Edward Churchill
ed.churchill00@gmail.com

Portfolio Construction with Hierarchical Risk Parity: a Comparison to the Minimum-Variance and Risk-Parity Methods

CONTENTS

Links

The data required for this project was found on Yahoo Finance. The Python notebooks were written on Kaggle and are stored here on GitHub. The 7 notebooks are as follows:

- Data Collection
- EDA and Data Cleaning
- Hierarchical Risk Parity Portfolio Construction
- Risk Parity Portfolio Construction
- Minimum Variance Portfolio Construction
- Equal Weight Portfolio Construction
- Results

Objective

The objective of this analysis is based on this 2016 paper by Marcos López de Prado called *Building Diversified Portfolios that Outperform Out-of-Sample*. He uses a technique called Hierarchical Risk-Parity (HRP) to construct portfolios that allocate risk more effectively than traditional techniques such as Risk-Parity and Minimum-Variance. HRP uses the unsupervised learning algorithm *hierarchical agglomerative clustering* to group 'similar' securities together before assigning each of them a weight in a smart way.

This analysis includes a practical implementation of three slightly different versions of HRP. We choose the one that performs best and compare it to the Risk-Parity and Minimum-Variance methods, as well as a naïve equally-weighted portfolio. However, before diving into the practical implementations, it is important to give some background on the various portfolio construction methods. We also consider each method's limitations to motivate the need for HRP.

Portfolio Construction

Returns and Covariance Matrices

Suppose we have an investment universe of n securities. To construct a portfolio, we have to assign each of them a weight w_i where each $w_i \geq 0$ and $\sum_{i=1}^{n} w_i = 1$. Suppose further that we have m time periods of historical returns for each of the n securities. These can be placed into an $m \times n$ matrix $R = (r_{ij})$ which we will call the returns matrix.

From the returns matrix R, we can construct an $n \times n$ covariance matrix K . It shows the covariance between any two securities' returns, which is a measure of how correlated they are. If securities a and b have returns series $\boldsymbol{r}_a$ and $\boldsymbol{r}_b$ respectively, then their covariance is given by:

$$\mathrm{Cov}(\boldsymbol{r}_a, \boldsymbol{r}_b) = \mathrm{Corr}(\boldsymbol{r}_a, \boldsymbol{r}_b)\sqrt{\mathrm{Var}(\boldsymbol{r}_a)\mathrm{Var}(\boldsymbol{r}_b)} = \mathrm{Corr}(\boldsymbol{r}_a, \boldsymbol{r}_b)\mathrm{Std}(\boldsymbol{r}_a)\mathrm{Std}(\boldsymbol{r}_b)$$

where $\mathrm{Corr}()$ is the correlation coefficient between two series, $\mathrm{Var}()$ is the variance of the series and $\mathrm{Std}()$ is the standard deviation of the series.

It is important to observe that along the diagonal, since $\mathrm{Corr}(\boldsymbol{r}_a, \boldsymbol{r}_a) = 1$, the formula simplifies to

$$\mathrm{Cov}(\boldsymbol{r}_a, \boldsymbol{r}_a) = \mathrm{Var}(\boldsymbol{r}_a)$$

This is simply the variance of the security's returns. This can be used as a measure of the security's volatility and therefore a measure of risk.

Exponentially-Weighted Returns

Notice that when calculating the covariance matrix K, we used all m rows (time periods) of the returns matrix R. There was no consideration made to place more emphasis on recent returns than older returns. However, when constructing portfolios, we should be more interested in recent returns than older returns from a correlation perspective. This is because recent returns correlation should be more indicative of future returns correlation than older returns correlation.

To address this, we can apply an exponential weighting to the returns matrix R to place less emphasis on older returns before calculating the covariance matrix K. We can multiply each r_{ij} by λ^{m-i} for some decay parameter $0 < \lambda < 1$. This will decrease the size of returns further into the past (those with a small i), while keeping the most recent return the same (since $\lambda^{m-m} = 1$).

How do we decide which value of λ to use? In theory, we can use any value between 0 and 1. However, it is common to define λ using a *half-life*. The *half-life* τ is the number of time periods taken for the exponential weights to halve. In other words, the half-life corresponding to the decay parameter λ is the number τ such that:

$$\lambda^{\tau} = \frac{1}{2}$$

which can be rearranged to give

$$\lambda = \left(\frac{1}{2}\right)^{\frac{1}{\tau}}$$

Therefore, to calculate our exponentially-weighted returns matrix $\hat{R} = \left(\hat{r}_{ij}\right)$, we can choose a half-life τ and set

$$\hat{r}_{ij} = \lambda^{m-i} r_{ij} = \left(\frac{1}{2}\right)^{\frac{m-i}{\tau}} r_{ij}$$

Finally, we calculate the covariance matrix $\widehat{K}$ on the exponentially-weighted returns matrix $\hat{R}$. This matrix $\widehat{K}$ can then be used for all three portfolio construction techniques we discuss: Minimum Variance, Risk Parity and HRP.

Minimum-Variance Portfolio Construction

Formulation

Minimum-Variance portfolio construction does what it says on the tin. It aims to construct a portfolio of securities that minimizes the risk of the portfolio, which we measure through variance. It's easy to measure the variance of an individual security, but how can we measure portfolio-level variance? We derive a formula that utilises the previously discussed (exponentially-weighted) covariance matrix.

Let $\boldsymbol{R}_p$ be a random variable representing the (exponentially-weighted) portfolio-level return. Let $\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_n$ be random variables representing the (exponentially-weighted) return of each of the n securities making up the portfolio. We know that portfolio-level return is simply the weighted sum of the returns of individual securities. In other words:

$$\boldsymbol{R}_p = \sum_{i=1}^{n} w_i \boldsymbol{R}_i$$

By the definition of variance of a random variable, we can define portfolio-level variance as:

$$\mathrm{Var}(\boldsymbol{R}_p) = \mathbb{E}\left[\left(\boldsymbol{R}_p - \mathbb{E}[\boldsymbol{R}_p]\right)^2\right]$$

Substituting in our formula for $\boldsymbol{R}_p$ gives us:

$$\mathrm{Var}(\boldsymbol{R}_p) = \mathbb{E}\left[\left(\sum_{i=1}^{n} w_i \boldsymbol{R}_i - \mathbb{E}\left[\sum_{i=1}^{n} w_i \boldsymbol{R}_i\right]\right)^2\right]$$

By the linearity of expectation, we can move the inner expectation inside the second sum to get:

$$\mathrm{Var}(\boldsymbol{R}_p) = \mathbb{E}\left[\left(\sum_{i=1}^{n} w_i \boldsymbol{R}_i - \sum_{i=1}^{\boldsymbol{n}} w_i \mathbb{E}[\boldsymbol{R}_i]\right)^2\right]$$

Both sums now have common w_i terms so we can factorise to get:

$$\mathrm{Var}(\boldsymbol{R}_p) = \mathbb{E}\left[\left(\sum_{i=1}^{n} w_i (\boldsymbol{R}_i - \mathbb{E}[\boldsymbol{R}_i])\right)^2\right]$$

We can then expand the brackets to get:

$$\mathrm{Var}(\boldsymbol{R}_p) = \mathbb{E}\left[\begin{array}{c} \sum_{i=1}^{n} w_i^2 (\boldsymbol{R}_i - \mathbb{E}[\boldsymbol{R}_i])^2 \\ + \sum_{i=1}^{n} \sum_{i \neq j} w_i w_j (\boldsymbol{R}_i - \mathbb{E}[\boldsymbol{R}_i])(\boldsymbol{R}_j - \mathbb{E}[\boldsymbol{R}_j]) \end{array}\right]$$

We can then once again use linearity of expectation to move the outer expectation inside the sums to get:

$$\mathrm{Var}(\boldsymbol{R}_p) = \left(\begin{array}{c} \sum_{i=1}^{n} w_i^2 \mathbb{E}[(\boldsymbol{R}_i - \mathbb{E}[\boldsymbol{R}_i])^2] \\ + \sum_{i=1}^{n} \sum_{i \neq j} w_i w_j \mathbb{E}[(\boldsymbol{R}_i - \mathbb{E}[\boldsymbol{R}_i])(\boldsymbol{R}_j - \mathbb{E}[\boldsymbol{R}_j])] \end{array}\right)$$

Notice that we now have the definition of variance and covariance inside each sum which simplifies the formula to:

$$\mathrm{Var}(\boldsymbol{R}_p) = \sum_{i=1}^{n} w_i^2 \mathrm{Var}(\boldsymbol{R}_i) \; + \; \sum_{i=1}^{n} \sum_{i \neq j} w_i w_j \mathrm{Cov}(\boldsymbol{R}_i, \boldsymbol{R}_j)$$

Finally, this formula can be succinctly written as the quadratic form:

$$\mathrm{Var}(\boldsymbol{R}_p) = \boldsymbol{w^T} K \boldsymbol{w}$$

where $\boldsymbol{w}$ is the weights vector $(w_1, w_2, \ldots\ldots, w_n)$ and K is the (exponentially-weighted) covariance matrix of the returns series $\boldsymbol{R_i}$. It is important to notice that since the variances and covariances are known, this formula is quadratic in w_i.

Now that we have the formula for portfolio-level risk, we just need to formulate and solve the optimization problem to construct our portfolio. A constraint-based optimization problem consists of an objective function that we seek to minimize or maximize subject to certain constraints. In our case, the objective is to minimize portfolio-level variance, which we have just derived the formula for. The constraints are simply that our weights must be non-negative and sum to 1. This is shown formally below.

Choose weight vector $\boldsymbol{w}$ to minimize
$f(\boldsymbol{w}) = \boldsymbol{w^T} K \boldsymbol{w}$

Subject to:

- $w_i \geq 0$ for $i = 1, 2, \ldots\ldots, n$
- $\sum_{i=1}^{n} w_i = 1$

where K is the (exponentially-weighted) $n \times n$ covariance matrix of returns.

Figure 1: Formulation of the Minimum-Variance portfolio construction problem to minimize portfolio-level variance.

Since the objective function is quadratic and the constraints are linear, this optimization problem can be solved using quadratic programming methods. In our case, we use the Python library *cvxpy* with its built-in quadratic solvers to solve the problem.

Condition Number

In order to discuss the main limitation of Minimum-Variance portfolio construction, we must first understand the *condition number* of a matrix in the context of matrix inversion.

Firstly, we define the *Euclidean norm* of a matrix A (with n columns) to be the maximum amount that the mapping induced by A can stretch vectors. More formally:

$$\| A \| = \max_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\| A\boldsymbol{x} \|}{\| \boldsymbol{x} \|} = \max_{\|\boldsymbol{x}\| = 1} \| A\boldsymbol{x} \|$$

When A is invertible, let's also consider the *minimum* amount that the mapping induced by A can stretch vectors. Let $\boldsymbol{y} = A\boldsymbol{x}$ so that this minimum is given by:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\| A\boldsymbol{x} \|}{\| \boldsymbol{x} \|} = \min_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\| \boldsymbol{y} \|}{\| A^{-1}\boldsymbol{y} \|} = \frac{1}{\max\limits_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\| A^{-1}\boldsymbol{y} \|}{\| \boldsymbol{y} \|}} = \frac{1}{\| A^{-1} \|}$$

The *condition number for inversion* $\kappa(A)$ is the ratio of the maximum stretching amount to the minimum stretching amount. Therefore, by the above, we have:

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

If a matrix is singular (has no inverse), it must map some vectors to the zero vector, and so the minimum stretching factor is 0. This leads to $\kappa(A) = \infty$.

The condition number can tell us about the size of the error when computing the inverse of a matrix given a small error in the matrix itself. For a small change E in the original matrix A, it can be shown that:

$$\frac{\| (A + E)^{-1} - A^{-1} \|}{\| A^{-1} \|} \leq \kappa(A) \frac{\| E \|}{\| A \|}$$

This shows that for a small relative error in the original matrix A, the relative error in the inverse matrix A^{-1} can be up to $\kappa(A)$ times as big. Therefore, inverting a matrix with a high condition number can lead to two very different results for only small changes in the original matrix.

Markowitz' Curse

The main limitation of the Minimum-Variance method is known as *Markowitz' Curse*. This is caused by the covariance matrix K having a high condition number.

Recall that for Minimum-Variance portfolio construction, we have to solve a quadratic programming problem. Part of this process requires the covariance matrix K to be inverted. When we solve the problem for the first time, we will get some set of weights at the output. However, imagine we want to solve the problem again but with a very small change in the covariance matrix, such as adding a new security or new time periods. If the condition number is high, this would lead to a completely different inverse matrix K^{-1}, which in turn leads to a completely different set of weights! To make matters worse, as we add correlated securities, the condition number of K will increase. This is *Markowitz' Curse*: the more correlated the investments, the greater the need for diversification and yet the more likely the solutions are unstable.

The instability of solutions to the Minimum-Variance method makes it extremely impractical to trade. For example, suppose you are running a portfolio using this method with monthly rebalancing. On day one it may tell you to buy a large amount of a particular security. Then when you rerun the optimization on new data the following month to rebalance, it may tell you to completely sell out of that position. Rerunning the following month it may result in you buying into the position again. These large position adjustments will generate large transaction costs and negatively affect the portfolio's performance.

Risk-Parity Portfolio Construction

Improving on Minimum-Variance

Risk-Parity is a technique to construct portfolios where each security is assigned a weight in such a way that it has some desired *percentage risk contribution* to the portfolio. More often than not, we choose to assign weights so that the percentage risk contribution of each security is equal. Risk-Parity addresses some of the limitations of Minimum-Variance:

- Risk-Parity does not require inversion of the covariance matrix K, leading to greater stability of solutions and therefore a more practical investment strategy with lower turnover and transaction costs.
- Risk-Parity leads to a less concentrated portfolio than Minimum-Variance, which may massively overweight individual low-risk securities, reducing diversification and potentially losing out on returns.

Formulation

The first thing we need to do is calculate the *percentage risk contribution* of each security to the portfolio. This measures how much risk an individual security contributes as a percentage of portfolio-level risk.

We already have a formula for portfolio-level risk from the Minimum-Variance section. That is:

$$\mathrm{Var}(\boldsymbol{R}_p) = \boldsymbol{w}^T K \boldsymbol{w}$$

To calculate a security's percentage risk contribution, we first need its *absolute risk contribution*. We can see from the above that $K\boldsymbol{w}$ is a vector containing the absolute risk of each individual security. That is, $Var(\boldsymbol{R_i}) = (K\boldsymbol{w})_i$ for any security i. To get the absolute risk contribution, we simply multiply the absolute risk by the weight. Denoting the risk contribution of security i by RC_i, we have:

$$\mathrm{RC}_i = \boldsymbol{w}_i \mathrm{Var}(\boldsymbol{R}_i) = \boldsymbol{w}_i (K\boldsymbol{w})_i$$

Finally, we can divide by portfolio-level risk to get the percentage risk contribution. Denoting percentage risk contribution by PRC_i, we have

$$\mathrm{PRC}_i = \frac{\mathrm{RC}_i}{\mathrm{Var}(\boldsymbol{R}_p)} = \frac{\boldsymbol{w}_i \mathrm{Var}(\boldsymbol{R}_i)}{\mathrm{Var}(\boldsymbol{R}_p)} = \frac{\boldsymbol{w}_i (K\boldsymbol{w})_i}{\boldsymbol{w}^T K \boldsymbol{w}}$$

Now that we have the formula for the percentage risk contribution of each security, we just need to formulate and solve the optimization problem to construct our portfolio. Recall that we want the percentage risk contribution of each security to be equal for Risk-Parity. In other words, we require each $\mathrm{PRC}_i = \frac{1}{n}$ where n is the number of securities in the portfolio. This can be rearranged to give $\mathrm{PRC}_i - \frac{1}{n}$ = 0 which can be achieved by minimizing for the sum of the square differences between each PRC_i and $\frac{1}{n}$. Our optimization problem therefore becomes:

Choose weight vector $\boldsymbol{w}$ to minimize

$$f(\boldsymbol{w}) = \sum_{i=1}^{n} \left(\mathrm{PRC}_i - \frac{1}{n}\right)^2 = \sum_{i=1}^{n} \left(\frac{w_i (K\boldsymbol{w})_i}{\boldsymbol{w}^T K \boldsymbol{w}} - \frac{1}{n}\right)^2$$

Subject to:

- $w_i \geq 0$ for $i = 1, 2, \ldots\ldots, n$
- $\sum_{i=1}^{n} w_i = 1$

where K is the (exponentially-weighted) $n \times n$ covariance matrix of returns.

Figure 2: Formulation of the Risk-Parity portfolio construction problem to ensure each security contributes equally to risk.

The objective function is non-linear due to the division by our variable $\boldsymbol{w}$. Fortunately, the Python library *scipy* can handle this with its built-in non-linear solvers.

Limitations

While Risk-Parity should provide more stable solutions than Minimum Variance (since it doesn't require the covariance matrix to be inverted), there are still some limitations.

The main limitation is that Risk-Parity can over-concentrate low-risk securities. This is because they need a larger weight in the portfolio to achieve the same percentage risk contribution as a higher-risk security. This can lead to a poorly diversified portfolio where there is too much weight in a particular sector or geography. In times of market stress, this lack of diversification can lead to poor performance. HRP directly addresses this issue and still doesn't require the inversion of the covariance matrix.

HRP Portfolio Construction

Improving on Risk-Parity

As mentioned previously, Risk-Parity can lead to poor diversification due to high concentrations in particular sectors or geographies. HRP addresses this by clustering securities together based on their correlation similarity, resulting in a hierarchical tree-like structure. Risk is then distributed in a smart way (which we will describe in more detail) across the hierarchy, leading to better diversification.

Formulation

Overview

HRP is a three step process:

1. **Hierarchical Agglomerative Clustering:** Create a tree-like structure called a *dendrogram* showing the relationship between securities from a correlation perspective.
2. **Quasi-Diagonalisation:** Reorder the rows and columns of the covariance matrix based on the results of hierarchical clustering. This makes the next step (assigning weights) easier.
3. **Recursive Bisection:** Starting from the top of the tree, look at each split into sub-trees. Allocate a total weight to each sub-tree so that the risk contribution of each sub-tree is equal. Recursively repeat this process, starting from the top of each subtree until each subtree only contains one security. This will result in a weight for each security so that there is risk-parity in each cluster, hence the name HRP.

Notice that there is no constraint-based optimization being performed as there was for Minimum-Variance and Risk-Parity. Instead, we rely on clustering and then a simple recursive algorithm to assign weights. We now go into more detail on each step.

Stage 1: Hierarchical Agglomerative Clustering

We want to create a hierarchical tree structure to capture the relationships between securities from a correlation perspective. To do this, we first take our returns matrix R and use it to construct an $n \times n$ correlation matrix $\rho = (\rho_{ij})$. It gives the correlation coefficient between any two securities' returns. If securities a and b have returns series $\boldsymbol{r}_a$ and $\boldsymbol{r}_b$ respectively, then their correlation coefficient ρ_{ab} is given by

$$\mathrm{Corr}(\boldsymbol{r}_a, \boldsymbol{r}_b) = \frac{\mathrm{Cov}(\boldsymbol{r}_a, \boldsymbol{r}_b)}{\sqrt{\mathrm{Var}(\boldsymbol{r}_a)\mathrm{Var}(\boldsymbol{r}_b)}} = \frac{\mathrm{Cov}(\boldsymbol{r}_a, \boldsymbol{r}_b)}{\mathrm{Std}(\boldsymbol{r}_a)\mathrm{Std}(\boldsymbol{r}_b)}$$

where $\mathrm{Cov}()$ is the covariance between two series, $\mathrm{Var}()$ is the variance of the series and $\mathrm{Std}()$ is the standard deviation of the series.

Next, we need to define some distance metrics to help us to identify similar securities that will be clustered together. Smaller distances should indicate similarity. We first define the distance between any two securities i and j via the function $d\colon \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\} \to [0, 1]$ where

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

Notice:

- If $\rho_{ij} = 1$ (the securities are perfectly positively correlated) then $d(i, j) = 0$.
- If $\rho_{ij} = -1$ (the securities are perfectly negatively correlated) then $d(i, j) = 1$.
- If $\rho_{ij} = 0$ (the securities are perfectly uncorrelated) then $d(i, j) = 0.5$.

The distance metric d allows us to define an $n \times n$ distance matrix $D = (d_{ij})$. This is symmetric since the correlation matrix ρ is symmetric. A simple 3×3 example is shown below.

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

Figure 3: Constructing the distance matrix from the correlation matrix. It shows how similar any two securities (from a correlation perspective) on a scale of 0 to 1 with smaller numbers meaning higher similarity.

Notice that the metric d only considers the single correlation coefficient between any two securities to calculate similarity. However, we should also consider each security's correlation with *all* other securities. To do this, we define a second distance metric that operates on any two column vectors D_i and D_j of the distance matrix D. This is given by:

$$\hat{d}(D_i, D_j) = \sqrt{\sum_{k=1}^{n} (d_{ki} - d_{kj})^2}$$

Notice:

- If $i = j$, then $\hat{d}(D_i, D_j) = 0$.
- Otherwise, similar securities still take values closer to 0 and less similar securities take larger values.

The distance metric $\hat{d}$ allows us to define an $n \times n$ 'distance of distance' matrix $\hat{D} = (\hat{d}_{ij})$. This is symmetric since the distance matrix D is symmetric. Below we show the 'distance of distance' matrix $\hat{D}$ for the same example as before.

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

Figure 4: Constructing the 'distance of distance' matrix from the distance matrix. It still shows how similar any two securities are (from a correlation perspective) but this time the whole correlation matrix is taken into account.

Now that we have a matrix $\hat{D}$ measuring distance between any two securities, we are ready to perform hierarchical agglomerative clustering, which is a bottom-up approach to clustering. The idea is to start with each security in its own cluster. We then iteratively combine the 'closest' clusters until there is only one cluster containing all the securities. Every time we merge two clusters together, we create a new level in our dendrogram. An example dendrogram is shown below followed by a detailed description of the algorithm.

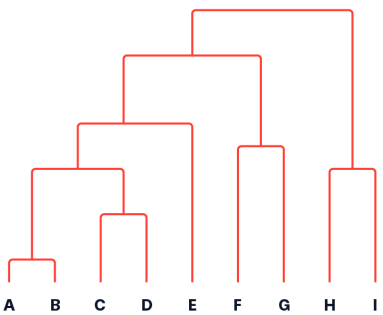


Figure 5: An example dendrogram which is the result of hierarchical agglomerative clustering. Here, securities A and B are merged to form a cluster, as are C and D. These two clusters are then merged to form a larger cluster which is then merged with the cluster containing E. The process continues until all securities are in one overarching cluster.

STEP 1

Start with each security i in its own cluster. Initialize the distance matrix $\widehat{D}$ to show the distance between any two clusters, which to start with is just the distance between the corresponding securities. We also take a copy $\widetilde{\mathrm{D}} = \widehat{\mathrm{D}}$ because $\widehat{\mathrm{D}}$ will get updated dynamically but we still need the original matrix for STEP 2.2.

STEP 2

This step merges clusters until we have only one cluster. This requires $n-1$ iterations which we index by k. We index the original clusters from STEP 1 with the numbers 1 to n. We index the cluster formed on iteration k by $n + k$.

Initialize the $(n-1) \times 4$ *linkage matrix* L with 0s. Once STEP 2 is complete, each row k will contain the following information from iteration k:

- Columns 1 and 2 contain the indices of the two clusters merged to form cluster $n + k$ on iteration k.
- Column 3 contains the distance between these two clusters.
- Column 4 contains the number of securities in cluster $n + k$.

FOR $k = 1, 2, \ldots\ldots, n-1$:

STEP 2.1

Find the most similar pair of clusters using the matrix $\widehat{D}$ by finding the pair of clusters (i^*, j^*) with $i^* \neq j^*$ such that:

$$(i^*, j^*) = \underset{(i,j)}{\operatorname{argmin}}\left(\hat{d}_{ij}\right)$$

If there are multiple such pairs, just pick one. Then combine clusters $i^*$and j^* into cluster $n + k$ so we have $n + k = i^* \cup j^*$. Update the linkage matrix L with:

- $L_{k1} = i^*$
- $L_{k2} = j^*$
- $L_{k3} = \hat{d}_{i^* j^*}$
- $L_{k4} = |n + k|$

Drop rows and columns i^* and j^* from matrix $\widehat{D}$. Add a row and column indexed $n + k$, initially containing zeros. In STEP 2.2, we will calculate the distance from our cluster $n + k$ to every other cluster still in the matrix $\widehat{D}$ and update the values in the new row and column accordingly.

STEP 2.2

Now we calculate the distance $\dot{d}_{i,n+k} = \dot{d}_{n+k,i}$ from our newly formed cluster $n + k$ and each cluster i still in the matrix $\widehat{D}$. There are a few different *linkage criteria* we can use to calculate the distance between clusters. Remembering that $\widetilde{D}$ is our original distance matrix between securities from STEP 1, we have the options:

- *Nearest Point*: The minimum distance between a security s_i in cluster i and a security s_{n+k} in cluster $n + k$. That is:

$$\dot{d}_{i,n+k} = \min_{s_i \in i, s_{n+k} \in n+k} \left(\tilde{\mathrm{d}}_{s_i, s_{n+k}}\right)$$

- *Average*: The mean distance from securities s_i in cluster i to securities s_{n+k} in cluster $n + k$. That is:

$$\dot{d}_{i,n+k} = \frac{\sum_{s_i \in i, s_{n+k} \in n+k} \tilde{\mathrm{d}}_{s_i, s_{n+k}}}{|i| \cdot |n + k|}$$

- *Ward*: This is a more complex formula considering the distance between cluster centroids. This distance metric aims to minimize the increase in cluster variance after merging. Recalling that clusters i^* and j^* were merged to create cluster $n + k$ in STEP 2.1, and letting c_i, c_{i^*}, c_{j^*} be the respective cluster centroids, the formula is:

$$\dot{d}_{i,n+k} = \frac{1}{\sqrt{|i| + |i^*| + |j^*|}} \sqrt{(|i| + |i^*|)\|c_i - c_{i^*}\|^2 + (|i| + |j^*|)\left\|c_i - c_{j^*}\right\|^2 - |i| \cdot \left\|c_{i^*} - c_{j^*}\right\|^2}$$

After picking one of these linkage criteria, we now have the distance $\dot{d}_{i,n+k} = \dot{d}_{n+k,i}$ from our newly formed cluster $n + k$ to each cluster i still in the matrix $\widehat{D}$. So we can simply update $\widehat{D}$ by setting

$$\hat{\mathrm{d}}_{i,n+k} = \hat{\mathrm{d}}_{n+k,i} = \dot{d}_{i,n+k}$$

Figure 6: The bottom-up hierarchical clustering algorithm, with a choice of linkage criteria.

Once this algorithm has terminated, we are left with the $(n-1) \times 4$ linkage matrix L that defines our dendrogram. This concludes stage one of the HRP process. We now move onto stage 2.

Stage 2: Quasi-Diagonalisation

Quasi-Diagonalisation is the next step in the HRP process. The algorithm details can be seen in De Prado's paper, but the basic idea is to reorder the covariance matrix so that the securities appear in the same order as they do in the dendrogram. This is just a preparation step in order to perform *recursive bisection*.

Stage 3: Recursive Bisection

Now that we have our dendrogram and the covariance matrix has been appropriately reordered, we are finally ready to allocate weights to each security. The idea is to ensure that every time there is a split in the tree, the same amount of risk is assigned to the left branch as the right branch, hence the name Hierarchical Risk Parity.

This is done recursively starting from the top of the tree. We describe the algorithm in detail below. For ease of notation, call the ordered securities $\{1, 2, \ldots, n\}$.

STEP 1
Initialize the weights vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ to have a weight of 1 for each security.

STEP 2
Initialize a set of clusters C, where each cluster is a set of securities. To start, we just have one cluster with all securities. That is $C = \{\{1,2,\ldots,n\}\}$.

STEP 3
Now we perform recursive bisection on C

STEP 3.1
If $|C| = 0$, return $\boldsymbol{w}$.

STEP 3.2
For each cluster $c \in C$, if $|c| = 1$, remove it from C. Otherwise bisect c into two clusters c_1 and c_2 with $|c_1| = \left\lfloor \frac{|c|}{2} \right\rfloor$ and $|c_2| = |c| - |c_1|$. Replace $c \in C$ with c_1 and c_2.

STEP 3.3
Now for every corresponding pair of clusters $c_1, c_2 \in C$, compute the cluster variances v_1 and v_2. We show how to do this for v_1 below (it's the same for v_2):

- Take the covariance matrix K and take a slice of it by only keeping rows and columns corresponding to securities in c_1. Call this matrix $\widehat{K}$.

- Take the diagonal elements of K which are the variances of each security in c_1. Store these in a vector $\boldsymbol{v}$.
- For each variance v in the vector $\boldsymbol{v}$, compute $\frac{1}{vS}$ where S is the sum of all elements of $\boldsymbol{v}$**.** Store these numbers in a vector $\boldsymbol{x}$.
- Finally, the variance v_1 of the cluster c_1 is defined by the quadratic form:

$$v_1 = Var(c_1) = \boldsymbol{x}^{\boldsymbol{T}} \widehat{K} \boldsymbol{x}$$

With the cluster variances v_1 and v_2, calculate the split factor

$$\alpha = \frac{1 - v_1}{v_1 + v_2}$$

Then for each security $i \in c_1$ and $j \in c_2$, update w_i to be αw_i and w_j to be $(1 - \alpha) w_j$

STEP 3.4
Repeat Steps 3.1 to 3.4 until the stopping condition in STEP 3.1 is met.

Figure 7: Recursive Bisection, the third stage of HRP. This utilizes the dendrogram to assign weights to each security in such a way that whenever a branch splits, the left branch and right branch have the same amount of risk, hence the name hierarchical risk parity.

Allocating risk in this way is beneficial because we allocate appropriate amounts of risk to a *cluster* of similar securities, rather than single securities as in Risk-Parity. This helps to overcome the diversification problems mentioned in the limitations of Risk-Parity. This concludes the portfolio construction process for HRP.

Limitations

HRP helps to overcome many of the issues with Minimum-Variance and Risk-Parity. However, it's important to make note of a few limitations:

- Just like the other methods, it's reliant on historical data, which may not be an accurate forecast of future market conditions.
- There is no explicit return maximization. It's just a better way of allocating risk. This may not always align with an investor's goals.
- It's fairly computationally expensive compared to Minimum-Variance and Risk-Parity.

That being said, it's still a very good portfolio construction technique for allocating risk in a better way than traditional methods. Now that we have the necessary theoretical background on Minimum-Variance, Risk-Parity and HRP, we are finally ready for a practical implementation!

Practical Implementation

Data Description

Now that we have the necessary theoretical background on various portfolio construction techniques, we are ready to apply them to a real dataset and compare their performance. For this project, our securities are *currency pairs* all with base currency USD.

A *currency pair* is a security XXX/YYY where XXX is the *base currency* and YYY is the *quote currency*. The value of the security is simply how much of the quote currency can be exchanged for 1 unit of the base currency. For example, if the pair USD/GBP has a value 0.65, it means that 1$ can buy you £0.65. For a pair USD/YYY:

- If the value of the pair goes up, it means that 1$ can buy you more of the quote currency and so the dollar has appreciated against the quote currency.
- If the value of the pair goes down, it means that 1$ can buy you less of the quote currency and so the dollar has depreciated against the quote currency.
- Sometimes, a country's central bank will 'peg' its currency against the dollar so that the value of the pair stays constant. This is a measure to stabilise the value of the currency. Central banks enter FX markets and buy/sell appropriate amounts of the pair to maintain the pegged rate. There may still be a small amount of volatility because the pair is still subject to the supply and demand dynamics of the FX market, but the central bank will do its best to mitigate this.

For our dataset, we initially take historical values (2006-2019) of various pairs from Yahoo Finance. A list of these pairs and a snippet of the initial dataset are shown below:

Pair	Quote Currency Description
USD/AUD	Australian Dollar
USD/BRL	Brazilian Real
USD/CAD	Canadian Dollar
USD/CNY	Chinese Yuan
USD/EUR	Euro
USD/INR	Indian Rupee
USD/IDR	Indonesian Rupiah
USD/JPY	Japanese Yen
USD/MXN	Mexican Peso
USD/ZAR	South African Rand
USD/KRW	South Korean Won
USD/GBP	British Pound
USD/CHF	Swiss Franc
USD/SEK	Swedish Krona
USD/NOK	Norwegian Krone
USD/NZD	New Zealand Dollar
USD/CZK	Czech Koruna
USD/HUF	Hungarian Forint
USD/PLN	Polish Zloty

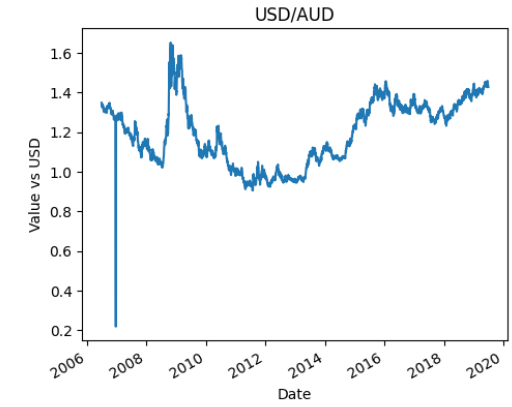
Figure 8: Currency pairs in the initial dataset.

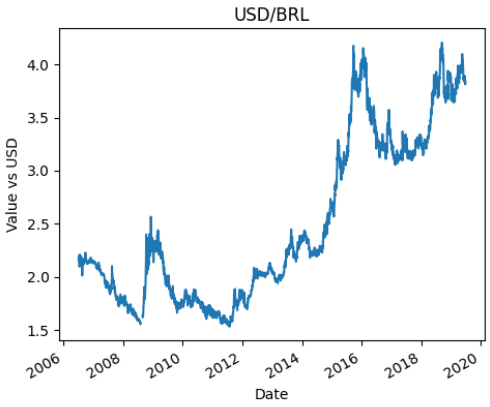
Date	USD/AUD	USD/BRL	USD/CAD	USD/CNY	USD/EUR	USD/INR	USD/IDR
2006-07-03 00:00:00+01:00	1.3463	2.1570	1.1116	7.9902	0.78119	45.939999	9050.0
2006-07-04 00:00:00+01:00	1.3444	2.1635	1.1087	7.9823	0.78186	45.900002	8964.0
2006-07-05 00:00:00+01:00	1.3479	2.1885	1.1121	7.9860	0.78530	45.950001	9020.0
2006-07-06 00:00:00+01:00	1.3410	2.1790	1.1125	7.9860	0.78272	45.939999	9020.0
2006-07-07 00:00:00+01:00	1.3298	2.1673	1.1114	7.9755	0.77979	45.785000	8965.0

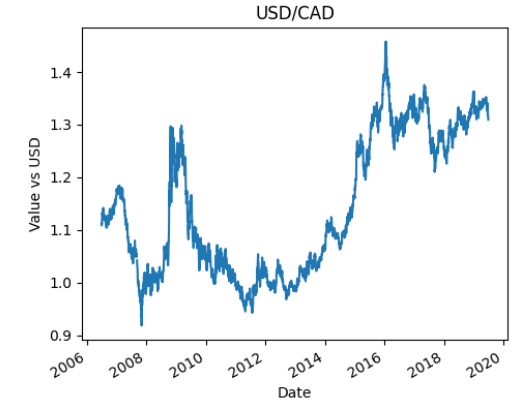
Figure 9: First few rows and columns of the dataset, showing values of currency pairs over time.

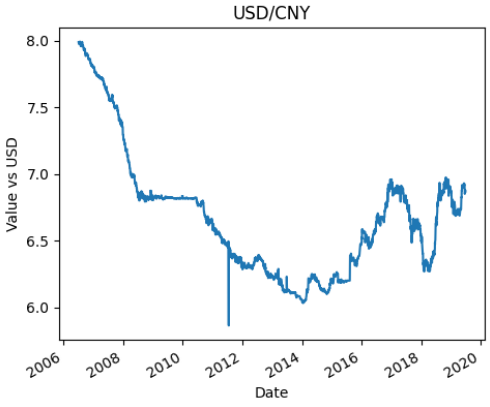
Data Cleaning

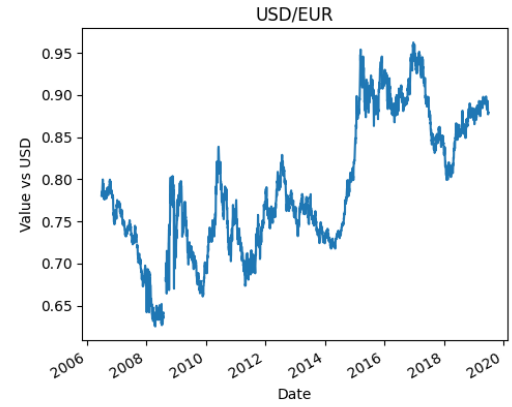
To better understand the data and to see if it needs any cleaning, we plot the values of all the pairs over time.

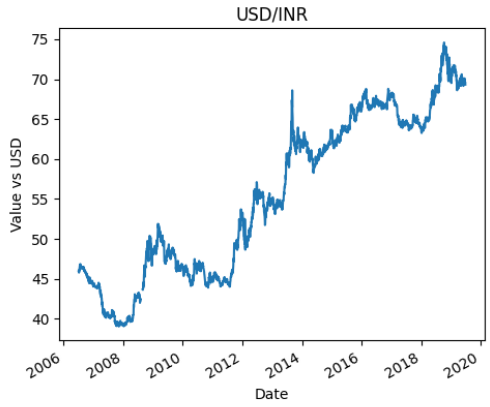

USD/AUD
Value vs USD
Date

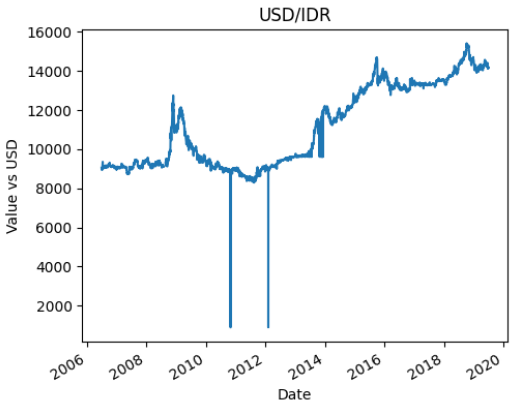

USD/BRL
Value vs USD
Date

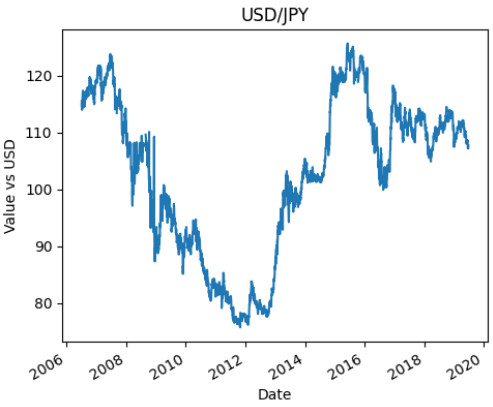

USD/CAD
Value vs USD
Date

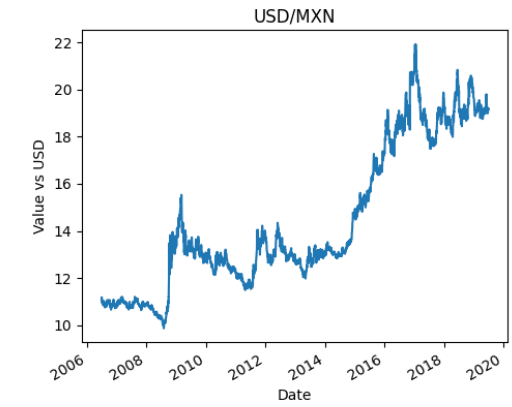

USD/CNY
Value vs USD
Date

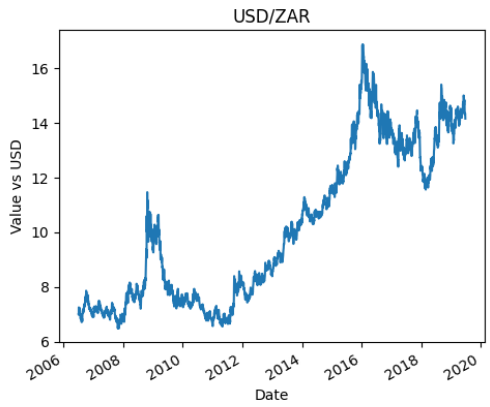

USD/EUR
Value vs USD
Date


USD/INR
Value vs USD
Date


USD/IDR
Value vs USD
Date


USD/JPY
Value vs USD
Date


USD/MXN
Value vs USD
Date


USD/ZAR
Value vs USD
Date

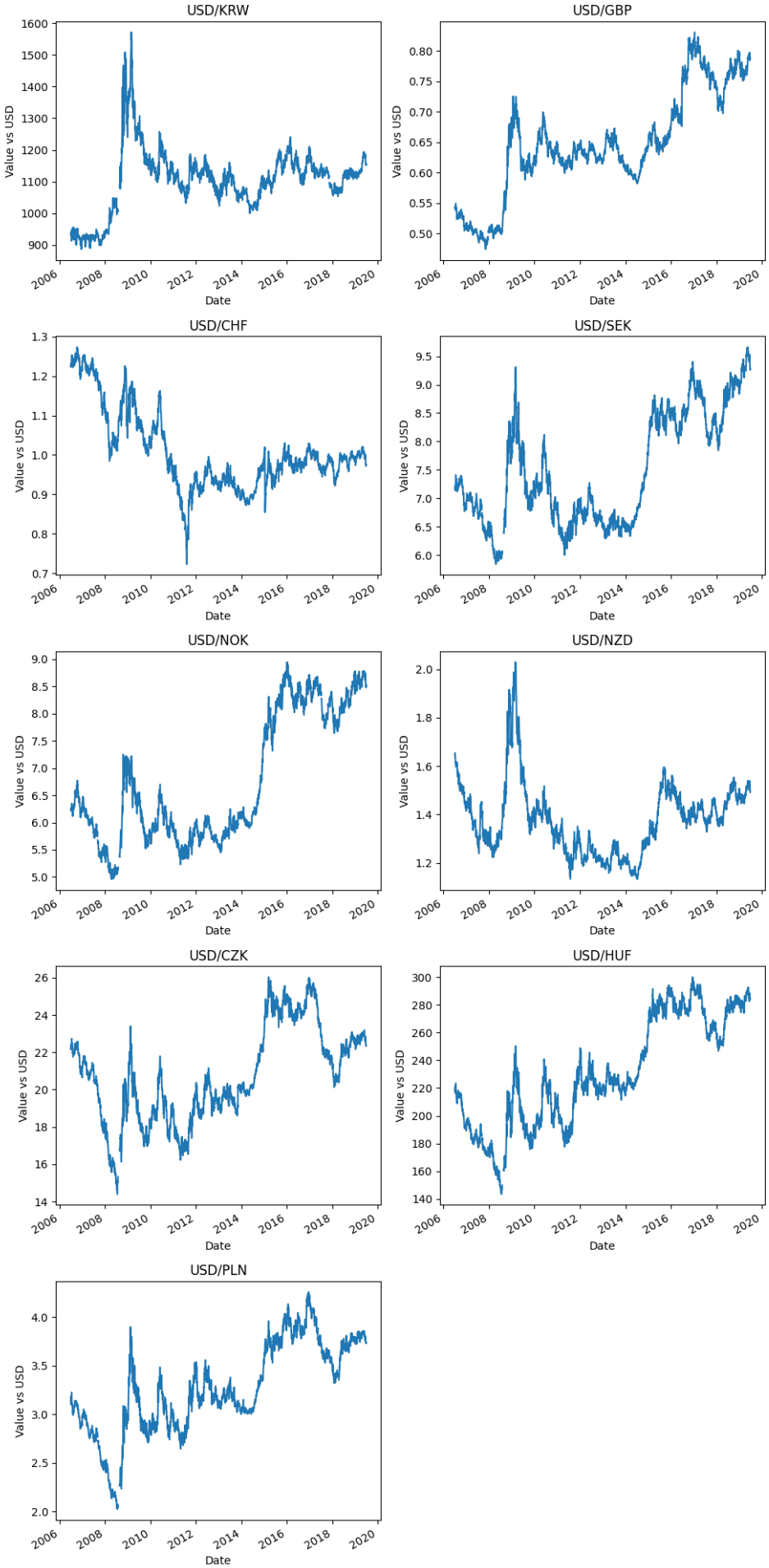


Figure 10: Values of all currency pairs over time (prior to any data cleaning). Notice there is some missing data and some probable outliers.

There are two primary observations to make:

- Towards the end of 2008, there is lots of missing data. To get around this, we simply trim the data to start from 2009.
- The currencies CNY and IDR have sustained periods of pegging and some outliers. Pegged currencies will not be useful for our portfolio construction because we want securities whose value changes over time against the US dollar. We therefore choose to exclude both of these currencies from our investment universe.

To handle any other null values, we forward fill values for each pair (replace any null values with the last available value).

EDA

Now that the data is clean, we take a look at some exploratory plots with descriptions.

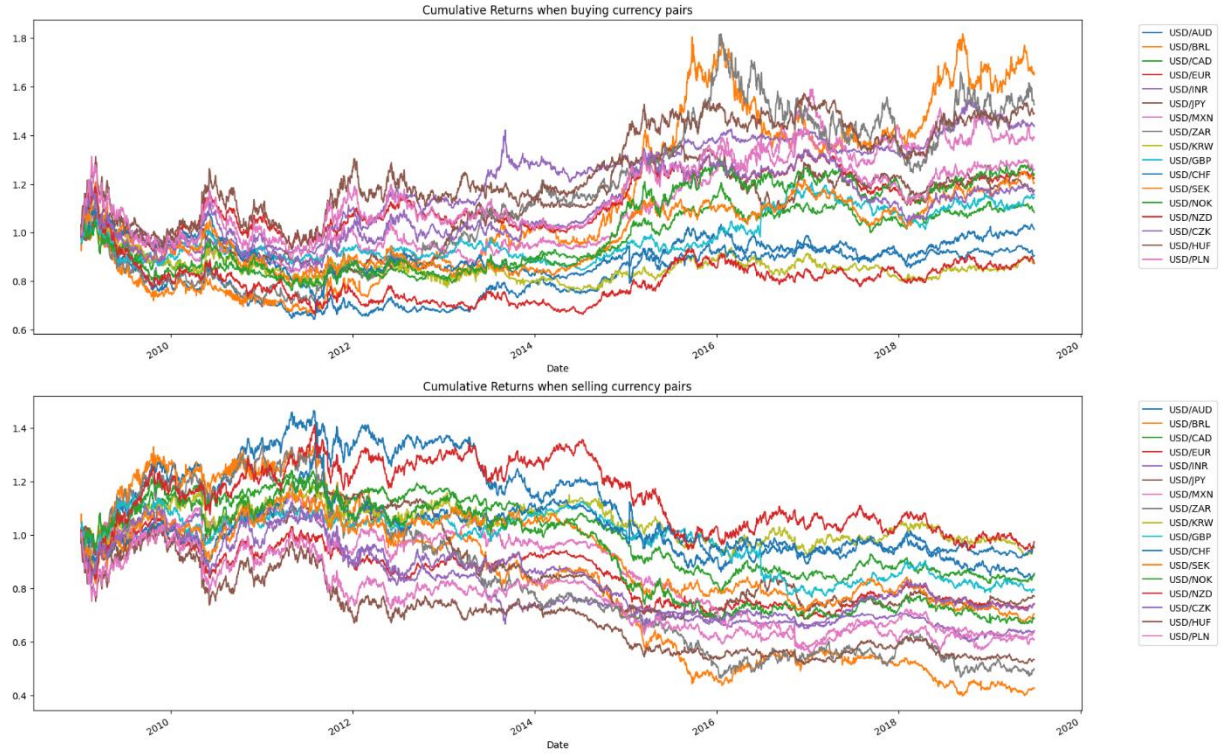


Figure 11: Cumulative returns of each currency pair when going long (top) or short (bottom). Notice that the values of the currency pairs go up on average time, implying that the USD has appreciated against other currencies.

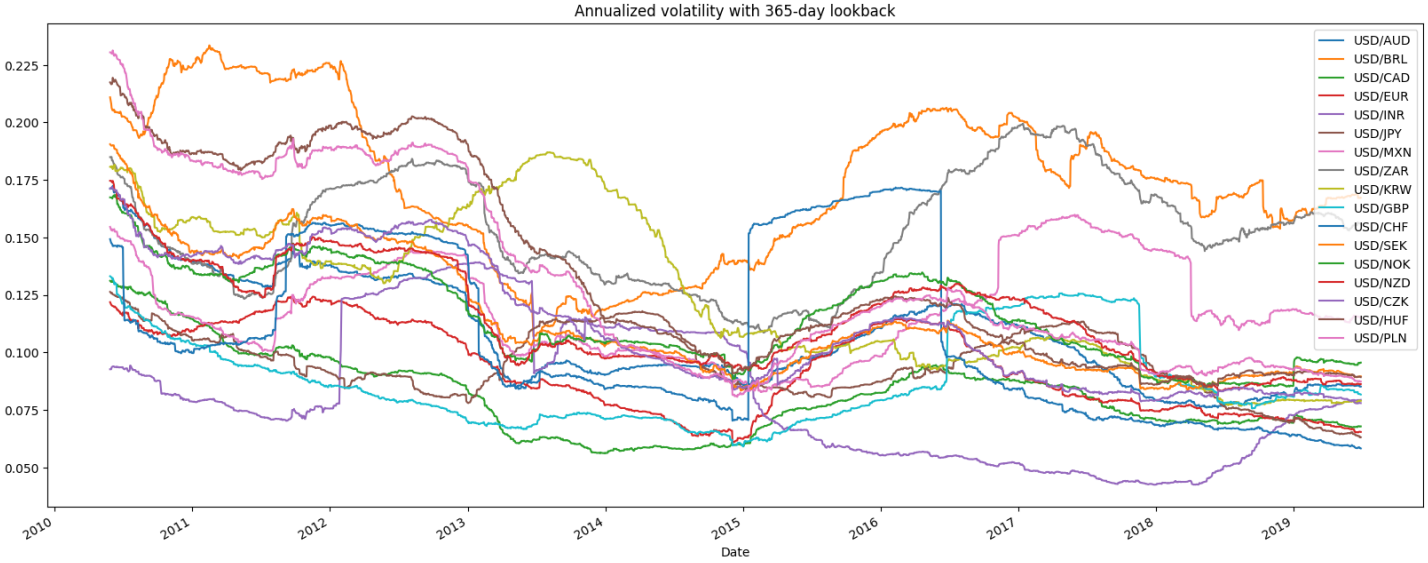


Figure 12: Annualized 365-day volatility of each currency pair. Notice that some currencies have big jumps implying that there are occasionally massive changes in the currency's value. This causes big jumps in volatility when those values enter/drop out of the 365-day rolling window.

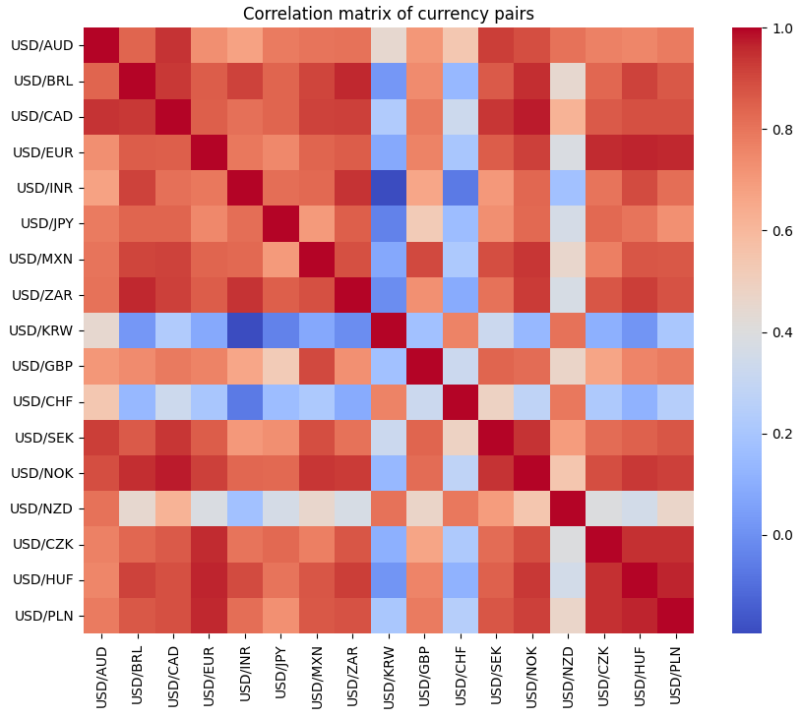


Figure 13: Correlation matrix of the values of the currency pairs. Notice most are highly correlated but some such as KRW (South Korean Won) and CHF (Swiss Franc) are relatively uncorrelated to the other pairs.

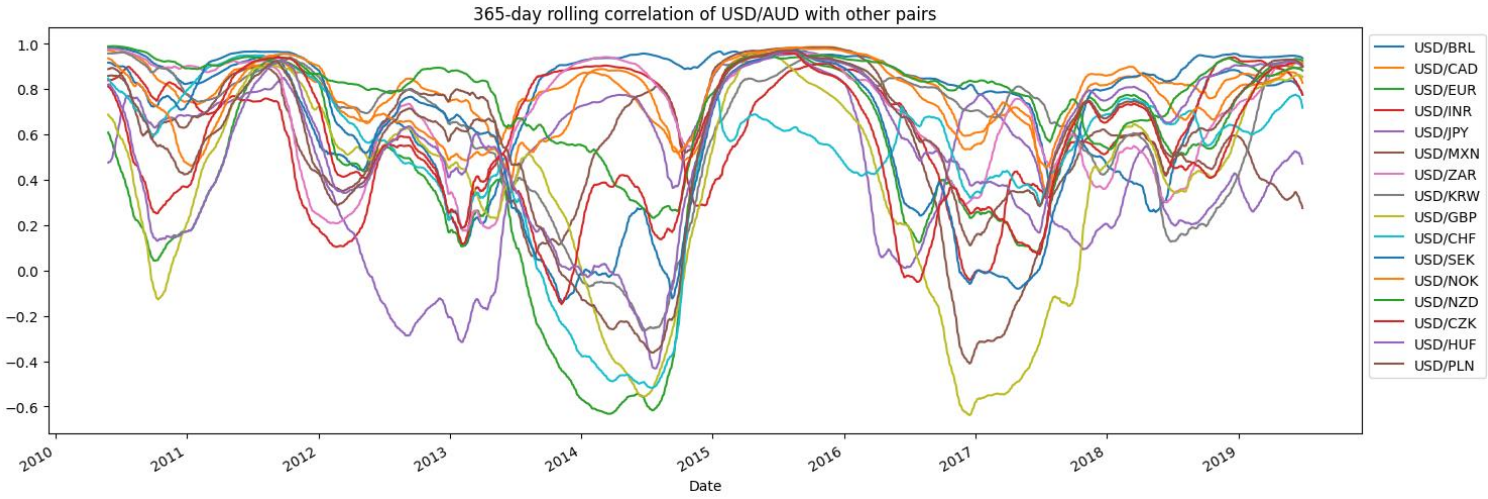


Figure 14: 365-day rolling correlation of USD/AUD with other pairs. The choice of AUD is arbitrary, The important takeaway is that the correlation coefficient of a pair with any other pair changes significantly over time, which is not clear just by looking at the correlation matrix in Figure 13.

Python Implementation

Investment Strategy

We are now ready to implement Minimum-Variance, Risk-Parity and HRP on our dataset of currency pairs. We first need to define an investment strategy that will be used across all three methods.

Investment Strategy

1. *From today's date, look at the last years' worth of historical returns. Then exponentially weight the returns with a half-life of 1 year.*
2. *Use this exponentially-weighted returns matrix to construct the covariance matrix and correlation matrix.*
3. *Run Minimum Variance/Risk Parity/HRP utilizing the appropriate matrix. This step will output weights w_i where each $w_i \geq 0$ and $\sum_{i=1}^{n} w_i = 1$. Buy the corresponding amount of each security and hold for 1 week.*
4. *After 1 week, rebalance the portfolio by running steps 1-3 again.*

Figure 15: The investment strategy used for Minimum Variance, Risk Parity and HRP.

Equally-Weighted

As a baseline, we will compare the performance of the portfolio construction techniques with a naïve equally-weighted portfolio. The code to implement an equally-weighted portfolio is extremely simple and can be seen here.

Minimum-Variance

To implement Minimum-Variance, we use the Python library *cvxpy* which can handle quadratic optimization problems. The code can be seen here.

Risk-Parity

To implement Risk-Parity, we use the Python library *scipy* which can handle non-linear optimization problems. The code can be seen here.

HRP

To implement HRP, we also use the Python library *scipy* which has built-in functionality to efficiently perform hierarchical agglomerative clustering. The library also lets us select our desired linkage criterion. The code can be seen here.

Results

Comparing Linkage Criterion

We are now ready to assess the performance of the various portfolio construction techniques. The first thing to do is compare how the three linkage criteria used for HRP (Ward, Nearest Point and Average) affect the clusters formed during hierarchical agglomerative clustering. We will then pick the linkage criterion that results in the most desirable clusters before comparing HRP with the other portfolio construction techniques.

Below, we show how the various linkage methods cluster the securities for a random holding period 2014/10/08 to 2014/10/14:

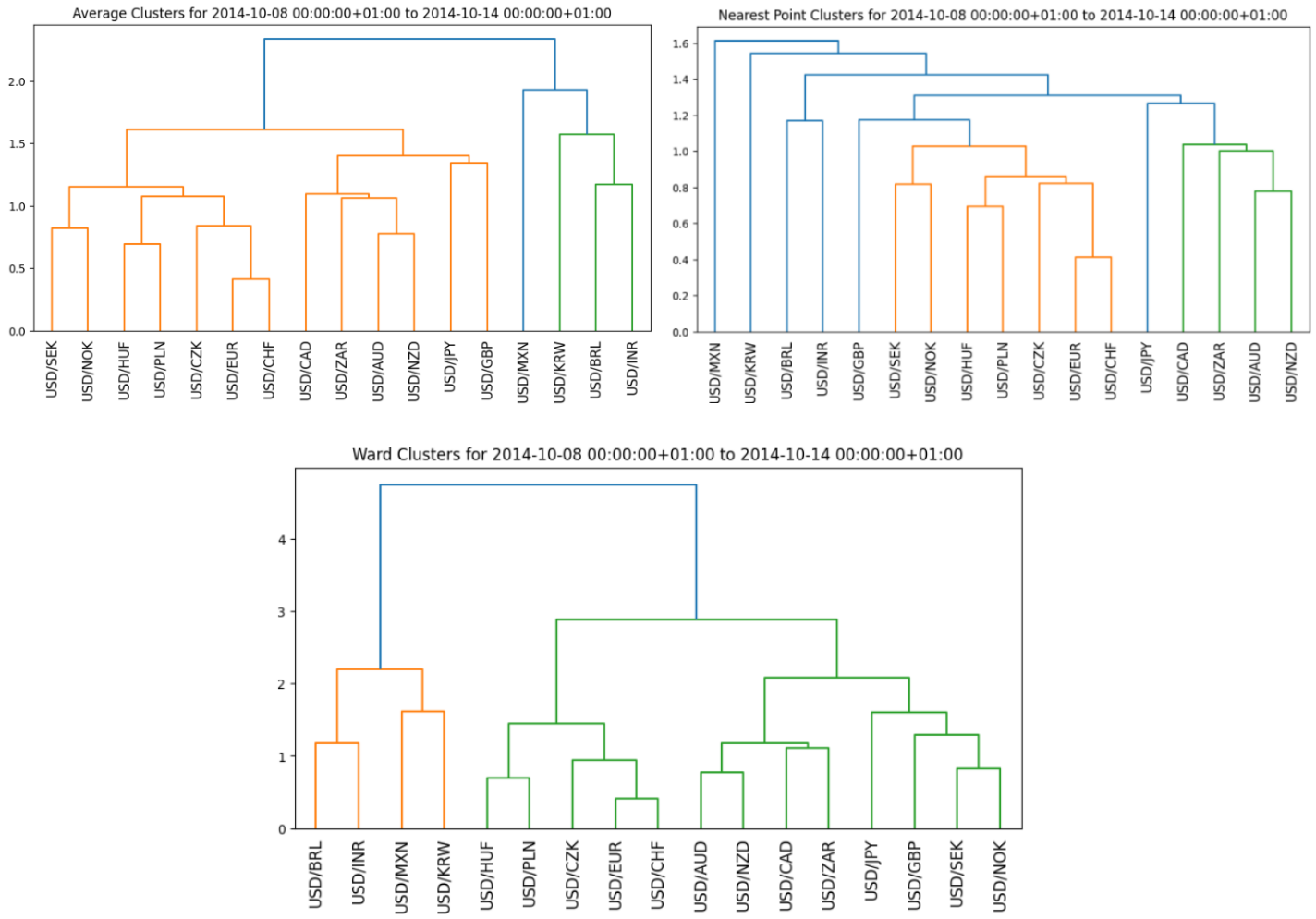


Figure 16: The results of hierarchical agglomerative clustering for HRP on a random holding period for the 'Average' (top left), 'Nearest Point' (top right) and 'Ward' (bottom) linkage criteria.

Taking a look at the hierarchical clusters, 'Ward' does the best job at demonstrating the relationships between pairs. Starting from the left of the 'Ward' diagram:

- The pairs in orange are for currencies in global emerging markets (Brazil, Indonesia, Mexico, South Korea)
- There is then a cluster of 5 European currencies. The emerging markets of Hungary and Poland get clustered together. Meanwhile, the more stable Euro and Swiss Franc get clustered, together with the Czech Koruna.
- We then have a very clear block of commodity exporters (Australia, New Zealand, Canada and South Africa) that get clustered together.
- Finally, we have a block of stable currencies from Japan, the UK, Sweden and Norway.

We will therefore use the 'Ward' linkage criterion for HRP when comparing results to other portfolio construction techniques.

Comparing Techniques

Note that the following analysis does not take the transaction costs of rebalancing into account when looking at returns.

Stability

Recall the discussion on stable solutions being desirable and how Minimum-Variance in particular can be extremely unstable due to the high condition number of the covariance matrix. This instability leads to high transaction costs and a strategy that's not practical to trade. Below, we show how the portfolio weights change over time for each strategy:

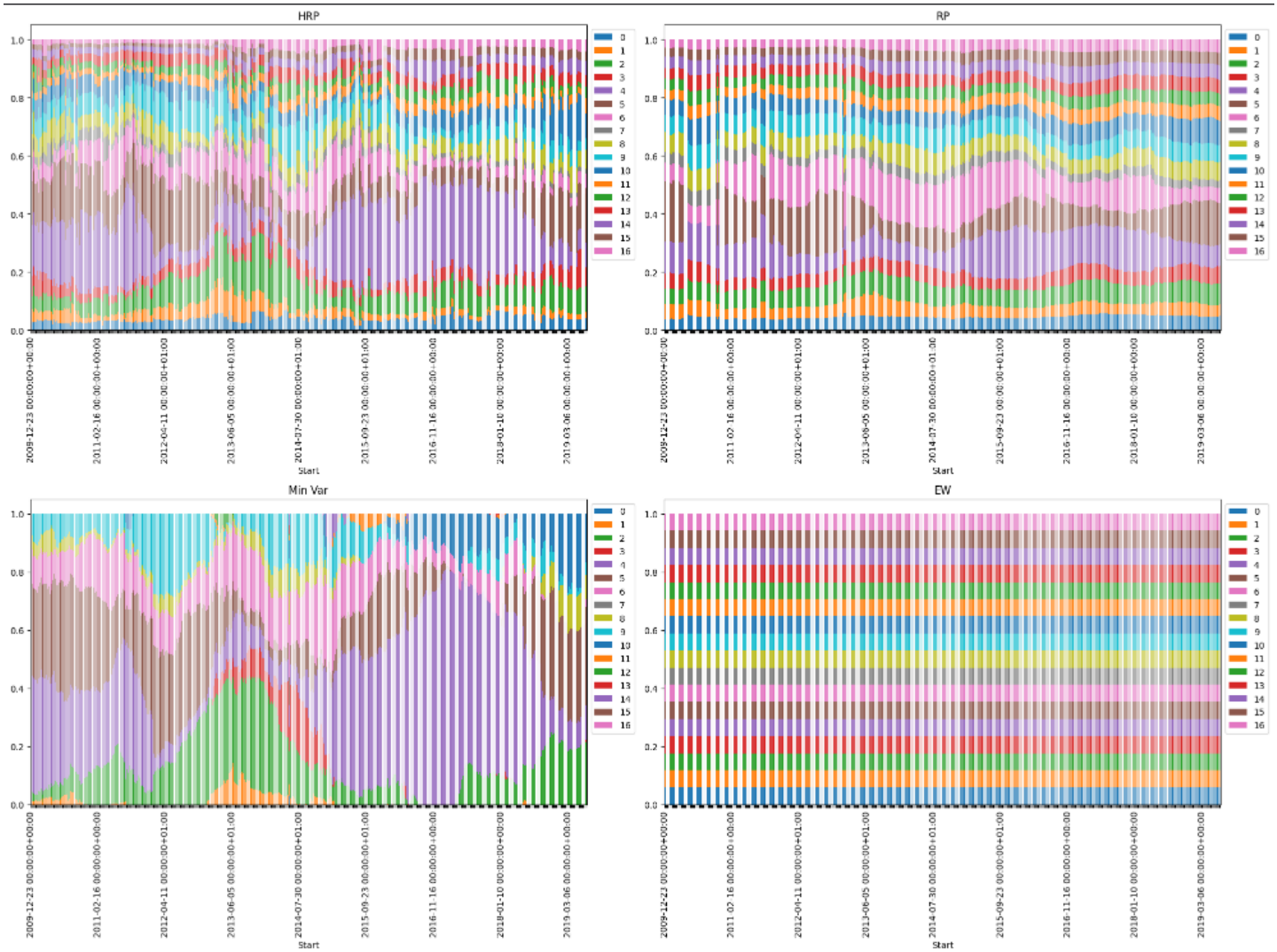


Figure 17: Portfolio weights over time for HRP (top left), Risk-Parity (top right), Minimum-Variance (bottom left), Equally-Weighted (bottom right). Notice the instability of Minimum-Variance.

Notice that the weights for Minimum-Variance change significantly over time. The portfolio can also become extremely concentrated (at some points, the security coloured purple takes up around 80% of the portfolio). Risk-Parity and HRP are far more stable and diversified as expected, making them more practical to trade.

Returns

We also look at the cumulative returns of each strategy to see how a lump sum investment would perform over time.

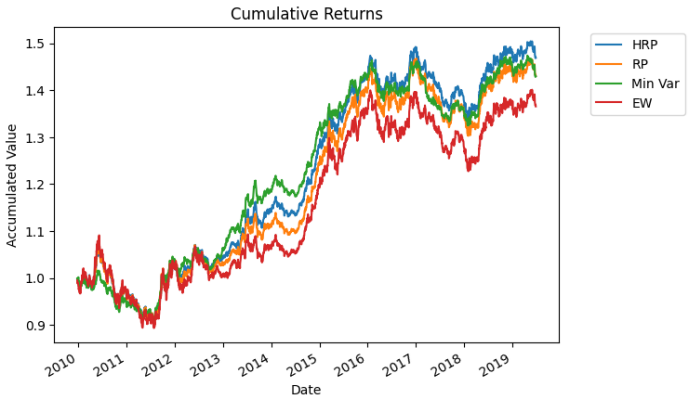


Figure 18: Cumulative returns (assuming no transaction costs) of each strategy over time. Notice that all three portfolio construction techniques outperform an equally-weighted portfolio, with HRP just about performing best.

Notice that HRP just about outperforms the other techniques. However, these returns have not been adjusted for risk. We will consider that in later sections.

Volatility

Next, we look at the annualized 1-year rolling volatility of each strategy. This is simply a 1-year rolling standard deviation of each strategy's returns, adjusted to an annual figure.

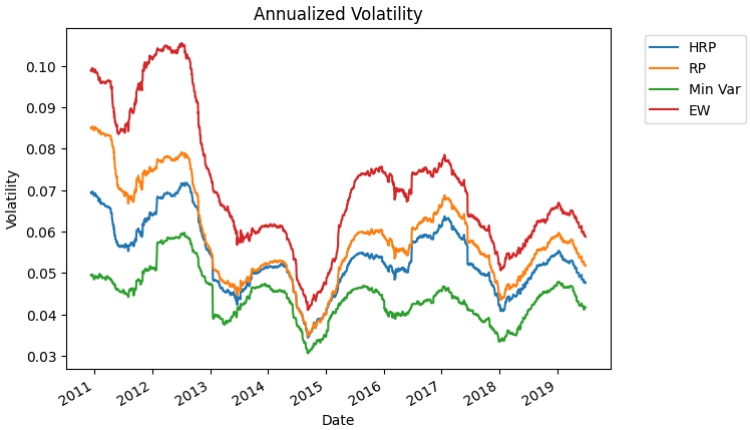


Figure 19: 1-year rolling annualized volatility of each strategy. Notice that by definition, Minimum-Variance has the lowest volatility, followed by HRP.

Minimum-Variance has the lowest volatility by definition but HRP also has a low volatility when compared to Risk-Parity. Since HRP has both higher returns and lower volatility than Risk-Parity, the risk-adjusted returns, which we measure through Sharpe ratio, will outperform Risk-Parity.

Sharpe Ratio

We also take a look at the annualized Sharpe ratio of each strategy, which measures risk-adjusted returns. If R is a random variable representing the daily returns of a strategy, r is the risk-free rate and we assume there are 252 trading days in a year, then the formula for annualized Sharpe ratio is:

$$Annualized\ Sharpe\ = \frac{\mathbb{E}[R]\ -\ r}{Std(R)}\sqrt{252}$$

Since we are comparing strategies, we can assume that $r = 0$. The annualized Sharpe ratios across the whole time period, as well as a 1-year rolling Sharpe ratio, are shown below:

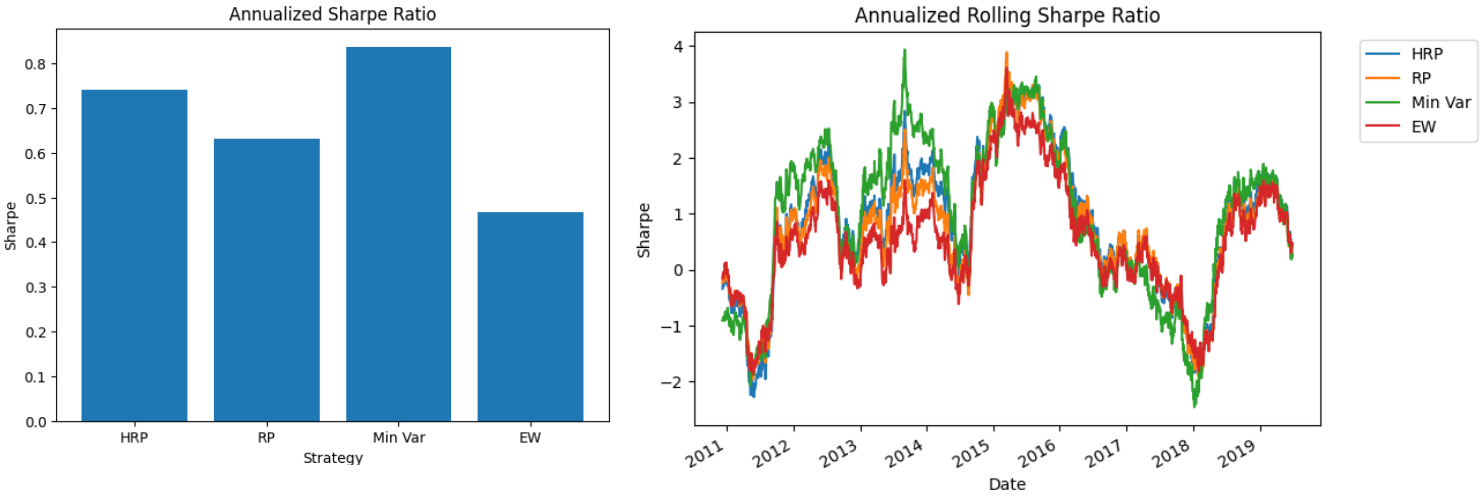


Figure 20: Left: Annualized Sharpe ratio across the whole time period. Right: Annualized 1-year rolling Sharpe ratio.

Notice that Minimum-Variance has the lowest Sharpe ratio thanks to its low volatility. Of the remaining strategies, HRP performs the best across the whole time period with a Sharpe ratio of around 0.73. For those familiar with equity strategies, this may seem like a very low Sharpe ratio but it's important to remember that that currency pairs are very slow moving compared to equities and won't generate large returns very quickly.

Conditional Value at Risk (CVaR)

Next, we look at annualized CVaR. This is an estimate with a given degree of confidence of how much (in percentage terms) can be lost by a strategy in one year. To calculate this quantity for a given strategy:

- Define a confidence interval c (e.g. $c = 0.95$)
- Look at *all values* in the $(1 - c)$th percentile of historical daily returns. These will be the smallest 5% of returns over the time period and will most likely contain negative values.
- Take the mean of those values and then multiply by $\sqrt{252}$ to annualize.

We plot annualized CVaR below for each strategy.

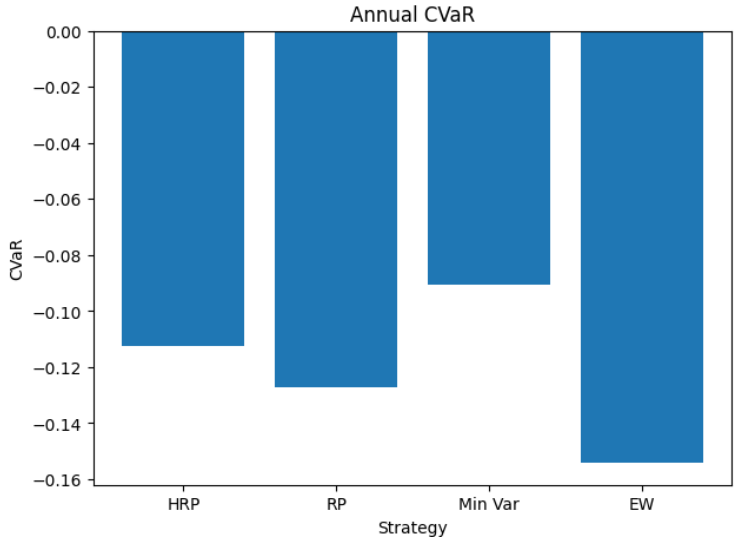


Figure 21: Annualized CVaR for each strategy. Notice that Minimum-Variance put the least value at risk as you'd expect, followed by HRP.

Drawdowns

Finally, we look at the drawdowns of each strategy. A drawdown is simply the percentage that the current cumulative value of the strategy is below its running maximum. Large and long drawdowns can massively hurt the performance of a portfolio. We plot the drawdowns for each strategy below.

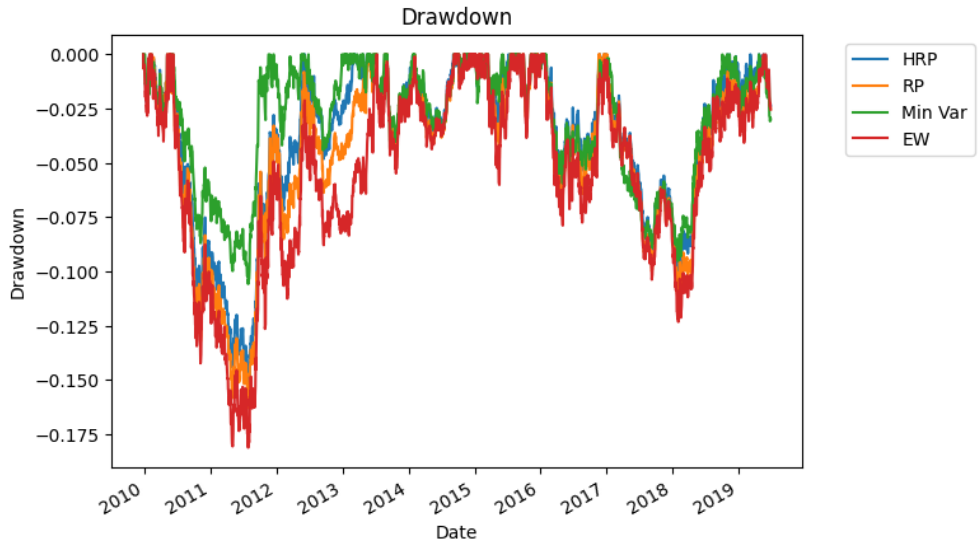


Figure 22: Drawdowns for each strategy. Notice Minimum-Variance has the smallest drawdowns as you'd expect but is once again followed by HRP.

Once again, HRP outperforms Risk-Parity with shorter and smaller drawdowns.

Recommendation

Overall, it is clear from the risk-adjusted performance of the strategies that Minimum-Variance and HRP are the two strategies that should be considered. However, while Minimum-Variance has strong risk-adjusted returns, these have been obtained via an overly-concentrated portfolio whose weights change significantly over time. In the real world, this portfolio would be poorly diversified and generate significant transaction costs.

Therefore, thanks to its smart way of allocating risk, strong risk-adjusted returns and superior stability, we conclude that HRP is the best strategy for this dataset. It would be a good exercise to apply it across a larger time period or to a different universe of securities.