WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Master Thesis

Title of Master Thesis:	Hierarchical Risk Parity (HRP) for Asset Allocation
Author (last name, first name):	Panko Aliaksandr
Student ID number:	11717357
Degree program:	Master in Quantitative Finance
Examiner (degree, first name, last name):	Assoc. Prof. Dr. Ronald Hochreiter

I hereby declare that:

1. I have written this Master thesis myself, independently and without the aid of unfair or unauthorized resources. Whenever content has been taken directly or indirectly from other sources, this has been indicated and the source referenced.
2. This Master Thesis has not been previously presented as an examination paper in this or any other form in Austria or abroad.
3. This Master Thesis is identical with the thesis assessed by the examiner.
4. (only applicable if the thesis was written by more than one author): this Master thesis was written together with

The individual contributions of each writer as well as the co-written passages have been indicated.

14/09/2019
Date

Signature

1 Abstract

This project compares the out-of-sample performance of two portfolio optimization algorithms: The Critical Line Algorithm (CLA) and Hierarchical Risk Parity (HRP) and answers the question "Does Hierarchical Risk Parity (HRP) provide lower risk out-of-sample than Critical Line Algorithm (CLA)?". The performance is measured in terms of risk-adjusted returns, total cumulative return of a portfolio and risk, represented by standard deviation of a portfolio. The research is done for two markets: American (Standard and Poor's 500 universe) and European (Euro STOXX 600 universe). Additionally, the results of two sophisticated algorithms are compared with those of the naive portfolio.

2 Introduction

The fact that investors are risk averse is widely known and adopted by the community. Investment in Treasury Bills and Treasury Notes, marketable U.S. government debt securities, carries no risk at all, however the returns are fairly low. Aiming at higher returns, investors are seeking strategies, which provide less risk. This is a general portfolio optimization problem.

One of the most significant breakthroughs in this field was the adoption of diversification and, later on, the invention of the efficient frontier by Harry Markowitz in 1952. H. Markowitz in his work "Portfolio selection" (Markowitz, H. [1952]) showed that, taking into account the correlation structure across alternative assets, one can build a diversified portfolio with a specified level of return and minimal risk. Then over the next 60 years, the portfolio optimization problem was solved using Markowitz's approach. One well-known algorithm solving the problem is the Critical Line Algorithm, which was developed by Markowitz working for the RAND Corporation in 1956 (Markowitz, H. [1956]). This algorithm solves the quadratic portfolio optimization problem efficiently, delivering weights for assets, which make a portfolio with minimal possible variance in-sample.

However, in the real world investors do not know what will happen in the future. This means that they are interested in an algorithm, which gives optimal out-of-sample weights for assets in their portfolio. It is clear that an algorithm with optimal in-sample performance is not always the one that produce optimal out-of-sample performance. Moreover, CLA has several well-known weak points that often make CLA's weights unreliable.

With the development of the machine learning field, an alternative way of risk reduction was introduced. A Developer of Hierarchical Risk Parity algorithm, Marcos Lopez de Prado, claims that his HRP algorithm provides lower risk out-of-sample. If the hypothesis is correct the asset managers and individual investors will have a new powerful tool to optimize their portfolio. Given extensive use of leverage, funds that follow this approach should benefit from adopting a more stable risk parity allocation method, thus achieving superior risk-adjusted returns and lower rebalance costs (de Prado, M. L. [2016]).

3 Theoretical Framework

3.1 Portfolio Optimization Problem

Portfolio optimization is one of the most important operations performed by asset managers. Most practitioners day-to-day need to optimize a portfolio. No investor would like to have large exposure to a couple of companies and for this reason portfolios should be diversified. Diversification means that asset weights should not be too big, resulting into a set of inequality constraints. Each constraint in the set contains a condition for a lower and an upper bound for every asset weight in a portfolio. Additionally, an equality constraint exists, meaning that all the money should be invested. This idea is represented as a condition that the weights add up to one.

This leads us to the next standard portfolio optimization problem, which is represented as a quadratic optimization problem as follows:

$$\begin{aligned} \min \quad & \frac{1}{2} w^T \Sigma w \\ \text{s.t.} \quad & 0 \leq w_i \leq u_i \\ & \sum_{i=1}^{n} w_i \mu_i = \mu_p \\ & \sum_{i=1}^{n} w_i = 1 \end{aligned}$$

where w_i is weight of asset i in a portfolio, w is a vector with assets weights, Σ is a covariance matrix of all the assets in an investment universe, $l_i = 0$, u_i are lower and upper bounds for asset's weight, μ_i is return of asset i and μ_p is return of a portfolio.

The solution to this problem is represented by a set of optimal weights w, which for the required level of portfolio return μ_p finds lower possible portfolio returns variance.

For the purposes of my research, I set up the lower bound for assets weights as 0 to prevent short selling. Short selling is normally risky and this requirement goes along with strategies of many funds that are aimed at risk avoidance.

The second important remark regarding the optimization problem is that working out-of-sample one needs to forecast individual asset returns and covariance matrix for an investment period. These two parameters are input for the problem, however they cannot be known in advance with enough precision.

3.2 Critical Line Algorithm (CLA)

The Portfolio optimization problem is a quadratic problem thus theoretically can be solved by any constrained optimization algorithm. "The Scipy library offers an optimization module called optimize, which bears five constrained optimization algorithms: The Broyden-Fletcher-Goldfarb-Shanno method (BFGS), the Truncated-Newton method (TNC), the Constrained Optimization by Linear Approximation method (COBYLA), the Sequential Least Squares Programming method (SLSQP) and the Non-Negative Least Squares solver (NNLS). Of those, BFGS and TNC are gradient-based and typically fail because they reach a boundary. COBYLA is extremely inefficient in quadratic problems, and is prone to deliver a solution outside the feasibility region defined by the constraints. NNLS does not cope with inequality constraints, and SLSQP may reach a local optimum close to the original seed provided" (de Prado, M. L. [2016]).

These examples show that general purpose optimization methods do not guarantee that the solutions they find are optimal. The reason for this issue is that such algorithms usually do not take into account the structure of the specific problem.

The Critical Line Method (CLA) is a quadratic optimization procedure that was developed by Harry Markowitz in 1956. The method was specifically designed for inequality-constrained portfolio optimization problems and guarantees that the exact solution is found after a known number of iterations, and that it ingeniously circumvents the Karush-Kuhn-Tucker conditions (Kuhn, H. W., & Tucker, A. W. [1951]). With some simple numerical improvements an implementation of original CLA significantly outperforms standard software packages in term of CPU time (de Prado, M. L. [2016]).

Despite the fact that CLA was invented almost seven decades ago and is by far more efficient than general methods; surprisingly only a small number of practitioners used this algorithm before 2013. The main reason for this paradox was the absence of open-source implementation of the algorithm. H. Markowitz initially developed and posted source code in Excel's Microsoft Visual Basic for Applications (VBA-Excel). This implementation was not convenient and there were no other open-source implementations until Bailey and Lopez de Prado in 2013 provided a Python implementation available for non-commercial usage (Bailey, D. H., & Lopez de Prado, M. [2013]).

However, there are portfolio optimization problems that cannot be represented in quadratic form. For example optimization problems that deal with skewness and kurtosis are not quadratic. Such problems cannot be solved by CLA. Moreover, CLA has several well-known disadvantages, which make CLA solutions unreliable.

The first and the most significant source of errors is returns forecasting. Returns for every asset in a universe should be estimated, however returns can rarely be forecasted with sufficient accuracy. Lack of precision in the estimation leads to huge errors. Even small deviations in the forecast of returns will cause CLA to produce significantly different portfolios (Michaud, R. O.[1989]).

Secondly, "inversion of a positive-definite covariance matrix is required, leading to large errors when the covariance matrix is numerically ill-conditioned" (Bailey, D. H., & Lopez de Prado, M. [2012]). Moreover, the greater the need for diversification the higher the chance of unstable solutions. In practice the benefits of diversification often are more than offset by estimation errors. Additionally, estimating an invertible covariance matrix of size 50 requires, at the very least, 5 years of daily independent identically distributed (IID) data, however, correlation structures normally change during such a long period of time (de Prado, M. L. [2016]).

There is a widely applied approach to reduce estimation errors in a covariance matrix. This is the Bayesian shrinkage procedure. The technique pulls the most extreme parameters toward universally constant values and in that way systematically enhance the out-of-sample performance (Jorion, P. [1986]; Ledoit, O., & Wolf, M. [2003]). In addition, Jagannathan and Ma suggested using data of higher frequency to achieve higher precision in estimators (Jagannathan, R., & Ma, T. [2003]).

Finally, CLA tends to produce highly concentrated solutions, meaning that a portfolio becomes poorly diversified and hence more risky. In real life, investors do not reshuffle portfolios much, since it would lead to huge transaction costs. However, constraints on weights can be introduced to mitigate this effect.

3.3 Hierarchical Risk Parity (HRP)

Hierarchical Risk Parity (HRP) is a portfolio optimization algorithm, which is based on risk. The algorithm generates diversified portfolios with robust out-of-sample properties without the need for a positive-definite return covariance matrix and estimation of expected returns (de Prado, M. L. [2016]).

The HRP approach addresses three major concerns of quadratic optimizers in general and Markowitz's Critical Line Algorithm (CLA) in particular:

- Instability
- Concentration
- Underperformance

HRP does not require the invertibility of the covariance matrix and returns forecasting. In this way the algorithm avoids both sources of CLA instability. Additionally, weights are allocated in the way, which solves concentration problem.

HRP applies graph theory and machine learning techniques to build a diversified portfolio based on the information contained in the correlation matrix. The biggest gap in traditional approaches is that correlation matrices lack the notion of hierarchy. This lack of hierarchical structure allows weights to vary freely in unintended ways, which is a root cause of CLA's instability. The idea of HRP is to introduce hierarchy and allow weight re-balancing only among peers at various hierarchical levels. Such allocation also means that less weight is given to similar assets. Additionally, The weights are distributed top-down, consistent with how many asset managers build their portfolios (e.g., from asset class to sectors to individual securities)(de Prado, M. L. [2016]).

It is worth mentioning that HRP is a rather fast algorithm. It solves the allocation problem in deterministic logarithm time (best case) and deterministic linear time (worst case).

3.3.1 Three Stages of HRP Algorithm

The HRP Algorithm consists of three stages:

1. Tree clustering
2. Quasi-diagonalization
3. Recursive bisection

The first step is Tree clustering. The clustering combines assets into a hierarchical structure of clusters, so that allocations can flow downstream through a tree graph. This step could be visualized via a dendogram:

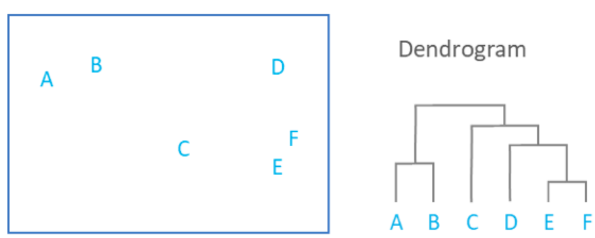

Figure 1: Hierarchical Structure

The first step, clustering, consists of several significant substeps:

1. Compute a correlation $N \times N$ matrix for stock returns with entries $\rho = \rho_{i,j}$, where $\rho_{i,j} = \rho[X_i, X_j]; i, j = 1...N$
2. Define a distance measure $d : (X_i, X_j) \subset B \to R \in [0 : 1], d_{i,j} = d[X_i, X_j] = \sqrt{\frac{1}{2}(1 - \rho_{i,j})}$, where B is the Cartesian product of items.
3. Compute a distance matrix $D = d_{i,j}; i, j = 1...N$
4. Compute the Euclidean distance between any two column-vectors of the distance matrix $\bar{d}_{i,j} = \sqrt{\sum_{n=1}^{N}(d_{n,i} - d_{n,j})^2}$
5. Cluster together the pair of assets taking ones with minimal distance $(i^*, j^*) = argmin(i,j)_{i \neq j}\bar{d}_{i,j}$
6. Recalculate distance between a newly formed cluster and the single (unclustered) items and update distance matrix. At this moment one can use difference definitions of distance between the cluster and other items. In my research I used the original approach of Lopez de Prado and defined distance as $d_{i,cluster} = min[\bar{d}_{i,j}]$, where $j \in cluster$
7. Repeat the procedure until the final cluster contains all of the original items

On the second step similar investments are placed together, and dissimilar investments are placed far apart. Quasi-diagonalization reorganizes the rows and columns of the covariance matrix, so that the largest values lie along the diagonal. Clusters are replaced with their constituents recursively, until no clusters remain. These replacements preserve the order of the clustering. The output is a sorted list of original (unclustered) items.

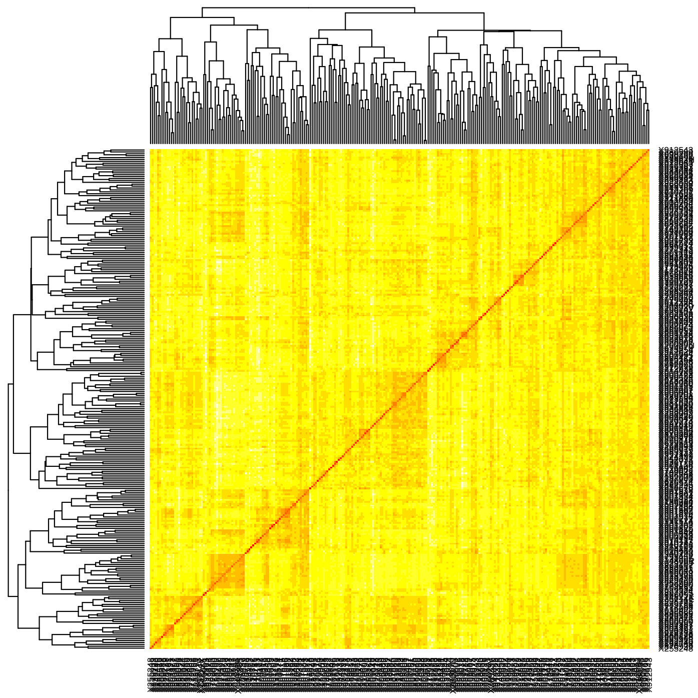

Figure 2: Quasi-Diagonalization Result

When quasi-diagonalization is done, all assets are reordered so as to group similar assets together. Next the algorithm executes recursive bisection to reallocate weights. This step consists of the following:

1. The allocation process starts with assigning unit weights to all assets and putting all the assets in one group L.

2. On each iteration, the algorithm recursively splits a group into 2 groups $L_1 \cup L_2 = L, |L_1| = int[\frac{1}{2}|L|]$, saving the initial order, until the group contains at least 2 elements. After the split the algorithm calculates the group variance as a quadratic form $\bar{V} = w'Vw$ for each subgroup, where V is the covariance matrix between the constituents of a corresponding group, $w = diag[V]^{-1} \frac{1}{tr[diag[V]^{-1}]}$, where diag[.] and tr[.] are the diagonal and trace operators.

3. After 2 subgroups are formed, a weights split factor α is computed as follows: $\alpha = 1 - \frac{\bar{V}^1}{\bar{V}^1 + \bar{V}^2}$. Here $\bar{V}^1$ and $\bar{V}^2$ are subgroups' group variance computed in 2. above.

4. Rescale current allocations in the first group by the factor of α_i

5. Rescale current allocations in the second group by the factor of $(1 - \alpha_i)$

6. Loop to step 2

4 Methodology

4.1 Data Structure

In the research two data sets are used. The first data set represents the US Stock Market, while the second one corresponds to the European Markets. The US stock market is represented by Standard and Poor's 500 index (S&P500). The European stock markets are represented by STOXX Europe 600 Index (SXXP).

To account for periods of recession, the research interval from 2003-06-01 to 2018-05-31 was chosen. This period includes one of the most dramatic crises and one of the longest expansion periods. Such choice makes the period reasonable enough for making conclusions.

Both data sets are combined as follows: a stock is included in a data set only if it was listed in an index during the whole period. This means that both data sets represent an intersection of all stocks, which were in the corresponding index during 2003-2018.

This leads to the first question, whether the number of stocks in the intersection is big enough. The table below states that the number of stocks in the intersection for the US market is 254, while for European market this number is 263. Both sample sizes are sufficient for the research.

The second concern could be connected with the characteristics of chosen stocks. Since all the stocks survived in the index for such a long period, one can conclude that they must be blue-chip stocks. This fact makes the research data set in some way limited, however it is an absolutely adequate choice. Taking into consideration the fact that both algorithms are supposed to deliver the lowest possible risk and the investment area (funds that invest mostly in blue-chip stocks), it is clear that this limitation is not significant.

The third question could be connected with diversification possibility. The question "Is diversification in described universe still possible?" should be answered. The minimum value of the correlation could be used to get the answer. The table below shows that the minimum correlation is even negative meaning that diversification is still possible.

Stocks weekly returns have been downloaded from Datastream, which is a world-wide known and reliable data provider.

	# Assets	# Obs.	StartDate	EndDate	# NA	Corr. min
USA	254.00	782.00	2003-06-01	2018-05-31	0.00	-0.01
Europe	263.00	782.00	2003-06-01	2018-05-31	0.00	0.03

4.2 Portfolio Creation Mechanism

Since the hypothesis claims better out-of-sample performance, a portfolio structure has to be introduced before the actual investment is done. For this purpose the rolling window approach is normally used. The usage of rolling window implies that train and test time interval should be chosen. However, this is one of the most difficult parts, since correlation structure changes over time. If the period under consideration is too long, one is exposed to underestimate significant updates in the correlation. Conversely, too short a period too short period leads to considerable increase in the estimation error. My data sets contain approximately 15 years of data and 5 of them are before before the crisis in 2008. I decided to take 3-years rolling window as a train data set and reshuffle a portfolio quarterly. According to Anne M.Tucker, the average holding period for all funds was in the range of 15 to 17 months, which makes this type of reshuffle reasonable (Tucker, A. M. [2017]).

The second critical step is estimation of expected returns for CLA algorithm. The literature review showed that APT (Yli-Olli, P., & Virtanen, I. [1992]) and CAPM (Jorion, P. [1985]; Grauer, R. R., & Hakansson, N. H. [1995]) models does not provide sufficiently good results here, so simple stock return means are used instead.

Both algorithms provide optimal weights, which are used to create an optimal portfolio. On each iteration these weights are recorded to calculate out-of-sample portfolio returns and standard deviations.

4.3 Implementation

The implementation of both algorithms is done using R language, which provides a powerful set of functions for data analysis and incredible visualization packages.

The algorithm contains five significant parts:

1. Data download
2. CLA implementation
3. HRP implementation
4. Results analysis
5. Visualization

The required data are downloaded using **read.csv()** function, which takes a .csv file as an argument and returns a data frame object.

For CLA implementation **CLA** package Version 0.95-1 developed by Yanhao Shi and Martin Maechler is used. The detailed documentation can be found here. The function CLA() has several important arguments:

- mu is a numeric vector, containing the expected returns for the assets
- covar is a covariance matrix of assets returns, must be positive definite
- lB, uB are vectors that contain lower and upper bounds for the asset weights.

As was mentioned previously, expected returns are estimated as simple historical means. The covariance matrix is estimated by R **cov()** function applied to asset returns during the rolling window period.

The lower bound is set to 0, to prevent short selling and the upper bound is set to 0.2 to reduce well-known CLA concentration problem.

HRP implementation is done following the approach described in Lopez de Prado's paper in 2016 (de Prado, M. L. [2016]). For the detailed step implementation, please refer to the appendix section.

The results are described by three parameters: portfolio standard deviation, portfolio cumulative return and risk-adjusted returns measured as return-to-standard deviation ratio. For these purposes two function from package **PerformanceAnalytics** are used.

The first function is **StdDev()**. The main two parameters of the function are a time series object of asset returns and weights. The function returns the portfolio standard deviation during analyzed time period. This means that running this function for a rolling window and using weights from the previous step (to make the result out-of-sample), the required standard deviation for a portfolio can be recorded at each step. Please refer the documentation for the further details.

The second function is **Return.portfolio()**. Using a time series of returns and any regular or irregular time series of weights for each asset, this function calculates the returns of a portfolio with the same periodicity of the returns data. The function has several parameters: a time series object of asset returns, a time series containing asset weights, as decimal percentages, treated as beginning of period weights, type of return (simple or geometric) and, finally, rebalance parameter. Please refer the documentation for the further details. Geometric returns and quartarly rebalancing are used in the research.

5 Results

5.1 USA

The analysis of the results starts with weight distribution. Both algorithms produce optimal weights, by which funds should be allocated when forming a portfolio. The pictures below show the number of assets in a portfolio with non-zero weights according to HRP and CLA.

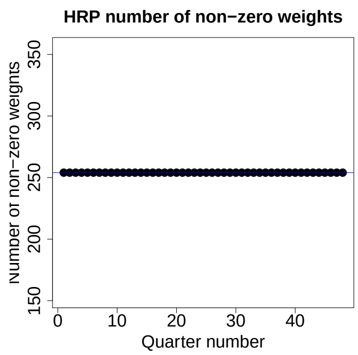

Figure 3: USA: HRP number of non-zero weights

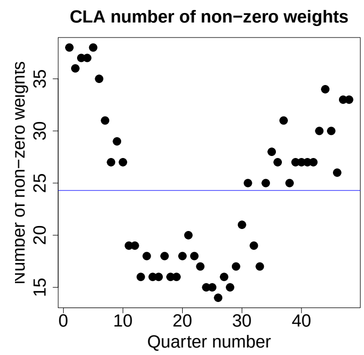

Figure 4: USA: CLA number of non-zero weights

The graphs show first significant finding: HRP allocates money to all assets in a universe, while CLA distributes weight among around 25 assets on average with distribution from about 15 to 35. This points out at CLA's concentration problem from one side and extreme weight distribution by HRP from another side.

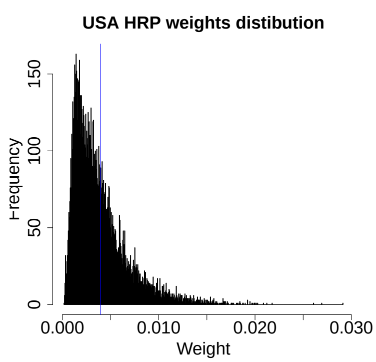

Figure 5: USA HRP weights distribution

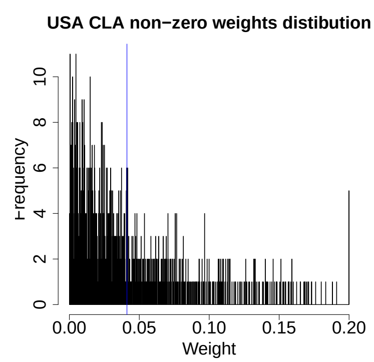

Figure 6: USA CLA non-zero weights distribution

It is not now a surprise that in general non-zero weights from CLA are approximately 10 times greater than ones provided by HRP. Actually, one can even say that HRP weights are too small.

These distributions suggest that to use both algorithms more efficiently one should pay attention to a universe, including and excluding assets that were previously know to be inappropriate.

In case of CLA one can reduce the upper bound to increase the number of assets in a portfolio, but this will lead to an increased number of assets with boundary weight.

This type of weights distribution could lead one to compare performance HRP with a simple equally weighted portfolio with weights $\frac{1}{N}$, where N is a number of assets. This idea was incorporated and the research represents results for equally weighted portfolio too.

First, the cumulative portfolio return is analyzed. The results for three portfolios are represented below.

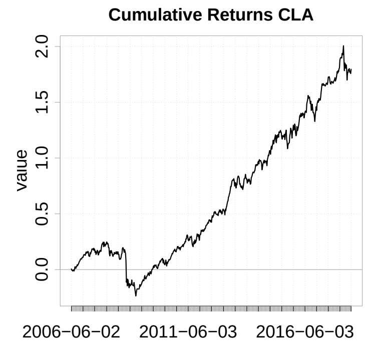

Figure 8: USA: Cumulative Returns CLA

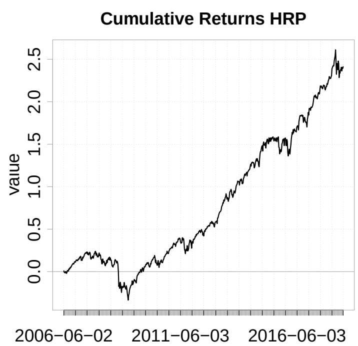

Figure 7: USA: Cumulative Returns HRP

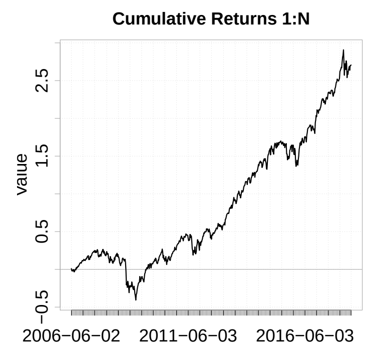

Figure 9: USA: Cumulative Returns CLA

In terms of performance, HRP showed approximately the same result as the naive approach and a 25% better result than CLA in the US market. This difference is more than significant, however, to make any conclusion risk should be also compared.

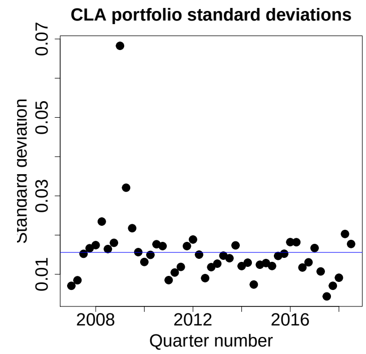

Figure 11: USA: CLA portfolio standard deviations

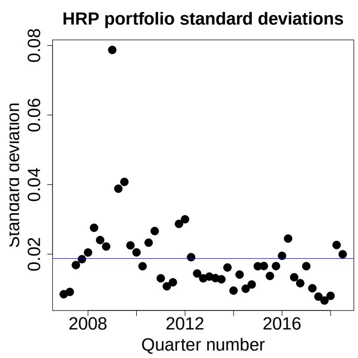

Figure 10: USA: HRP portfolio standard deviations

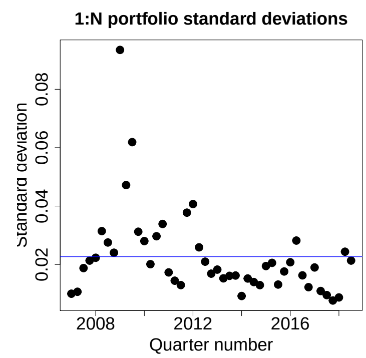

Figure 12: USA: 1:N portfolio standard deviations

Regarding risk, represented by standard deviation, the naive portfolio gives the highest risk, following by HRP, which resulted in slightly higher values than CLA. This result reflects the risk-return law, which states that higher returns are caused by higher risk. To make a final conclusion, the return-risk ratio should be analyzed.

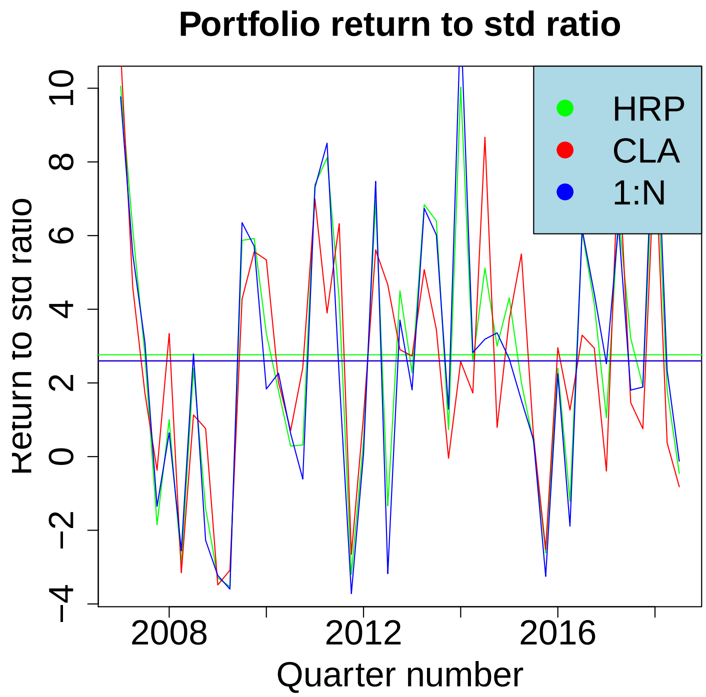

Figure 13: USA: Return to risk ratio

In 55.32 % of cases HRP outperforms CLA, in 55.32 % of cases HRP outperforms one-over-N and in 44.68 % of cases CLA outperforms one-over-N! This result means that in the US market HRP outperforms CLA and the naive portfolio. Surprisingly, the naive portfolio showed better results than CLA!

5.2 Europe

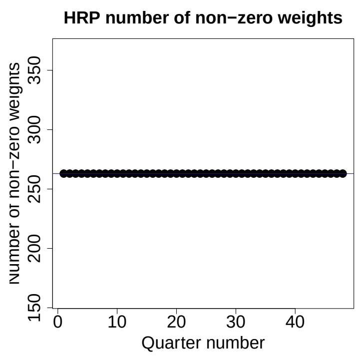

Figure 14: Europe: HRP number of non-zero weights

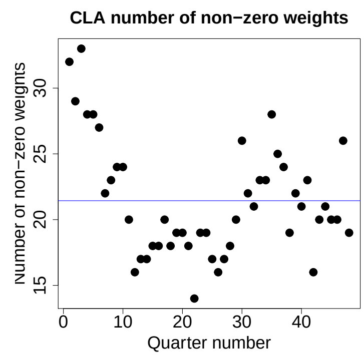

Figure 15: Europe: CLA number of non-zero weights

For Europe the picture is the same. As was mentioned before, HRP allocates weights to all the assets in a universe and CLA still produces highly concentrated solutions.

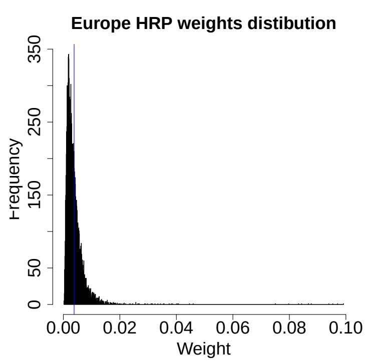

Figure 16: Europe: HRP weights distribution

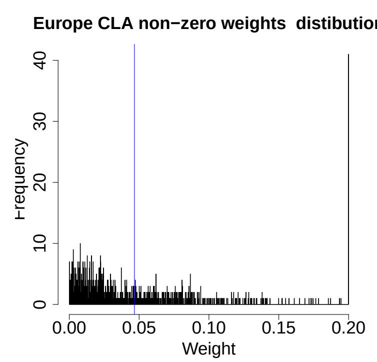

Figure 17: Europe: CLA non-zero weights distribution

Compared to the US, Europe has much more extreme weights. In fact, CLA results assigned the upper bound weight of 20% much more frequently than all other non-zero weights. HRP also in several cases produced relatively big weights, suggesting the idea that European market significantly differs from the US one.

When analyzing cumulative returns for European markets, the picture is a bit different.

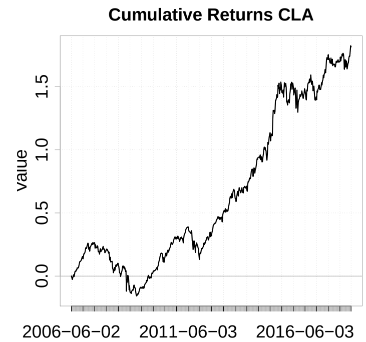

Figure 19: Europe: Cumulative Returns CLA

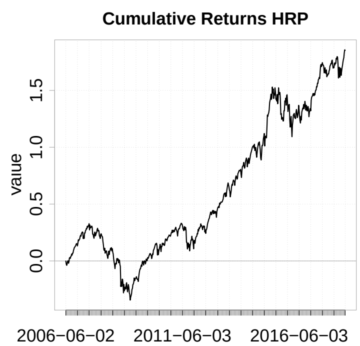

Figure 18: Europe: Cumulative Returns HRP

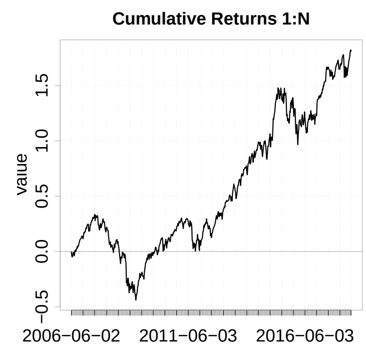

Figure 20: Europe: Cumulative Returns 1:N

In contrast to American market, in Europe all three algorithms performed more or less the same in terms of cumulative return.

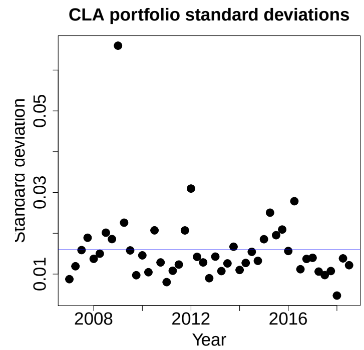

Figure 22: Europe: CLA portfolio standard deviations

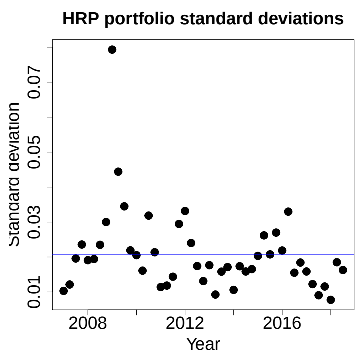

Figure 21: Europe: HRP portfolio standard deviations

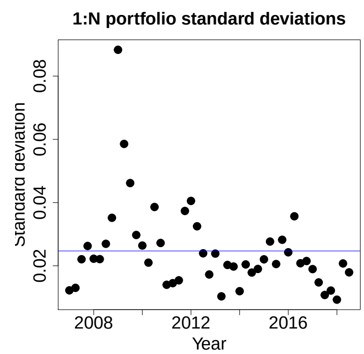

Figure 23: Europe: 1:N portfolio standard deviations

Regarding portfolio standard deviation, the results for the European markets show the same pattern as the US market: the naive portfolio is more volatile than HRP, which is in it's turn more volatile than CLA.

Finally, the return-to-risk ratio is analyzed. The results are represented below.

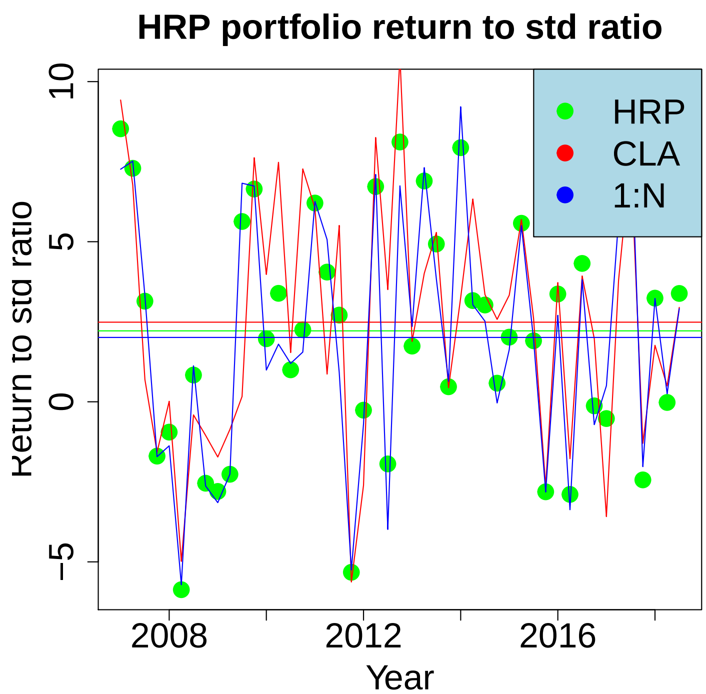

Figure 24: Europe: Return to risk ratio

Overall, in 36.17 % of cases HRP outperforms CLA, in 57.45 % of cases HRP outperforms one-over-N and in 65.96 % of cases CLA outperforms one-over-N. Here HRP was not able to outperform CLA and the naive portfolio showed significantly worse result as HRP and CLA

6 Conclusion

According to "Building Diversified Portfolios that Outperform Out-of-Sample" (de Prado, M. L. [2016]) HRP shows better out-of-sample performance. That claim was based on Monte Carlo simulation and generated data set. The real data, however, showed different results.

First of all, my research could not support the claim that HRP provides lower out-of-sample variance. Furthermore, HRP in both markets showed on average higher standard deviation.

Secondly, taking into consideration risk-adjusted returns, which probably has more sense, the results depend on market. In the US market HRP, indeed, outperformed CLA, however, in the European markets CLA produces better results.

Surprisingly, the naive portfolio performed well in the American market - outperforming CLA - while, in the European markets the naive portfolio performed poorly.

Overall, the HRP methodology is worthwhile for forming portfolios. However, it would be untrue to say that this approach constantly outperforms CLA. Moreover, it is important to mention that there are many factors that could influence the final result, namely: method of forecasting returns and its accuracy, covariance structure of assets in a universe, upper and lower bound values in CLA, the method of defining distance measure in HRP. Furthermore, in the research a portfolio is reshuffled quarterly and a 3-years rolling window is used, while it is possible to use another frequency and another time period instead.

7 References

1. Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7(1), 77-91

2. Markowitz, H. (1956). The optimization of a quadratic function subject to linear constraints. Naval research logistics Quarterly, 3(1-2), 111-133.

3. de Prado, M. L. (2016). Building diversified portfolios that outperform out of sample. The Journal of Portfolio Management, 42(4), 59-69.

4. Markovitz, H. (1959). Portfolio selection: Efficient diversification of investments. NY: John Wiley.

5. Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear programming, in (J. Neyman, ed.) Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability.

6. Bailey, D. H., & Lopez de Prado, M. (2013). An open-source implementation of the critical-line algorithm for portfolio optimization. Algorithms, 6(1), 169-196.

7. Michaud, R. O. (1989). The Markowitz optimization enigma: Is 'optimized'optimal?. Financial Analysts Journal, 45(1), 31-42.

8. Bailey, D. H., & Lopez de Prado, M. (2012). Balanced baskets: a new approach to trading and hedging risks. Journal of Investment Strategies (Risk Journals), 1(4).

9. Jorion, P. (1986). Bayes-Stein estimation for portfolio analysis. Journal of Financial and Quantitative Analysis, 21(3), 279-292.

10. Ledoit, O., & Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. Journal of Empirical Finance, 10(5), 603-621.

11. Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. The Journal of Finance, 58(4), 1651-1683.

12. Tucker, A. M. (2017). The Long and the Short: Portfolio Turnover Ratios & Mutual Fund Investment Time Horizons. J. Corp. L., 43, 581.

13. Yli-Olli, P., & Virtanen, I. (1992). Some empirical tests of the arbitrage pricing theory using transformation analysis. Empirical Economics, 17(4), 507-522.

14. Jorion, P. (1985). International portfolio diversification with estimation risk. Journal of Business, 259-278.

15. Grauer, R. R., & Hakansson, N. H. (1995). Stein and CAPM estimators of the means in asset allocation. International Review of Financial Analysis, 4(1), 35-66.

8 Appendix

```
library(xts)
library(lubridate)
library(CLA)
library(PerformanceAnalytics)
library(xtable)
library(graphics)

# Global Constants
wd <- "D:/ESG Results"
start_date <- as.Date("2003-06-01")
end_date <- as.Date("2018-06-01")
train_end_date <- as.Date("2006-06-01")

# Functions
correlDist <- function(corr){
  value = sqrt(1/2. * (1 - corr))
  # Default euqlidian as required
  dist = dist(value)
  return(dist)
}
getIVP <- function(covMat) {
  invDiag <- 1/diag(as.matrix(covMat))
  weights <- invDiag/sum(invDiag)
  return(weights)
}
getClusterVar <- function(covMat, cItems) {
  covMatSlice <- covMat[cItems, cItems]
  weights <- getIVP(covMatSlice)
  cVar <- t(weights) %*% as.matrix(covMatSlice) %*% weights
  return(cVar)
}
getRecBipart <- function(covMat, sortIx) {
  w <- rep(1,ncol(covMat))
```

```r
  w <- recurFun(w, covMat, sortIx)
  return(w)
}
recurFun <- function(w, covMat, sortIx) {
  subIdx <- 1:trunc(length(sortIx)/2)
  cItems0 <- sortIx[subIdx]
  cItems1 <- sortIx[-subIdx]
  cVar0 <- getClusterVar(covMat, cItems0)
  cVar1 <- getClusterVar(covMat, cItems1)
  alpha <- 1 - cVar0/(cVar0 + cVar1)

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

  if(length(cItems0) > 1) {
    w <- recurFun(w, covMat, cItems0)
  }
  if(length(cItems1) > 1) {
    w <- recurFun(w, covMat, cItems1)
  }
  return(w)
}
visualize <- function(
                        USA_CLA_nonzero_num,
                        USA_HRP_nonzero_num,
                        USA_HRP_sd_v,
                        USA_CLA_sd_v,
                        USA_one_n_sd_v,
                        USA_HRP_portfolio_return,
                        USA_CLA_portfolio_return,
                        USA_one_n_portfolio_return,
                        USA_HRP_ret_over_sd,
                        USA_CLA_ret_over_sd,
                        USA_one_n_ret_over_sd,
                        Europe_CLA_nonzero_num,
                        Europe_HRP_nonzero_num,
                        Europe_HRP_portfolio_return,
                        Europe_CLA_portfolio_return,
                        Europe_one_n_portfolio_return,
                        Europe_one_n_sd_v,
                        Europe_HRP_sd_v,
                        Europe_CLA_sd_v,
                        Europe_HRP_ret_over_sd,
                        Europe_CLA_ret_over_sd,
                        Europe_one_n_ret_over_sd
){


  plot(USA_HRP_nonzero_num, xlab = "Quarter number", ylab = "Number of non-zero weights",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
  title("HRP number of non-zero weights",cex.main=2)
  abline(h = mean(USA_HRP_nonzero_num), col = 'blue')

  plot(USA_CLA_nonzero_num,  xlab = "Quarter number", ylab = "Number of non-zero weights",cex.lab=2, cex.axis=2, cex.main=2,
```

```
  cex.sub=2, pch = 19, cex = 2)
title("CLA number of non-zero weights",cex.main=2)
abline(h = mean(USA_CLA_nonzero_num), col = 'blue')

hist(as.matrix(USA_HRP_weights), xlab = "Weight",  main = "USA HRP weights distibution",breaks = 1000, cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
abline(v = mean(as.matrix(USA_HRP_weights)), col = 'blue')

a = as.matrix(USA_CLA_weights)
b = a[a != 0]
hist(b, xlab = "Weight",  main = "USA CLA non-zero weights distibution",breaks = 1000, cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
abline(v = mean(as.matrix(b)), col = 'blue')

chart.CumReturns(USA_HRP_portfolio_return$return, main="Cumulative Returns HRP", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)

chart.CumReturns(USA_CLA_portfolio_return$returns, main="Cumulative Returns CLA", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)

chart.CumReturns(USA_one_n_portfolio_return$returns, main="Cumulative Returns 1/N", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)

plot(y = USA_HRP_sd_v,x = names(USA_HRP_sd_v), xlab = "Quarter number", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("HRP portfolio standard deviations",cex.main=2)
abline(h = mean(USA_HRP_sd_v), col = 'blue')


plot(y = USA_CLA_sd_v, x = names(USA_CLA_sd_v),  xlab = "Quarter number", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("CLA portfolio standard deviations",cex.main=2)
abline(h = mean(USA_CLA_sd_v), col = 'blue')

plot(y = USA_one_n_sd_v, x = names(USA_one_n_sd_v),  xlab = "Quarter number", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("1:N portfolio standard deviations",cex.main=2)
abline(h = mean(USA_one_n_sd_v), col = 'blue')


plot(y = USA_HRP_ret_over_sd,x = names(USA_HRP_sd_v), xlab = "Quarter number", ylab = "Return to std ratio",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "green")
title("Portfolio return to std ratio",cex.main=2)
abline(h = mean(USA_HRP_ret_over_sd), col = 'green')
# Calculate standard deviation of returns
lines(y = USA_CLA_ret_over_sd, x = names(USA_CLA_sd_v),  xlab = "Quarter number", ylab = "Return to std ratio",cex.lab=2, cex.
```

```
axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "red")
abline(h = mean(USA_CLA_ret_over_sd), col = 'red')

lines(y = USA_one_n_ret_over_sd, x = names(USA_one_n_sd_v),  xlab = "Quarter number", ylab = "Return to std ratio",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "blue")
abline(h = mean(USA_one_n_ret_over_sd), col = 'blue')

# Add a legend
# Add extra space to right of plot area; change clipping to figure
par( xpd=TRUE)
legend("topright", legend=c("HRP", "CLA","1:N"), col=c("green", "red", "blue"),box.lty=0, cex=2,pch=19, bg='lightblue')
l = length(USA_CLA_ret_over_sd)

U_percent_HRP_g_CLA =  round(sum(USA_HRP_ret_over_sd > USA_CLA_ret_over_sd) / l * 100, 2)
U_percent_HRP_g_one_n =  round(sum(USA_HRP_ret_over_sd > USA_one_n_ret_over_sd) / l * 100, 2)
U_percent_CLA_g_one_n =  round(sum(USA_CLA_ret_over_sd > USA_one_n_ret_over_sd) / l * 100, 2)

# Europe
plot(Europe_HRP_nonzero_num, xlab = "Quarter number", ylab = "Number of non-zero weights",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("HRP number of non-zero weights",cex.main=2)
abline(h = mean(Europe_HRP_nonzero_num), col = 'blue')

plot(Europe_CLA_nonzero_num,  xlab = "Quarter number", ylab = "Number of non-zero weights",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("CLA number of non-zero weights",cex.main=2)
abline(h = mean(Europe_CLA_nonzero_num), col = 'blue')

# Europe
hist(as.matrix(Europe_HRP_weights), xlab = "Weight", main = "Europe HRP weights distibution",breaks = 1000, cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
abline(v = mean(as.matrix(Europe_HRP_weights)), col = 'blue')

# Europe
a = as.matrix(Europe_CLA_weights)
b = a[a != 0]
hist(b, xlab = "Weight", main = "Europe CLA non-zero weights distibution",breaks = 1000, cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
abline(v = mean(as.matrix(b)), col = 'blue')

chart.CumReturns(Europe_HRP_portfolio_return$return, main="Cumulative Returns HRP", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
chart.CumReturns(Europe_CLA_portfolio_return$returns, main="Cumulative Returns CLA", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)
```

```r
chart.CumReturns(Europe_one_n_portfolio_return$returns, main="Cumulative Returns 1/N", cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2)

# HRP Calculate standard deviation of returns
plot(y = Europe_HRP_sd_v,x = names(Europe_HRP_sd_v), xlab = "Year", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("HRP portfolio standard deviations",cex.main=2)
abline(h = mean(Europe_HRP_sd_v), col = 'blue')

# CLA Calculate standard deviation of returns
plot(y = Europe_CLA_sd_v, x = names(Europe_CLA_sd_v),  xlab = "Year", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("CLA portfolio standard deviations",cex.main=2)
abline(h = mean(Europe_CLA_sd_v), col = 'blue')

# 1/N Calculate standard deviation of returns
plot(y = Europe_one_n_sd_v, x = names(Europe_one_n_sd_v),  xlab = "Year", ylab = "Standard deviation",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2)
title("1:N portfolio standard deviations",cex.main=2)
abline(h = mean(Europe_one_n_sd_v), col = 'blue')

# Calculate standard deviation of returns
plot(y = Europe_HRP_ret_over_sd,x = names(Europe_HRP_sd_v), xlab = "Year", ylab = "Return to std ratio",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "green")
title("HRP portfolio return to std ratio",cex.main=2)
abline(h = mean(Europe_HRP_ret_over_sd), col = 'green')
# Calculate standard deviation of returns
lines(y = Europe_CLA_ret_over_sd, x = names(Europe_CLA_sd_v), xlab = "Year", ylab = "Return to std ratio",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "red")
abline(h = mean(Europe_CLA_ret_over_sd), col = 'red')
lines(y = Europe_one_n_ret_over_sd, x = names(Europe_one_n_sd_v),   xlab = "Year", ylab = "Return to std ratio",cex.lab=2, cex.axis=2, cex.main=2, cex.sub=2, pch = 19, cex = 2, col = "blue")
abline(h = mean(Europe_one_n_ret_over_sd), col = 'blue')
# Add a legend
par( xpd=TRUE)
legend("topright", legend=c("HRP", "CLA", "1:N"), col=c("green", "red", "blue"),box.lty=0, cex=2,pch=19, bg='lightblue')
l = length(Europe_CLA_ret_over_sd)

E_percent_HRP_g_CLA =  round(sum(Europe_HRP_ret_over_sd > Europe_CLA_ret_over_sd) / l * 100, 2)
E_percent_HRP_g_one_n =  round(sum(Europe_HRP_ret_over_sd > Europe_one_n_ret_over_sd) / l * 100, 2)
E_percent_CLA_g_one_n =  round(sum(Europe_CLA_ret_over_sd > Europe_one_n_ret_over_sd) / l * 100, 2)


return_list <- list(U_percent_HRP_g_CLA,
                    U_percent_HRP_g_one_n,
                    U_percent_CLA_g_one_n,
```

```
                         E_percent_HRP_g_CLA,
                         E_percent_HRP_g_one_n,
                         E_percent_CLA_g_one_n)
  return(return_list)
}

file_names <- c("required_stocks_USA.csv", "required_stocks_Europe.
    csv")

# Final Data Frames

# Non-zero number
# USA
USA_CLA_nonzero_num <- data.frame()
USA_HRP_nonzero_num <- data.frame()
# Europe
Europe_CLA_nonzero_num <- data.frame()
Europe_HRP_nonzero_num <- data.frame()

# Weights
# USA
USA_CLA_weights <- data.frame()
USA_HRP_weights <- data.frame()
# Europe
Europe_CLA_weights <- data.frame()
Europe_HRP_weights <- data.frame()

# Portfolio Cumulative Return
# USA
USA_HRP_portfolio_return <- data.frame()
USA_CLA_portfolio_return <- data.frame()
USA_one_n_portfolio_return <- data.frame()
# Europe
Europe_HRP_portfolio_return <- data.frame()
Europe_CLA_portfolio_return <- data.frame()
Europe_one_n_portfolio_return <- data.frame()

# Standard Deviations
# USA
USA_HRP_sd_v <- c()
USA_CLA_sd_v <- c()
USA_one_n_v <- c()
# Europe
Europe_HRP_sd_v <- c()
Europe_CLA_sd_v <- c()
Europe_one_n_v <- c()

# Portfolio returns for a quarter
# USA
USA_CLA_portfolio_return_q_v <- c()
USA_HRP_portfolio_return_q_v <- c()
USA_one_n_portfolio_return_q_v <- c()
# Europe
Europe_CLA_portfolio_return_q_v <- c()
Europe_HRP_portfolio_return_q_v <- c()
Europe_one_n_portfolio_return_q_v <- c()

```

```r
# Risk-adjusted returns quarterly
# USA
USA_CLA_ret_over_sd <- c()
USA_HRP_ret_over_sd <- c()
USA_one_n_ret_over_sd <- c()
# Europe
Europe_CLA_ret_over_sd <- c()
Europe_HRP_ret_over_sd <- c()
Europe_one_n_ret_over_sd <- c()

# The same algorithm is executed for USA and Europe
  for(j in 1:2){
# Step 1: Prepare data
  intersection_df <- read.csv(file = paste0(wd,"/", file_names[j]), header = TRUE)
  rownames(intersection_df) <- intersection_df$date
  intersection_df <- subset(intersection_df, select = -c(date))
  intersection_xts <- xts(x = intersection_df, order.by = as.Date(rownames(intersection_df)))

  # Number of years in the test period
  test_n_years <- year(end_date) - year(train_end_date)

  # Prepare data frame which will store stock weights
  v <- as.vector(c(rep(x = 0, times = ncol(intersection_xts))))

  # Set names
  names(v) <- colnames(intersection_xts)

  # Create data frame from column-vector
  HRP_weights_df <- data.frame(t(v))
  CLA_weights_df <- data.frame(t(v))

  # Create vectors to store standard deviations of portfolios
  HRP_sd_v <- c()
  CLA_sd_v <- c()
  one_n_sd_v <- c()

  # Create vectors to store quarter returns of portfolios
  HRP_portfolio_return_q_v <- c()
  CLA_portfolio_return_q_v <- c()
  one_n_portfolio_return_q_v <- c()

  dates <- c()
  #number of quarters
   for(i in 0:((test_n_years*4)-1)){

    train_period_data_xts <- window(x = intersection_xts,
                                    start = start_date + months(3*i),
                                    end = train_end_date + months(3*i))

    dates <- c(dates, index(train_period_data_xts)[nrow(train_period_data_xts)])
    cov <- cov(train_period_data_xts)
    corr <- cov2cor(cov)
```

```r

  # Step 3: Calculate Distance
  dist <- correlDist(corr = corr)

  # Step 4: Clusterization
  link <- hclust(d = dist, method = "single")

  # Step 5: Calculate weights
  # HRP
  weights <- getRecBipart(covMat = cov, sortIx = link$order) # weights - sequence
  names <- reorder(link$labels,link$order)
  named_weights <- setNames(weights, names) # column
  named_weights <- t(as.data.frame(named_weights)) # data frame with 1 row

  # 1/N:
  one_n_weights <- rep(1./ncol(named_weights), times = ncol(named_weights))
  one_n_weights <- setNames(one_n_weights, names) # column

  # weights in CLA
  r <- CLA(mu = colMeans(train_period_data_xts),
           covar = cov,
           lB = 0,
           uB = 0.2)

  # Take final optimal weights
  last_iteration <- ncol(r$weights_set)
  CLA_weights <- t(r$weights_set[,last_iteration]) # data frame with 1 row

  # the order will be different but it doesn't matter,
  # since rbind combains by column name
  HRP_weights_df <- rbind(HRP_weights_df, named_weights)
  CLA_weights_df <- rbind(CLA_weights_df, CLA_weights)

  if (i == 0){
    # initial iteration
    # drop first 0 row and change type to dataframe
    HRP_weights_df <- HRP_weights_df[-1,]
    CLA_weights_df <- CLA_weights_df[-1,]
  }
  if(i > 0){
    # Store portfolio deviation during this period
    test_period_data_xts <- window(x = intersection_xts,
                                  start = train_end_date + months(3*i),
                                  end = train_end_date + months(3*i) + months(3))

    # HRP: Calculate standard deviation of a portfolio quarterly
    HRP_sd <- StdDev(R = test_period_data_xts,
                     portfolio_method = "single",
                     weights = as.numeric(HRP_weights_df[i,]))

    HRP_sd_v <- c(HRP_sd_v, HRP_sd)
```

```
    # CLA: Calculate standard deviation of a portfolio quarterly
    CLA_sd <- StdDev(R = test_period_data_xts,
                     portfolio_method = "single",
                     weights = as.numeric(CLA_weights_df[i,]))

    CLA_sd_v <- c(CLA_sd_v, CLA_sd)

    # 1/N: Calculate standard deviation of a portfolio quarterly
    one_n_sd <- StdDev(R = test_period_data_xts,
                     portfolio_method = "single",
                     weights = one_n_weights)

    one_n_sd_v <- c(one_n_sd_v, one_n_sd)


    # Save portfolio return for a quarter
    # HRP
    HRP_portfolio_return_q <-  as.vector(Return.cumulative(as.
  matrix(test_period_data_xts) %*% weights))
    HRP_portfolio_return_q_v <- c(HRP_portfolio_return_q_v, HRP_
  portfolio_return_q)
    # CLA
    CLA_portfolio_return_q <- as.vector(Return.cumulative(as.
  matrix(test_period_data_xts) %*% t(CLA_weights)))
    CLA_portfolio_return_q_v <- c(CLA_portfolio_return_q_v, CLA_
  portfolio_return_q)
    # 1/N
    one_n_portfolio_return_q <- as.vector(Return.cumulative(as.
  matrix(test_period_data_xts) %*% one_n_weights))
    one_n_portfolio_return_q_v <- c(one_n_portfolio_return_q_v,
  one_n_portfolio_return_q)

  } #end if
} #end for

# Data frame with stores calculated weights on each iteration
HRP_weights_df <- data.frame(HRP_weights_df)
CLA_weights_df <- data.frame(CLA_weights_df)

# set dates
row.names(HRP_weights_df) <- as.Date(dates)
row.names(CLA_weights_df) <- as.Date(dates)

# Time series object with weights
HRP_weights_xts <- xts(x = HRP_weights_df,
                       order.by = as.Date(row.names(HRP_weights_df
  )))

CLA_weights_xts <- xts(x = CLA_weights_df,
                       order.by = as.Date(row.names(CLA_weights_df
  )))

# Show portfolio structure: number of non-zero weights
CLA_nonzero_num <- rowSums(CLA_weights_df != 0)
HRP_nonzero_num <- rowSums(HRP_weights_df != 0)
```

```r
# Create labels for quarters: CLA Standard Deviation
names(CLA_sd_v) <- names(HRP_sd_v) <- seq(from = 2007, to = 2018.5, by = 0.25)
names(HRP_sd_v) <- names(one_n_sd_v) <- names(CLA_sd_v)
# Calculate cumulative return
test_period_xts <- window(x = intersection_xts, start = train_end_date , end = end_date)

# HRP
HRP_portfolio_return <- Return.portfolio(test_period_xts,
                                         weights = HRP_weights_df,
                                         verbose = TRUE,
                                         geometric = TRUE,
                                         rebalance_on = 'quarters')
# CLA
CLA_portfolio_return <- Return.portfolio(test_period_xts,
                                         weights = CLA_weights_df,
                                         verbose = TRUE,
                                         geometric = TRUE,
                                         rebalance_on = 'quarters')
# 1/N
one_n_portfolio_return <- Return.portfolio(test_period_xts,
                                           weights = one_n_weights,
                                           verbose = TRUE,
                                           geometric = TRUE,
                                           rebalance_on = 'quarters')


if(j == 1){
  # USA
  USA_CLA_nonzero_num <- CLA_nonzero_num
  USA_HRP_nonzero_num <- HRP_nonzero_num
  USA_CLA_weights <- CLA_weights_df
  USA_HRP_weights <- HRP_weights_df
  USA_CLA_portfolio_return <- CLA_portfolio_return
  USA_HRP_portfolio_return <- HRP_portfolio_return
  USA_one_n_portfolio_return <- one_n_portfolio_return
  USA_CLA_sd_v <- CLA_sd_v
  USA_HRP_sd_v <- HRP_sd_v
  USA_one_n_sd_v <- one_n_sd_v
  USA_HRP_portfolio_return_q_v <- HRP_portfolio_return_q_v
  USA_CLA_portfolio_return_q_v <- CLA_portfolio_return_q_v
  USA_one_n_portfolio_return_q_v <- one_n_portfolio_return_q_v
  USA_HRP_ret_over_sd <- USA_HRP_portfolio_return_q_v / USA_HRP_sd_v
  USA_CLA_ret_over_sd <- USA_CLA_portfolio_return_q_v / USA_CLA_sd_v
  USA_one_n_ret_over_sd <- USA_one_n_portfolio_return_q_v / one_n_sd_v
}
else if(j == 2){
  # Europe
  Europe_CLA_nonzero_num <- CLA_nonzero_num
```

```
    Europe_HRP_nonzero_num <- HRP_nonzero_num
    Europe_CLA_weights <- CLA_weights_df
    Europe_HRP_weights <- HRP_weights_df
    Europe_CLA_portfolio_return <- CLA_portfolio_return
    Europe_HRP_portfolio_return <- HRP_portfolio_return
    Europe_one_n_portfolio_return <- one_n_portfolio_return
    Europe_CLA_sd_v <- CLA_sd_v
    Europe_HRP_sd_v <- HRP_sd_v
    Europe_one_n_sd_v <- one_n_sd_v
    Europe_HRP_portfolio_return_q_v <- HRP_portfolio_return_q_v
    Europe_CLA_portfolio_return_q_v <- CLA_portfolio_return_q_v
    Europe_one_n_portfolio_return_q_v <- one_n_portfolio_return_q_v
    Europe_HRP_ret_over_sd <- Europe_HRP_portfolio_return_q_v /
    Europe_HRP_sd_v
    Europe_CLA_ret_over_sd <- Europe_CLA_portfolio_return_q_v /
    Europe_CLA_sd_v
    Europe_one_n_ret_over_sd <- Europe_one_n_portfolio_return_q_v /
     Europe_one_n_sd_v
  }#end if
}# end for
```