.003301	0.011475	0.005239	-0.033181	-0.024098	-0.020381	0.011764	-0.001208	-0.006630	-0.007901	-0.012897	-0.010100
ILTB	-0.049204	-0.037567	-0.053489	-0.080306	-0.084567	-0.085093	-0.073951	-0.062058	-0.050435	-0.034964	-0.043954	-0.030052
IMTB	0.003456	-0.009753	-0.003100	0.154517	0.169257	0.244772	0.087941	0.040958	0.010759	0.060165	0.149576	0.099337
ISTB	0.876758	0.862546	0.866023	0.282206	0.240973	0.197261	0.430159	0.541917	0.637045	0.783097	0.737557	0.745863
IXG	0.030518	0.010792	0.007724	0.005371	0.008351	-0.005251	0.008686	0.016050	0.010796	0.010879	0.001431	0.011498
IXJ	-0.015984	-0.007628	-0.000768	0.006191	0.010331	0.009096	-0.010578	-0.002430	-0.002228	0.001202	-0.001712	-0.010280
IXN	-0.001193	-0.008588	-0.004107	0.003579	0.006802	-0.004392	-0.007115	0.005238	0.001489	-0.000823	0.001907	0.004111
IXP	0.025888	0.009362	-0.001856	0.028650	0.012959	0.011937	0.000425	0.003404	0.004109	-0.009318	-0.003518	-0.004219
JXI	-0.008840	-0.006401	0.000603	-0.024184	-0.022436	-0.016639	-0.013855	0.001879	-0.010697	-0.002095	-0.001149	0.002981
KXI	0.004173	-0.008978	0.003531	0.004939	0.005747	0.005985	0.033610	0.002630	0.011294	0.009754	-0.000639	-0.007996
MXI	0.016508	0.038713	0.010164	0.002107	-0.009204	0.012365	-0.008639	-0.007322	-0.001214	0.007197	0.017240	0.022407
REET	-0.000926	0.003497	-0.002984	-0.014505	-0.008858	-0.017523	-0.009574	-0.004352	0.006117	0.001708	0.004668	0.006296
RXI	-0.003626	0.004019	0.000587	-0.007920	-0.015017	-0.016638	-0.018257	-0.003490	-0.001233	-0.003235	-0.001589	-0.015279

Table 9: Markowitz Minimum Variance Portfolios Computed Using Ledoit and Wolf's Covariance Matrix Shrinkage Method.

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	0.032811	0.074911	0.045240	0.192216	0.122726	0.120196	0.043577	0.013795	0.055816	0.051528	0.025567	0.016700
DBB	0.034267	0.048099	0.073946	0.080114	0.188400	0.161384	0.050316	-0.017734	0.041015	0.051158	0.021449	0.009343
DBE	-0.001807	-0.009283	0.025195	0.040569	0.016234	0.020109	0.016335	0.018363	-0.003967	0.035367	0.010314	0.006258
DBP	0.078851	0.096887	0.027217	0.064560	0.051172	0.061035	0.045289	-0.007857	0.028293	0.061835	0.003511	0.002103
EXI	-0.000309	-0.010486	0.019688	0.018627	-0.004510	-0.008370	-0.041522	-0.025087	-0.024646	0.054420	-0.007124	0.000969
FILL	0.021737	0.050274	-0.054366	-0.019386	-0.003055	-0.007827	0.017188	0.005666	0.040439	0.014077	0.020106	0.009607
IAGG	0.289481	0.161603	0.185080	0.119219	0.250609	0.262007	0.283301	0.757734	0.226918	0.193762	0.428044	0.319314
IGF	-0.008943	0.001718	0.014051	-0.073307	-0.077078	-0.078318	-0.012113	-0.013061	-0.094410	0.124577	-0.021287	-0.010942
ILTB	-0.019761	0.075141	0.127338	0.035383	0.015993	0.010985	0.053933	-0.023244	0.161840	0.142354	0.014640	0.026692
IMTB	0.052456	0.098129	0.150736	0.245802	0.131771	0.146700	0.223848	0.198604	0.214064	0.247841	0.035780	0.238809
ISTB	0.356468	0.216125	0.189507	0.115929	0.135023	0.140186	0.256900	0.043992	0.223400	0.203850	0.445492	0.380955
IXG	0.151394	0.172500	0.037880	0.038852	0.062131	0.041430	0.002772	0.046164	0.032511	0.109581	0.028855	0.007132
IXJ	-0.048911	-0.051251	0.006515	0.112622	0.087222	0.092307	0.029408	-0.000190	0.034867	0.156841	0.007233	-0.004429
IXN	-0.023753	-0.056151	-0.048853	-0.043480	-0.022197	-0.042047	-0.051147	0.041389	-0.011672	-0.134816	0.002663	0.019154
IXP	0.076797	0.064715	0.020796	0.033335	0.023335	0.029857	0.001707	-0.009238	0.017253	0.059961	-0.004913	-0.028478
JXI	-0.032809	0.000525	0.018449	-0.061183	-0.111083	-0.077424	-0.045056	0.014279	0.022697	-0.018940	0.002515	0.011592
KXI	0.037548	0.034388	0.090682	0.103077	0.112110	0.115919	0.131500	0.002423	0.016619	0.081870	0.014346	0.000452
MXI	-0.010919	0.020643	0.040102	0.039472	0.044799	0.043653	-0.047340	-0.025787	-0.018287	-0.227072	-0.007793	0.020073
REET	0.041221	0.054798	-0.002386	-0.016984	0.018083	0.013960	0.010249	-0.011533	0.014262	-0.199278	0.001229	0.002260
RXI	-0.025818	-0.043286	0.033183	-0.025437	-0.041685	-0.045743	0.030854	-0.008680	0.022988	-0.008915	-0.020627	-0.027565

Table 10: Markowitz Minimum Variance Portfolios Computed Using De Nard's Covariance Matrix Shrinkage Method.

		2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
Equities	**EXI**	1	2	3	3	1	0	2	2	1	0	1	3
	FILL	1	0	3	2	0	1	3	1	1	0	1	3
	IGF	0	1	2	0	3	2	2	2	0	0	1	3
	IXG	1	2	3	3	1	0	2	2	1	0	1	3
	IXJ	0	1	3	0	1	0	0	3	3	2	2	0
	IXN	1	2	0	3	1	0	0	3	3	2	2	0
	IXP	1	2	0	3	1	0	0	3	3	2	2	0
	JXI	0	1	2	0	1	0	1	2	0	0	1	3
	KXI	0	1	2	0	1	0	0	1	0	0	2	0
	MXI	1	2	3	3	1	0	2	1	1	2	2	1
	RXI	1	2	0	3	1	0	2	2	3	2	2	0
Fixed Income	**IAGG**	3	3	2	2	3	2	1	0	2	1	0	2
	ILTB	3	3	2	2	0	1	1	3	2	1	0	2
	IMTB	3	3	2	1	2	3	1	0	2	1	1	2
	ISTB	3	3	2	2	0	1	0	3	2	3	0	2
Real Estate	**REET**	0	1	2	3	2	3	1	2	0	0	1	3
Commodities	**DBA**	2	0	0	1	2	3	3	2	3	3	3	1
	DBB	2	0	3	0	3	2	0	1	3	3	3	1
	DBE	2	3	1	1	2	3	3	1	1	3	3	1
	DBP	3	3	1	2	0	1	0	3	2	1	3	1

Table 11: De Nard's Blockbuster Clustering Method Results over Time.

F Hierarchical Risk Parity Portfolios

	2020-01-02 10:00:00	2020-02-03 10:00:00	2020-03-02 10:00:00	2020-04-01 10:00:00	2020-05-01 10:00:00	2020-06-01 10:00:00	2020-07-01 10:00:00	2020-08-03 10:00:00	2020-09-01 10:00:00	2020-10-01 10:00:00	2020-11-02 10:00:00	2020-12-01 10:00:00
DBA	0.009747	0.008611	0.007197	0.061275	0.045903	0.051535	0.008027	0.008423	0.008529	0.006954	0.004167	0.006448
DBB	0.024404	0.026175	0.034225	0.150762	0.123860	0.134981	0.147170	0.064841	0.042798	0.072417	0.193952	0.099519
DBE	0.012178	0.006417	0.003372	0.015343	0.013030	0.021506	0.010914	0.005343	0.002536	0.001552	0.002793	0.000980
DBP	0.111303	0.182303	0.217312	0.415535	0.510832	0.510970	0.394915	0.412534	0.395083	0.171594	0.122178	0.205282
EXI	0.013956	0.010979	0.014047	0.014827	0.011571	0.011129	0.020553	0.017425	0.014971	0.009980	0.009700	0.016660
FILL	0.738136	0.705798	0.700888	0.164918	0.126497	0.121919	0.395199	0.478281	0.520145	0.717617	0.653601	0.659258
IAGG	0.011196	0.009710	0.000413	0.005222	0.005687	0.008254	0.000406	0.000254	0.000593	0.000852	0.000510	0.000802
IGF	0.016266	0.006206	0.001559	0.003143	0.002808	0.008867	0.002616	0.001086	0.002165	0.002106	0.000981	0.000741
ILTB	0.007028	0.004737	0.001336	0.003511	0.005860	0.003475	0.001740	0.000473	0.001014	0.001276	0.000741	0.000841
IMTB	0.005047	0.003839	0.000604	0.002656	0.002474	0.001499	0.000597	0.000578	0.000813	0.001186	0.000724	0.001887
ISTB	0.005562	0.001913	0.002217	0.004634	0.002636	0.003135	0.002353	0.000935	0.001304	0.002838	0.000654	0.001931
IXG	0.008690	0.002424	0.000441	0.004524	0.010792	0.010837	0.001937	0.000268	0.000740	0.000859	0.001627	0.000482
IXJ	0.002837	0.000660	0.000187	0.003829	0.002659	0.004336	0.000842	0.002066	0.000456	0.000581	0.000578	0.000475
IXN	0.004080	0.002247	0.000860	0.005401	0.005290	0.006849	0.001473	0.003009	0.000545	0.000586	0.000666	0.000707
IXP	0.004843	0.002874	0.000990	0.008864	0.007034	0.005890	0.000625	0.000963	0.000861	0.000708	0.001255	0.000848
JXI	0.007217	0.001166	0.000907	0.009860	0.009350	0.003753	0.000689	0.000989	0.000246	0.000585	0.001373	0.000618
KXI	0.007204	0.003014	0.000975	0.022708	0.013236	0.004973	0.002093	0.000276	0.000365	0.001275	0.001317	0.000691
MXI	0.006760	0.010429	0.009023	0.082065	0.087044	0.075281	0.006353	0.001659	0.005561	0.004707	0.001829	0.001051
REET	0.001104	0.002935	0.002450	0.015431	0.007956	0.005733	0.000796	0.000419	0.000909	0.001387	0.000906	0.000509
RXI	0.002442	0.007563	0.000997	0.005492	0.005481	0.005078	0.000702	0.000179	0.000366	0.000940	0.000449	0.000271

Table 12: Hierarchical Risk Parity Portfolio Allocation over Time.