Unsupervised Clustering of Financial Time Series

Theodore Curtil
Ecole Polytechnique Fédérale de Lausanne

Industry supervisor : Thierry Bochud (Encelade Ltd.)
Academic supervisor : Matthieu Wyart (PCSL, EPFL)

August 13, 2021

Abstract

Clustering of financial assets has been extensively used by investors to build diversified portfolios. From the naive sectorial classification of companies, to more sophisticated clustering procedures based on statistical analysis, there is not one method that rules them all. Instead, the desirability of one method or the other strongly depends on the underlying application. In this work, the focus is set on developing a robust, stable, fully unsupervised rolling framework for the clustering of assets over time; based on stock returns data. The algorithm should be unsupervised, in the sense that the number of clusters to identify is not an input of the algorithm, but is rather learnt during the clustering to adapt to changing market structure; illustrating the non-stationarity of financial markets. The unsupervised clustering algorithm we implemented is borrowed from the field of network theory, and builds upon the redefinition of the celebrated Louvain method for time series data. Building on the clustering algorithm for time series proposed by MacMahon and Garlaschelli [2015], and the mathematical toolbox borrowed from Random Matrix Theory which supports this algorithm; we introduce the notion of bootstrapped clustering, parent to the notions of probabilistic clustering and co-occurrence matrix, the central result of this work. The co-occurrence matrix is simply explained as the square matrix whose entries represent the frequency at which two stocks end up in the same community for different samples of the clustering. This simple, yet powerful, statistics obtained by aggregating bootstrapped clusterings can be interpreted as a weighted adjacency matrix; with coefficients indicating the strength of the link between two nodes. Adjacency matrices being the one element that we did not have initially to apply the vanilla Louvain algorithm, we manage to extract it from the redefined Louvain algorithm for correlation matrices. Finally, the co-occurrence matrix can be used as an adjacency matrix representing the strength of links between nodes (stocks) in the market at a given time, and fed back to the vanilla Louvain algorithm, closing the loop.

Contents

Executive summary

In this work we describe the research process we followed to develop a framework for robust unsupervised clustering of financial time series. Intermediary results are shown and discussed along the way; guiding the reader through our work from initial naive solutions to final ones. The proposed clustering procedure was not tailored for a specific application, but is rather general and undifferentiated. Some improvements and applications to trading strategies or risk management are proposed and evaluated in the last section of the present paper.

Results and applications of our clustering algorithm are computed from stock returns data between 1995 and 2020. The algorithm is rolled over the time dimension of the dataset to simulate results that one could have obtained in the past; and on each date where clusterings are estimated, only stocks belonging to the S&P500 on that date are considered. Hence, the simulations and results that we propose could have been obtained by an investor restricting his investment universe to the S&P500 over the time period from 1995 to 2020. Nevertheless, the implementation of the clustering algorithm is general enough to accept a stock universe of supposedly less correlated assets, by including returns of different asset classes and different markets.

The clustering algorithm that we propose is based on the Louvain clustering algorithm borrowed from network theory. This algorithm that we will refer to as the vanilla Louvain algorithm only uses an adjacency matrix to cluster nodes into communities[1] of nodes. For a dynamic system of nodes[2], a time series of adjacency matrices can be used to obtain a time series of clusterings, where each clustering is a snapshot of the system at a point in time.

Our clustering algorithm builds upon existing work by MacMahon and Garlaschelli [2015] who proposed to use correlation matrices as proxies for adjacency matrices in a redefined Louvain algorithm. We implement their clustering procedure in a more general framework and propose to inject some randomness into the clustering procedure. This allows us to introduce the notion of probabilistic clustering and of co-occurrence matrix. We find the co-occurrence matrix to be a robust and powerful transformation of a correlation matrix into an adjacency-like matrix suitable for applications from network theory, and in particular a suitable input to the vanilla Louvain algorithm.

The major result from this work is the robustification of the algorithm proposed by MacMahon and Garlaschelli [2015] through bootstrapping of clusterings. Bootstrapped clusterings are then the building blocks for probabilistic clusterings and for the creation of co-occurrence matrices. We use this executive summary to synthesize our application of the clustering algorithm, and to rationalize the choices we made.

We first address the case of correlation matrices, the statistics used to bootstrap clusterings. Correlation matrices can be estimated using different measures of correlation offering a tradeoff between computation time and robustness to outliers. Furthermore, Empirical correlation matrices estimated from finite time series are inevitably subject to estimation noise. We will describe our automated treatment of correlation matrices cleaning using results from Random Matrix Theory (RMT). Then we describe the different implementations of the unsupervised clustering algorithm that we made; how we robustified those methods; as well as the different uses of our implementations depending on the underlying applications.

An `HTML` version of this work exists, hosted on this website : theodorecurtil.github.io/index.html. The `HTML` version shows the code used to generate all figures and analysis.

[1] In this paper we use the words 'clustering' and 'community' interchangeably. The word 'community' is used in the description of Louvain algorithm as a group of nodes which are internally more densely connected with each other that with the rest of the network.

[2] Stocks in our case.

Correlation matrix cleaning

Following the reference work by MacMahon and Garlaschelli [2015], two ingredients are required to apply the community detection algorithm : one empirical correlation matrix at a specific point in time; and its de-noised version. Those two components allow to build the null-model to which the empirical correlation matrix is compared. It is then by finding subsets of the empirical correlation matrix where the correlation is higher than would be expected in the null-model that communities are detected and labeled.

The results of Random Matrix Theory applied to financial instruments' correlation matrices are described in Utsugi et al. [2004] and the idea of cleaning or de-noising empirical correlation matrices using RMT is described in Potters et al. [2005]. Those results will be derived again in Appendix 6.1 and verified with a sample from our data.

Unsupervised clustering algorithm

Another building block of this work is the clustering algorithm itself, which is implemented in a consistent manner to accept arbitrary asset returns' time series as input. The algorithm can be rolled over the dataset to output a time series of clusterings at a given frequency.

The unsupervised clustering algorithm investigated is the Louvain algorithm introduced by Blondel et al. [2008]; which comes from network theory. As a comer from network theory, it is based on an adjacency matrix (either binary or weighted) at a point in time to find communities of nodes more densely connected together than to nodes in other communities. A common statistical measure used as a proxy for adjacency matrices in applications to financial markets is the cross-correlation matrix : the symmetric matrix $\mathbf{C}$ with coefficients $C_{ij} = C_{ji} = \rho_{ij}$ where ρ_{ij} is the correlation between assets X_i and X_j.

It was shown by MacMahon and Garlaschelli [2015] that a direct application of the vanilla Louvain algorithm to correlation matrices is biased and only small-sized communities are correctly identified, while larger communities are melt together into one large community. The authors propose a redefinition of the Louvain algorithm which is consistent with the use of empirical correlation matrices, which is the main result from their paper and the foundation to our work.

Bootstrapped clustering

The first part of this work consisted in creating in `R` code all the functions needed to estimate clusterings in a similar fashion to that presented by MacMahon and Garlaschelli [2015]; except that it is in a rolling framework over a less biased dataset.

In the second part of this work we propose an alternative procedure by first introducing the notion of repeated clustering; and then the notion of bootstrapped clustering. Repeated clusterings and bootstrapped clusterings are the same objects in our framework, but they differ in the way we use them.

Repeated clustering was introduced as a way of exploring a greater volume in the space of clusterings and overcoming the potential local optimum found during the optimization step of the clustering algorithm. Indeed, we also added some randomness to the clustering algorithm to deal with the problem of local optimum. By repeating the clustering procedure we are able to find a best partition out of the bootstrapped one; where it is the best according to a metric. This 'repeat and select' procedure was found to be efficient and added stability to the clustering over time.

We later realized that repeating the clustering procedure to select only one outputted partition was a waste of resource. All obtained partitions could be used to robustify the corresponding time t clustering. We designed two frameworks to collapse all the sampled clusterings into a unique object : either a probabilistic clustering; or a co-occurrence matrix.

Probabilistic clusterings address a shortfall of the vanilla Louvain algorithm and of its modified version for correlation matrices : they output non-overlapping clusterings. This means that one node (or stock) is assigned to a unique cluster and no two clusters share a node. For each node, its membership in a cluster is binary : it belongs to 1 cluster, and it does not belong to the $M-1$ other clusters; if M is the number of clusters found. A probabilistic clustering is actually an overlapping clustering where each node belongs to a superposition over the different clusters. The membership of a node in a cluster is no longer binary; it is rather a weighting over the different clusters found.

The notion of co-occurrence matrix is our most remarkable result because it is so straightforward to compute and so convenient to manipulate. The information from all the bootstrapped clusterings is then condensed in a $N \times N$ matrix comparable to an adjacency matrix from network theory where all coefficients $C_{ij}^{cooc} \in [0,1]$. What is further remarkable is that its construction is based on no assumption but only on the joint dynamics of pairs of stocks across all bootstrapped clusterings. The co-occurrence matrix can then be inputted to the vanilla Louvain algorithm and collapsed into a non-overlapping clustering. It is in a way a mapping from a correlation matrix to an adjacency matrix. Something very elegant about this result is that it is a workaround to a workaround to Louvain algorithm for correlation matrices. We use the Louvain algorithm adapted for correlation matrices proposed by MacMahon and Garlaschelli [2015] to actually bootstrap a corresponding adjacency matrix which can then be inputted to the vanilla Louvain algorithm.

Applications to trading strategies

Based on the framework developed during this work to estimate unsupervised clusterings on a rolling basis for an arbitrary investment universe, some applications to trading strategies are implemented and benchmarked. Those applications presented in chapter 4 use our proprietary clustering procedures to either apply classical trading strategies within each community; or use results from clusterings as trading signals.

In particular, the usual Markowitz portfolio selection algorithm is benchmarked as well as a naive momentum strategy within clusters. In the context of those two applications, our clustering procedure is expected to enforce diversification. We also evaluate applications based on pairs of assets since information related to pairs' dynamics is readily available in the co-occurrence matrix.

We also propose in chapter 4 a further robustification of our clustering procedure through bootstrapping over the space of the parameters; as well as the implementation of a time resolved clustering algorithm which identifies the dynamics of clusters over time from a time series of clusterings.

Some applications to risk management can also be thought of. For instance, we could study the co-occurrence matrix and clustering results in the tails of stock returns' distributions to estimate the concentration of a portfolio into similar assets that could happen in case of correlated extreme events.

Description of the data

In this work we only use two sources of data. The data used in all applications is a time series of daily stock log-returns. Data is available for all stocks which have been at least one day in the S&P500 index between 1996 and 2021.

For each stock and on every time stamp, there is a finite value if the stock existed and were traded that day; otherwise there is a `NA`. The dataset of log-returns is presented as a large matrix with time stamps as rows, and stocks as columns. Stocks are anonymized to avoid any bias in the clustering procedure. Indeed, with anonymized data, we will not end up tailoring the algorithm to obtain clusters which are similar to an industry classification for instance.

Another step that is taken to unbias our results is to work with the components of the S&P500 over time[3]. Contrarily to many financial papers including our reference paper by MacMahon and Garlaschelli [2015]; this project aims at developing a dynamic rolling clustering procedure which can be applied to real world applications. In their work, the authors mostly apply their clustering procedure to a large dataset of 10 years worth of data and do not test it on a different time period. Also, they only select stocks that were continuously

[3]This means that when estimating a time t clustering, only companies included in the index on time t are considered.

traded during the period, hence without `NA`. Instead, we use a dataset of daily S&P500 membership to select only those stocks that belonged to the S&P500 on any individual day. By not only selecting those stocks which survived in the S&P500 over the whole lookback window, we avoid survivorship bias. Also, by applying our procedure on a rolling window, we avoid time-window bias which could have resulted from selecting only one time window; but some time-window bias remains since we apply our rolling procedure from 1996 to 2021 only.

By selecting those stocks that were in the S&P500 on the time stamp for which we compute a clustering; there might be some stocks that were not traded at the beginning of the lookback window : it can be a spinoff from an existing company. Hence, our rolling matrix of stock log-returns contains sparse `NA` which would arise in a real world application. An appropriate treatment of missing values is thus required.

In Table 1 are shown a sample of our log-returns dataset. Top and bottom 5 log-returns are shown for the first 5 stocks of the dataset. Notice that stocks are anonymized and `NAs` are present in the dataset; they indicate that the stock was either not traded that day, or that the ticker is no longer traded. The company may have gone out of business, or maybe merged with another company and changed ticker.

The S&P500 membership for those 5 same assets is shown in Table 2. The membership dataset is used on each date for which a clustering is computed to obtain the composition of the S&P500 on this day. This mimics what an investor or trader restricted to trading the S&P500 universe would do; and the information he would have on each individual time stamp.

Table 1: Head and tail filtered log-returns for the 5 first assets of the dataset. The index is the date, and the assets are anonymized to avoid bias. NA indicate that the asset was not traded that day, or that the ticker is no longer traded; maybe the company went out of business or merged.

index	08ee7cf7	bbc9302c	73978ca7	c193bf28	48386a5b
Head					
1995-01-03	0.0332256	-0.0161554	0.0240975	-0.0203397	-0.1397619
1995-01-04	-0.0065574	0.0257249	-0.0143887	0.0136057	0.1397619
1995-01-05	-0.0199342	-0.0127797	0.0096155	0.0000000	0.0000000
1995-01-06	0.0066890	0.0202890	-0.0096155	-0.0067797	-0.0219789
1995-01-09	0.0000000	-0.0180185	0.0096155	0.0067797	-0.0454624
Tail					
2020-12-18	NA	-0.0160173	NA	NA	0.0396312
2020-12-21	NA	0.0123587	NA	NA	0.0082146
2020-12-22	NA	0.0280669	NA	NA	0.0218018
2020-12-23	NA	-0.0070005	NA	NA	0.0056730
2020-12-24	NA	0.0076827	NA	NA	-0.0021071

Table 2: Head and tail of S&P500 membership for the 5 first assets of the dataset. The index is the date, and the assets are anonymized to avoid bias. A 1 indicates that the stock belonged to the S&P500 on that date, and a 0 indicates that it did not. NA indicates that the stock is not traded anymore.

index	08ee7cf7	bbc9302c	73978ca7	c193bf28	48386a5b
Head					
1995-01-03	1	1	1	0	0
1995-01-04	1	1	1	0	0
1995-01-05	1	1	1	0	0
1995-01-06	1	1	1	0	0
1995-01-09	1	1	1	0	0
Tail					
2020-12-18	NA	1	NA	NA	1
2020-12-21	NA	1	NA	NA	1
2020-12-22	NA	1	NA	NA	1
2020-12-23	NA	1	NA	NA	1
2020-12-24	NA	1	NA	NA	1

Motivation

Clustering - or grouping objects that share similar characteristics - has been extensively used by investors to build diversified portfolios. Indeed, having a partition of the investment universe into groups of stocks whose internal dynamics are similar and whose dynamics across the groups is different can help in the portfolio construction process. An investor without a prediction on the groups' future dynamics could diversify his holdings across the different groups, while an investor with a prediction on the groups' future dynamics could concentrate his holdings into the companies of the expected best performing group. Whatever the approach, the partition of the investment universe assists the investment decision process.

Now, such a partition can be established in different manners. Companies could be classified according to fundamental data like their business segment[4], the countries they operate in, their debt level or any combination of companies' fundamental data. Alternatively, they can be classified using statistical data derived from a finer granular data source : companies' stock prices. Stock prices reflect investors' expectations on the global economy, as well as on the underlying company's business; and stock prices data can be obtained at an extremely high frequency, below the second for most liquid stocks. Compared to fundamental data issued by corporations on every quarter or to industry classifications reviewed twice a year by the issuing companies; stock prices updated every second better capture structural changes and changes in investors' expectations at the company level with a much higher resolution. This motivates the creation of clustering techniques based on advanced statistical methods. A famous clustering procedure is the k-means clustering algorithm which partitions observations into k clusters, and where each observation is assigned to the cluster with the nearest mean, after the consistent definition of a statistics and a metrics over this statistics. This algorithm is supervised in the sense that the number of clusters is initially decided by the user; and there is no trivial procedure to decide on the optimal number of clusters to expect. Instead, one could turn to unsupervised clustering algorithms, where the number of clusters is not an input to the algorithm but rather is learnt during the clustering process by the algorithm. This class of clustering algorithm is better suited to identifying changing market conditions transitioning from states of large numbers of clusters, to states of low numbers of clusters when the dynamics of the whole investment universe is driven by a common unique factor.

In this work, we aim at executing such unsupervised clustering algorithm to effectively cluster assets from a given investment universe into groups of similar objects. We decide to implement a modified version of the Louvain method (Blondel et al. [2008]) borrowed from network theory which is consistent with the use of empirical correlation matrices instead of adjacency matrices. This work builds upon previous work by MacMahon and Garlaschelli [2015] who proposed such a redefinition of the Louvain method for financial time series.

[4]This is the case of the Global Industry Classification Standard (GICS) issued by MSCI and Standard & Poor's.

1 Louvain community detection algorithm

The Louvain algorithm introduced by Blondel et al. [2008] is a community detection algorithm to cluster network's nodes into communities. A community is defined as a set of nodes which are more densely connected together than to the rest of the network.

This chapter briefly recalls the properties of the Louvain clustering algorithm and its consistent redefinition proposed by MacMahon and Garlaschelli [2015]. Indeed, the redefinition of the null-model for the correlation matrix in terms of the so-called "random" and "market" components allows for the replacement of the usual adjacency matrix in the clustering algorithm.

1.1 Why use Louvain algorithm

The Louvain community detection algorithm clusters nodes in a network based on the binary - or weighted - adjacency matrix describing the network. The method belongs to the family of greedy algorithm which only take steps that increase their objective function. Those algorithms do not guarantee to find the optimal solution, but find at least locally optimal solutions in a reasonable amount of time; depending on the ordering of evaluation of potential solutions.

Note that this procedure creates non-overlapping communities of nodes; meaning that one node can only be put into one unique community and cannot belong to different communities at the same time, with different weightings. We developed a method which circumvents this limitation of the initial algorithm; and allows individual stocks to weight over different communities; by introducing the notion of probabilistic clustering.

We chose to use Louvain algorithm for different reasons.

- It is a member of the most popular family of community detection algorithms for graphs : modularity optimization algorithms. A review can be found in Fortunato [2010].
- Its implementation is straightforward with time complexity almost linear in the size of the network : $\mathcal{O}(N \log N)$, where N stands for the number of nodes in the network. Its fast speed is due to the simple formula to evaluate the gains in modularity associated with moving one node from its community to any other communities.
- The Louvain method provides an unsupervised clustering procedure, meaning that the optimal number of communities to be created is not an input to the algorithm, but is rather learnt and optimized during the algorithm's evaluation. This property contrasts with the popular k-means clustering algorithm; where the number of communities k must be chosen a priori.
- The algorithm clusters nodes from an adjacency matrix; and we happen to have a comparable measure of similarity between stock returns : correlation matrices; although correlation matrices cannot be fed directly to the Louvain algorithm, as we shall see later.

1.2 Louvain algorithm

In this section we briefly recall the method known as the Louvain algorithm to detect communities in large networks; as well as its key variables.

Given a network of N nodes described by a binary ($A \in \{0,1\}^{N\times N}$) or weighted ($A \in [0,1]^{N\times N}$) adjacency matrix, Louvain algorithm will create a partition of the nodes into n non-overlapping communities, where n is not known a priori. The partition of the nodes can be represented by a N-dimensional vector $\vec{\sigma}$ where the i-th component σ_i identifies the community to which node i belongs. The modularity of the partitioned network is then expressed by equation (1).

$$Q(\vec{\sigma}) = \frac{1}{A_{\text{tot}}} \sum_{i,j} \left[A_{ij} - \langle A_{ij} \rangle\right] \delta\left(\sigma_i, \sigma_j\right) \tag{1}$$

The modularity $Q(\vec{\sigma})$ depends on a null-model $\langle A_{ij} \rangle$ for the adjacency matrix, to which the empirical adjacency matrix is compared. The null-model is given by equation (2)

$$\langle A_{ij} \rangle = \frac{k_i k_j}{2m} \tag{2}$$

where k_i is the degree of node i. This definition is for undirected networks, hence, no distinction is made between in- and out-degrees. $A_{\text{tot}} \equiv \sum_{i,j} A_{ij}$ is simply a normalization factor to ensure that the modularity $Q(\vec{\sigma}) \in [-1, 1]$. In the case of binary adjacency matrices, $A_{\text{tot}} \equiv \sum_{i,j} A_{ij} = 2m$ is twice the number of links in the network.

As a matter of fact, the null-model $\langle A_{ij} \rangle = \frac{k_i k_j}{2m}$ simply represents the probability of having a link between nodes i and j under the null-hypothesis that the structure of the network is explained by the degrees of the different nodes, and not by a particular partition between communities of densely connected nodes. Indeed, if one assumes that the probability that there is a link between nodes i and j is independent from the membership or not of the two nodes to a same community; then $\frac{k_i k_j}{2m}$ is the probability that a link sampled from the m links in the network is attached to i on one side and to j on the other side. The factor $\frac{1}{2}$ accounts for the fact that all links are considered twice[5].

The modularity function in equation (1) is used as an objective function to maximize. For a partition $\vec{\sigma}$ of the network; a large value of the modularity means that within communities - the $\delta(\sigma_i, \sigma_j)$ term -, the connectivity of the nodes is greater than what would be expected under the null-model of no communities - the term $A_{ij} - \langle A_{ij} \rangle$ is large. Most community detection algorithms optimize this objective function.

The Louvain algorithm is a repetition of two distinct phases until no improvement to the modularity can be found. Initially, all N nodes are assigned a different community[6]. Then, for each node i, the gain in modularity from moving i out of its community and to the community of each of its neighbors j is evaluated. The node i is then moved to the community that maximizes the gain in modularity if there exists at least one such move with a positive gain in modularity. Otherwise, i remains in its community[7]. This process is applied sequentially and repeated over all nodes i of the network until no further improvement is possible. This concludes the first phase of the algorithm.

The efficiency of the Louvain algorithm comes from the formula for evaluating the gain in modularity ΔQ obtained when moving node i into the community of its neighbor j. Indeed, the change in modularity is given by equation (3)

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,\text{ in}}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{3}$$

where the notation is the same as that used in Blondel et al. [2008]. The point being the whole modularity does not need to be computed for each potential move. Only the nodes belonging to the community of i or to the community of the considered neighbor j will contribute to the change in modularity.

We mentioned that the algorithm finds only locally optimal solutions. This is because the resulting partition is dependent on the ordering of the nodes i. Running the procedure with two different orderings of the nodes will lead to differing partitions. This observation motivates the introduction of random fluctuations to overcome the problem of local solutions; which is not proposed in the original paper.

The second phase of the algorithm is called the network renormalization. It consists in building a new network whose nodes are the communities found during the first phase; and the weight of the link between two nodes of the new network - i.e. two communities of the old network - is now the sum of the weights of the links between nodes in the corresponding two communities. The nodes are literally merged into large nodes and the sum of the weights in the network $\frac{1}{2} \sum_{ij} A_{ij} = m$ is conserved. Once the second phase is

[5]The link from node i to j is the same as that from j to i.

[6]Initially, there are N communities.

[7]Hence, the modularity during the optimization process only takes positive or null steps.

done, the first phase is applied again to the renormalized network, and so on. The procedure stops when no gain in modularity can be achieved by further merging communities.

1.3 Redefining the null model for correlation matrices

It was shown by MacMahon and Garlaschelli [2015] that equation (1) for modularity $Q(\vec{\sigma})$ is inconsistent when a correlation matrix C_{ij} is used instead of an adjacency matrix A_{ij}. In particular, the null model derived from expectations on adjacency matrices is not consistent with the use of correlation matrices. The authors stress that for infinitely long time series the correct null model is $\langle C_{ij} \rangle = \delta_{ij}$. This means that the expected correlation matrix should be the identity matrix I_N, and the empirical correlation matrix C_{ij} should be compared to it.

The naive[8] null-model is given in equation (4)

$$\langle C_{ij} \rangle_{\text{naive}} = \phi \frac{n_{\sigma_i^*} n_{\sigma_j^*}}{\sum_{A=1}^{c} n_A^2} \tag{4}$$

where $n_{\sigma_i^*}$ is the number of time series in the community of time series i; and $\sum_{A=1}^{c} n_A^2$ is a normalization term constant over all pairs of time series (i, j). This shows the inconsistency of this naive null-model when dealing with time series. Equation (4) can never lead to off-diagonal zeros in the expected null-model.

The naive null-model in equation (4) is derived assuming an ideal setup where N time series are divided into c true communities described by a partition vector $\vec{\sigma}^*$. In this setup, each community A_i is made up of n_{A_i} standardized time series, satisfying $\sum_{i=1}^{c} n_{A_i} = N$. All time series belonging to a same community are perfectly correlated with each other and completely uncorrelated to the time series in other communities. Hence, the idealized setup is such that

$$C_{ij} = \text{Corr}[X_i, X_j] = \text{Cov}[X_i, X_j] = \delta(\sigma_i^*, \sigma_j^*)$$

An even worst property is the bias introduced by the naive null-model in equation (4) with respect to communities' sizes. For heterogeneously sized communities, pairs of stocks belonging to large communities lead to large values of $\langle C_{ij} \rangle_{\text{naive}}$ and low values of $C_{ij} - \langle C_{ij} \rangle_{\text{naive}}$; while it is the opposite for stocks belonging to small sized communities. Recall that the algorithm is based on the maximization of the modularity (1); hence favoring larger values of $C_{ij} - \langle C_{ij} \rangle_{\text{naive}}$. This means the algorithm with the naive null will more accurately identify small sized communities than larger communities, paradoxically.

To circumvent this, the authors suggest the use of another null-model when dealing with time series and empirical correlation matrices. They give a new definition of the modularity $Q(\vec{\sigma})$ to optimize, that relies on results from Random Matrix Theory (RMT). They propose to decompose the correlation matrix into a random mode, a group mode, and a common market mode $\mathbf{C} = \mathbf{C}^{(r)} + \mathbf{C}^{(g)} + \mathbf{C}^{(m)}$. The random mode would be attributed to measurement noise, and the common market mode to a dynamics common to all stocks in the investment universe. If the stock universe is the S&P500 as it is in our case, a strong market mode is expected since all stocks belong to the same index. The group mode is the one which captures the groups' dynamics; where each group or community is considered relative to others.

Finally, the authors suggest defining the null-model $\langle C_{ij} \rangle$ as the sum of a random component and a common market component $C_{ij}^{(r)} + C_{ij}^{(m)}$. This means that under the assumption of no communities - the null-model - the expected empirical correlation matrix is the sum of random noise and a common market dynamics which biases assets' correlations upwards because they are in the same index or trade in the same market. Hence, any deviation from the null-model can be seen as a manifestation of groups' dynamics.

The redefined modularity for correlation matrices now reads

[8] In the context of Louvain algorithm with correlation matrices, we use the word "naive" to refer to the use of the vanilla null-model from network theory : $\langle A_{ij} \rangle = \frac{k_i k_j}{2m}$.

$$Q(\vec{\sigma}) = \frac{1}{C_{\text{norm}}} \sum_{i,j} \left[C_{ij} - C_{ij}^{(r)} - C_{ij}^{(m)} \right] \delta\left(\sigma_i, \sigma_j\right) \tag{5}$$

In Appendix 6.2 we provide the results by MacMahon and Garlaschelli [2015] which show that the Louvain procedure can be consistently adapted to correlation matrices and to the redefined null-model.

2 Correlation matrix cleaning

In this chapter we discuss and compare the different ways to estimate a correlation matrix. Four different methods to estimate correlation matrices are investigated in the present work : Pearson, Spearman, Kendall and Gaussian rank correlation estimates. There is not one method that rules them all. Instead, all methods have pros and cons. A trade-off is to be made between computation time and robustness to outliers. Arguments on robustness are extracted from paper by Boudt et al. [2010].

We then discuss the basics of Random Matrix Theory and introduce the Marcenko-Pastur distribution as well as its applications for correlation matrix cleaning. The Marcenko-Pastur distribution describes the asymptotic distribution of eigenvalues in Wishart matrices. A thorough review on Random Matrix Theory was published by Bun et al. [2017].

2.1 Correlation measures

Correlation matrices estimated from time series of asset returns can be determined using different measures. The most classical estimator is the Pearson correlation coefficient; which is a measure of linear correlation between to series of data. This estimator is based on the moments of the distributions of the two underlying series of data; which makes it highly sensitive to the presence of outliers in the datasets.

To circumvent this limitation of Pearson correlation, estimators based on ranks can be used. The two most famous rank correlation estimators are Spearman and Kendall coefficients. They are robust to outliers but their computational efficiency is low, especially in the context of correlation matrix estimators where their lower computational efficiency is leveraged by a factor $\mathcal{O}(N^2)$ where N is the number of time series considered. Those three correlation measures are formally introduced and discussed in Appendix 6.3.

As a tradeoff between computational efficiency and robustness to outliers, Boudt et al. [2010] propose the Gaussian rank correlation. This estimator is especially interesting for our work since the correlation matrices based on the Gaussian rank correlation are guaranteed to be positive semidefinite[9] and efficient to compute in high dimensions.

2.1.1 Gaussian rank correlation

The Gaussian Rank correlation estimator was presented by Boudt et al. [2010] for its robustness. It is equal to the Pearson correlation coefficient computed from the normal scores of the ranks of the observations. It is a robust correlation estimator with breakdown point located at 12%. This means that it takes sending 12% of the data to arbitrarily large values to obtain an arbitrarily large value of the estimator. At the time of our work, no `R` function were available to compute Gaussian rank correlation based correlation matrices with appropriate treatment of missing values. The function `BSL::gaussianRankCorr` was used and adapted to deal with missing values working block by block[10]. The price to pay when building the correlation matrix block by block with time series of unequal length is the loss of the positive semidefiniteness of the correlation matrix. Indeed, as we will see later, some negative eigenvalues emerge.

For a bivariate sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, the Gaussian rank correlation is constructed by first computing the rank of the observations $R(x_i)$, scaled between 0 and 1 with a factor $1/(n+1)$. The series of scaled ranks are transformed into their corresponding series of normal scores by applying the quantile function Φ^{-1} of the standard normal distribution.

Finally, the Gaussian rank correlation is defined by the usual product-moment correlation coefficient :

$$\hat{\rho}_G = c_n \sum_{i=1}^{n} \Phi^{-1}\left(\frac{R(x_i)}{n+1}\right) \Phi^{-1}\left(\frac{R(y_i)}{n+1}\right)$$

[9]Hence they are invertible; a requirement of the Markowitz portfolio selection formula.
[10]By block of data without missing values.

with $1/c_n = \sum_{i=1}^{n} \Phi^{-1}\left(\frac{i}{n+1}\right)^2$ and where $1/\left((n-1)c_n\right)$ is the variance of the scores.

While the Gaussian rank correlation estimator is asymptotically as efficient as the sample correlation coefficient, it is much more robust. Contrarily to correlation matrices computed from Kendall or Spearman which are not guaranteed to be positive semidefinite, Gaussian rank correlation based correlation matrices are. Those combined facts motivate the use of this newly introduced correlation coefficient estimator for correlation matrix construction.

2.1.1.1 Breakdown point In the case of correlation estimates, the breakdown point is the contamination needed to invert the sign of the correlation estimate :

$$\varepsilon_n\left(\hat{\rho}; Z_n\right) = \min_k \left\{ \frac{k}{n} : \inf_{Z_n^k} \hat{\rho}\left(Z_n^k\right) \hat{\rho}\left(Z_n\right) \leq 0 \right\}$$

where Z_n^k is the contaminated dataset in which k observations of Z_n are replaced by arbitrary values.

The asymptotic value of $\varepsilon_n\left(\hat{\rho}_G\right)$ is the solution of

$$\varepsilon/2 - \Phi^{-1}(\varepsilon/2)\phi\left(\Phi^{-1}(\varepsilon/2)\right) = \frac{1}{4}$$

with ϕ the density of a standard normal. Solving this equation numerically yields $\varepsilon_n\left(\hat{\rho}_G\right) = 0.124$, the asymptotic breakdown of the Gaussian rank correlation estimate. Pearson has $\varepsilon(\hat{\rho}_P) = 0$, Spearman has $\varepsilon(\hat{\rho}_S) = 0.206$ and Kendall has $\varepsilon(\hat{\rho}_K) = 0.293$. This implies that Spearman and Kendall estimators are more robust than the Gaussian rank to contamination by outliers. Still, estimators constructed from ranks of observations are much more robust than Pearson.

In the present work, the treatment of outliers in the dataset which could lead to spurious correlations was handled at the source. Indeed, our dataset of log-returns consists of filtered log-returns which are filtered for extreme one-off events using a BIP-GARCH model (Boudt et al. [2013]).

2.2 Comparison between the different correlation estimators

Now that we introduced the four correlation measures that we would like to investigate, we benchmark them on synthetic data and on real data from our dataset. The two differentiating factors of interest to us being computation time and robustness to outliers, we compare the different measures on this basis.

2.2.1 Computation time

Let us begin with the computation time of each method. Since our clustering algorithm is expected to be applied on a rolling window, correlation matrices will be estimated at each date when a clustering has to be carried out. At a daily frequency on our dataset; this amounts to about 6500 estimates of correlation matrices. The fact that we are computing rolling correlation matrices will amplify the computation time difference between two methods. If one method is faster than the other by one second on a regular correlation matrix (with sizes $N \approx 500$ and $T \sim 10^2$ or 10^3 for our dataset), then 2 hours of computation time will be saved on the full-sample. Computation times are estimated with the R package `microbenchmark`.

All four methods were benchmarked with increasing values of N. The results are displayed as boxplots in Figure 1; and summary statistics on computation time are shown in Table 3.

From Table 3 one clearly sees that Kendall and Spearman rank correlation matrices take significantly more time to compute than Pearson and Gaussian rank. Indeed, both take more than one second to compute with returns matrices of size 252×500. Moreover, for most applications one would use a larger value of T to achieve $Q > 1$ which means $T > 500$ when dealing with S&P500 data, resulting in even longer computation time.

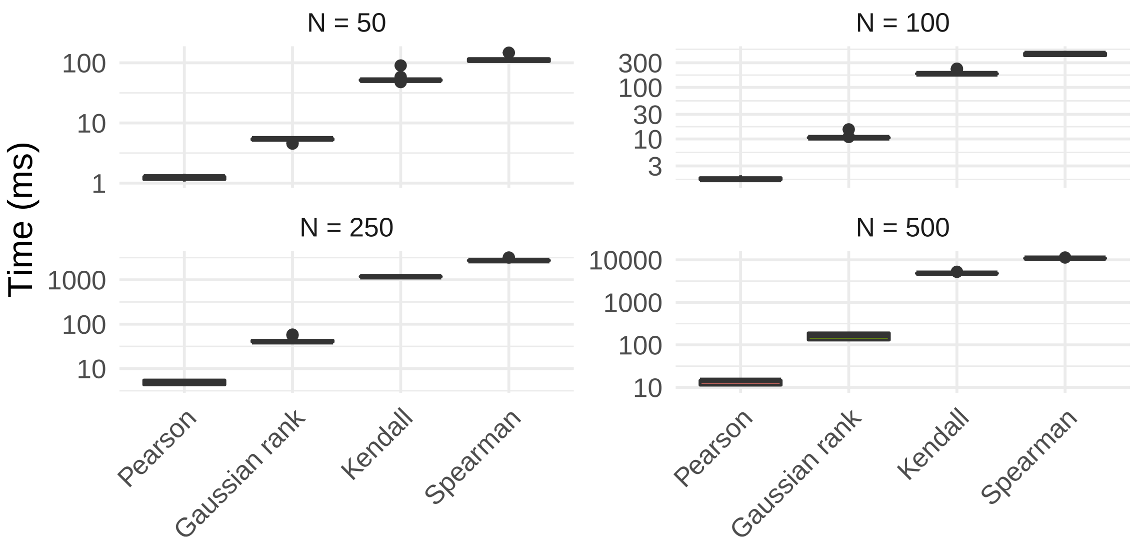


Figure 1: Boxplots of computation time for correlation matrices using different correlation estimators. All methods are benchmarked with increasing values of N to show the non-linear effect of N on the computation time. 10 realisations were executed for each setting.

Table 3: Statistics of computation time for the 4 different correlation measures; and for varying values of N, the number of assets considered. The number of time stamps used T is set to 252. lq is the lower quartile and uq is the upper quantile. neval is the number of times each couple method and size has been evaluated to account for random fluctuations in computaion time. Time is shown in milliseconds.

Method	min	lq	mean	median	uq	max	neval
N = 50							
Pearson	1.061	1.148	1.233	1.262	1.282	1.428	10
Gaussian rank	4.537	5.227	5.261	5.433	5.469	5.598	10
Kendall	47.827	50.705	55.505	51.412	52.297	90.197	10
Spearman	104.547	105.250	113.903	110.881	116.690	147.060	10
N = 100							
Pearson	1.494	1.623	1.700	1.646	1.775	1.977	10
Gaussian rank	10.340	10.502	11.054	10.572	10.660	15.341	10
Kendall	181.503	183.181	192.003	183.674	184.466	230.160	10
Spearman	410.692	412.502	440.682	451.874	459.883	469.518	10
N = 250							
Pearson	3.977	4.303	4.889	4.953	5.514	5.709	10
Gaussian rank	39.338	39.834	43.992	40.419	43.206	58.230	10
Kendall	1151.253	1170.391	1178.988	1179.460	1188.967	1198.647	10
Spearman	2663.530	2700.117	2759.550	2713.914	2745.402	3179.711	10
N = 500							
Pearson	10.589	11.316	13.263	14.536	14.650	14.721	10
Gaussian rank	127.490	129.923	163.339	171.144	191.352	203.009	10
Kendall	4787.970	4801.631	4850.563	4803.939	4821.983	5230.896	10
Spearman	10781.361	10828.382	10893.474	10842.869	10874.884	11338.989	10

Even though Gaussian rank correlation estimator takes significantly more time than Pearson's to calculate, computation time is not the single criterion to pick a correlation method for our applications. Another criterion is robustness to outliers.

2.2.2 Robustness to outliers

We designed a few statistics to compare the two methods with respect to the presence of outliers in the dataset. We expect the gaussian rank estimate to be more robust than Pearson's. We compare the two methods with our datasets of raw and filtered returns. Filtered returns are engineered from raw returns to filter out local one-off events which may affect the correlation dynamics between stocks when it should not (Boudt et al. [2013]). The designed procedure goes as follows. Each month we compute a correlation matrix based on a rolling one year window of returns from both datasets : $\mathbf{C}_t^{\text{raw,method}}$ and $\mathbf{C}_t^{\text{filtered,method}}$ for each measure of correlation.

The robustness of the two correlation measures is evaluated using three statistics. d_1 is the sum of absolute element-wise differences between the two matrices, normalized by the number of elements in the correlation matrix $\frac{N(N-1)}{2}$. It is defined for each method by:

$$d_1^{\text{method,t}} = \frac{2}{N(N-1))} \Sigma_{i<j} |C_{t,ij}^{\text{raw,method}} - C_{t,ij}^{\text{filtered,method}}|$$

d_2 is the square root of the sum of the squared element-wise differences normalized by the number of elements in the correlation matrix as well. Normalizing by the number of elements removes the size effect of the correlation matrices on the computed statistics. d_2 is defined by:

$$d_2^{\text{method,t}} = \sqrt{\frac{2}{N(N-1))} \Sigma_{i<j} \left(C_{t,ij}^{\text{raw,method}} - C_{t,ij}^{\text{filtered,method}} \right)^2}$$

Finally, $d_{\max}$ is the maximum absolute element in the matrix $\mathbf{C}_t^{\text{raw,method}} - \mathbf{C}_t^{\text{filtered,method}}$. It is trivially defined by:

$$d_{\max}^{\text{method,t}} = \max_{(i,j);i<j} (|C_{t,ij}^{\text{raw,method}} - C_{t,ij}^{\text{filtered,method}}|)$$

d_1 and d_2 are normalized to remove the impact of the correlation matrix dimension. The statistics $d_{\max}$ is a point value which only considers the pair of assets for which correlation is most different when using raw or filtered returns. The rolling measures are shown in Figure 2. Based on our analysis, the Gaussian rank correlation estimator is much more robust to outliers than Pearson's; with less bias and variance.

2.3 The Marcenko-Pastur distribution

The Marcenko-Pastur distribution has been used to study the spectral decomposition of empirical correlation matrices in the financial markets for more than 20 years; and was first suggested by Laloux et al. [2000]. Indeed, the estimation of correlation matrices is subject to measurement noise. For a set of N different assets with returns series of length T, there are $\frac{N(N-1)}{2}$ coefficients in the correlation matrix to be estimated from $N \times T$ observations. If T is not large enough compared to N, then one should expect that the estimated correlation matrix is to a large extent a random matrix.

The Marcenko-Pastur distribution is the limiting eigenvalue distribution of a Wishart matrix when $N \to \infty$ at a fixed value of $Q = \frac{T}{N}$. Consider the $N \times T$ matrix of standardized stock returns M where each row corresponds to the time series of a particular asset's returns. Then Pearson correlation reads $C = \frac{1}{T} M M^T$ in matrix form. Suppose elements of M are independent and identically distributed random variables drawn from a Gaussian distribution. Then MM^T is a so-called Wishart matrix; and for a fixed value of $Q = \frac{T}{N} \geq 1$, the density of eigenvalues of $C = \frac{1}{T} M M^T$ is exactly known in the limit $T \to \infty$, $N \to \infty$ and is given by the so-called Marcenko-Pastur distribution in equation (6)

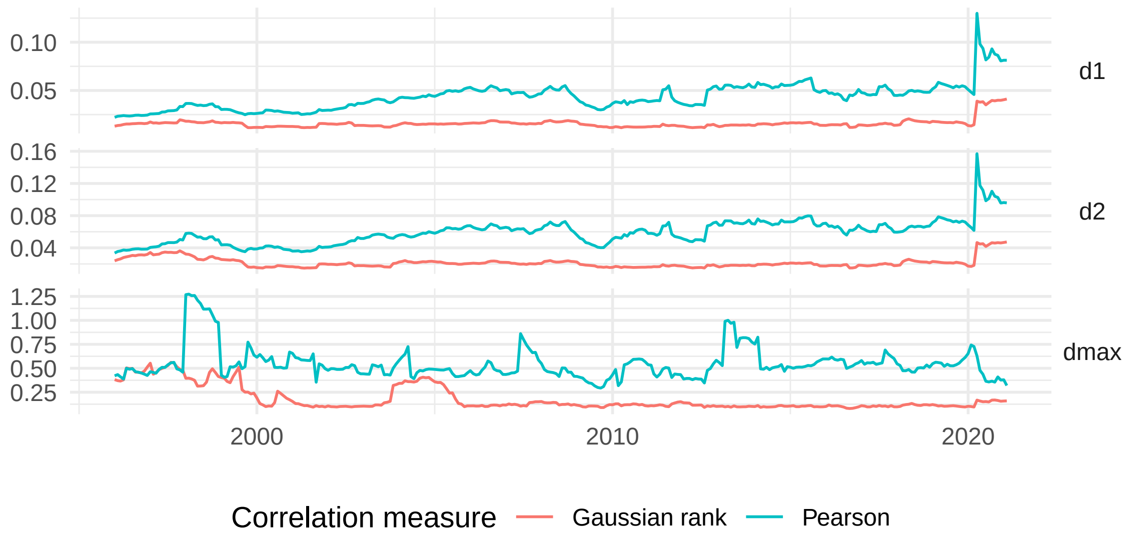


Figure 2: Robustness of the two selected correlation methods to outliers. The statistics are computed on a rolling basis with one year worth of data every month on stocks belonging to the S&P500 on this date. For each of the statistics, a lower value corresponds to a more robust method where the use of filtered returns instead of raw returns has little impact on the correlation matrix.

$$
\begin{aligned}
\rho_C(\lambda) &= \frac{Q}{2\pi\sigma^2}\frac{\sqrt{(\lambda_{\max}-\lambda)(\lambda-\lambda_{\min})}}{\lambda} \\
\lambda_{\min}^{\max} &= \sigma^2(1+1/Q \pm 2\sqrt{1/Q})
\end{aligned}
\tag{6}
$$

The admissible eigenvalues λ are restricted to the interval $[\lambda_{\min}, \lambda_{\max}]$; and σ^2 is the variance of the elements of M. A derivation of the Marcenko-Pastur result is in Appendix 6.1 following the work of Potters et al. [2005].

Figure 3 illustrates an example of such eigenvalue density on top of a simulated empirical eigenvalue density. 1000 time series of length 10000 ($Q = 10$) are sampled. Each time series is drawn from a standard normal; and all time series are independent. The correlation matrix is then estimated from the formula $C = \frac{1}{T}MM^T$. Finally, the eigenvalue decomposition is obtained using the R function `base::eigen`. The empirical eigenvalues remain between the theoretical bounds, and the agreement between the empirical eigenvalue density and the theoretical distribution as predicted by equation (6) is noteworthy.

The idea behind using Marcenko-Pastur's result for correlation matrix cleaning is to compare the spectrum of empirical correlation matrices with the theoretical distribution predicted by equation (6). Any deviation from the random case would indicate the presence of non-trivial information in the empirical correlation matrix. To summarize, the Marcenko-Pastur distribution allows to separate signal from noise in empirical correlation matrices.

To compare with the purely random case, let us now consider some stock returns data. We select a period in our dataset and study the correlation matrix of stocks in the S&P 500 on that date[11]. Figure 4 shows the empirical density of eigenvalues of an estimate of the correlation matrix of the S&P 500 on 2005-03-01 with $N = 491$ stocks and an effective $Q = 2.03$ for a total of $T = 997$ trading days. The range of displayed eigenvalues if limited to $\lambda \in [0, 5]$ for readability. The inset shows the full eigenvalue spectrum.

[11] We recall that we work with the "rolling" S&P500. At each time stamp t; we restrict our investment universe to the components of the S&P500 at time t.

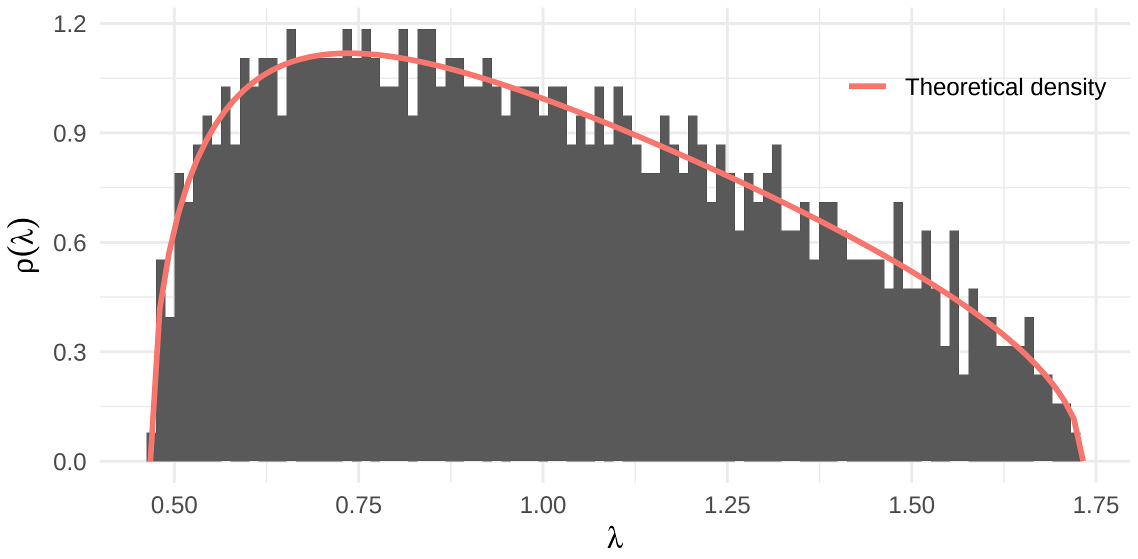


Figure 3: Eigenvalue spectrum obtained in the purely random case for N = 1000 and Q = 10 . See the agreement between Marcenko-Pastur predictions and the empirical eigenvalue density

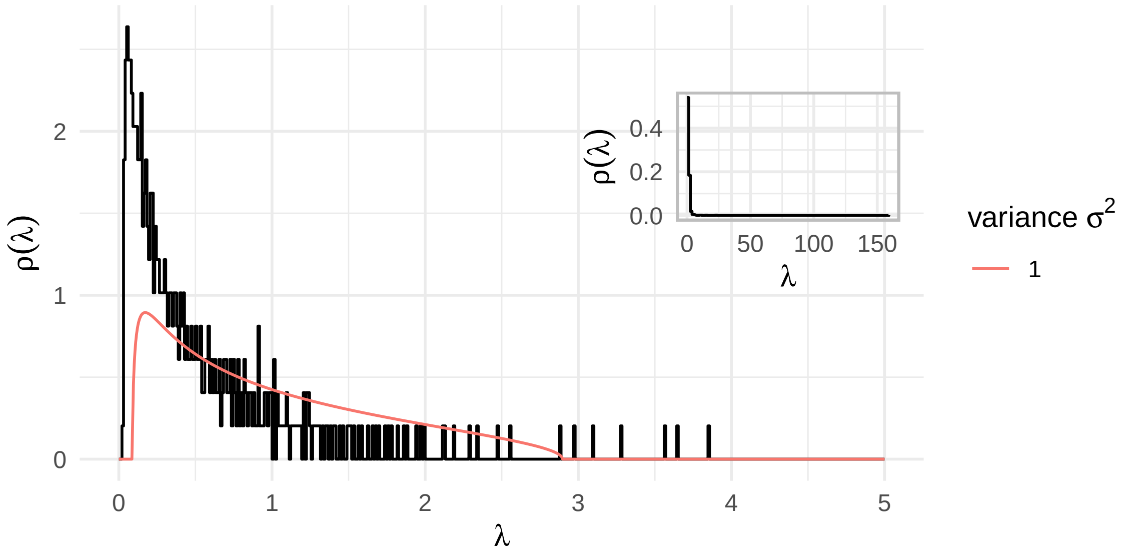


Figure 4: Empirical eigenvalue spectrum and corresponding theoretical fit. The number of assets considered is N = 491 and Q = 2.03 . Inset shows the full spectrum and shows the largest eigenvalue which is much larger than the predicted upper eigenvalue threshold $\lambda_{\max} = 2.9$.

The value of $\lambda_{\max}$ predicted by equation (6) is $\lambda_{\max} = 2.9$ ($Q = 2.03$, $\sigma^2 = 1$), while the actual largest empirical eigenvalue we observe is about 55 times larger. The eigenvector corresponding to this largest eigenvalue is interpreted as the so-called market mode with approximately equal weighting on all N components. For example, 10 components are (0.0426, 0.0477, 0.0487, 0.0384, 0.038, 0.0442, 0.0367, 0.0464, 0.0402, 0.0533). A normalized vector with equal weighting on all components would have components $\frac{1}{\sqrt{N}} = 0.0451$; showing good agreement with the elements in the eigenvector associated with the largest eigenvalue. This property already contrasts with the eigenvector associated with the second largest eigenvalue ($\lambda = 22.03$), which has the following first 10 components : (0.0764, 0.0215, -0.0472, -0.0271, -0.0353, 0.0014, -0.0534, 0.0828, 0.068, 0.104) with different signs.

The pure-noise hypothesis is inconsistent with the observed value of λ_1, the largest empirical eigenvalue. Laloux et al. [2000] suggest that components of the correlation matrix orthogonal to the market mode is pure noise. This amounts to subtracting the contribution of λ_1 from the initial value of $\sigma^2 = 1$ in equation (6), to define $\sigma^2 = 1 - \frac{\lambda_1}{N}$. Removing the largest empirical eigenvalue leads to the theoretical spectrum shown in Figure 5. The procedure can be iterated to obtain a better and better fit. By removing the p largest modes, the variance becomes $\sigma^2 = 1 - \frac{\sum_{i=1}^{p} \lambda_i}{N}$. See how successive theoretical fits approach the empirical distribution as largest modes are iteratively removed in Figure 5.

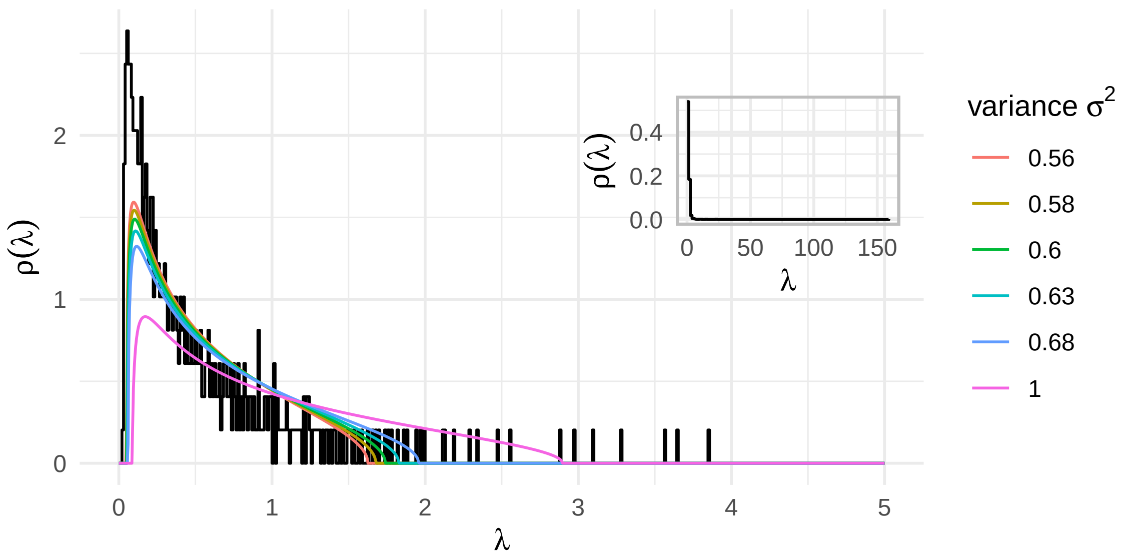


Figure 5: Empirical eigenvalue spectrum and corresponding theoretical fits for different numbers of maximum eigenvalues removed. The number of assets considered is N = 491 and Q = 2.03 . Inset shows the full spectrum and shows the largest eigenvalue which is much larger than the predicted upper eigenvalue threshold.

2.3.1 Optimal theoretical distribution

As it was previously shown, the parameter σ^2 can be adjusted by iteratively removing largest modes. But how can we decide how many of the largest eigenvalues should be removed? We could either decide a priori to remove either the largest, or the p largest eigenvalues; or we could adapt to data by dynamically determining an optimal number of eigenvalues to remove. We developed an unsupervised algorithm to remove largest modes until a no-improvement condition is reached. The algorithm is described below.

Algorithm for optimal threshold identification

1. Start from $i = 1$ and $j = i + 1 = 2$.
2. Compute $\lambda^i_{\max} = (1 - \frac{\sum_{l=1}^{i} \lambda_l}{N})(1 + 1/Q \pm 2\sqrt{1/Q})$.
3. Identify the two eigenvalues in the empirical eigenvalue distribution which are just below and just above $\lambda^i_{\max}$; such that $\lambda^{-,i}_{\text{emp}} \leq \lambda^i_{\max} \leq \lambda^{+,i}_{\text{emp}}$.
4. Do the steps 2 to 3 for j as well to identify the two successive empirical eigenvalues $\{\lambda^{-,j}_{\text{emp}}, \lambda^{+,j}_{\text{emp}}\}$ which sandwich $\lambda^j_{\max}$.
5. If the two intervals of empirical eigenvalues are the same, stop the algorithm and output the eigenvalue threshold $\lambda^i_{\max}$. Otherwise, removing one more eigenvalue allowed to significantly reduce the eigenvalue threshold : increment i and j by 1 and repeat the steps from step 2.

The idea of the algorithm is to iteratively remove the largest eigenvalues until the change in $\lambda_{\max}$ predicted by Marcenko-Pastur is less than the spacing between two empirical eigenvalues. The procedure assumes that the empirical density is sufficiently dense and the number of empirical eigenvalues is large.

The optimal fit we obtain is shown in Figure 6 for the same empirical eigenvalue distribution as in Figure 4 and 5. The small-eigenvalues part of the empirical eigenvalue distribution is well fitted by the theoretical distribution, at the price of a lowered empirical eigenvalue threshold[12]. Note the better fit on $\lambda_{\min}$ as well.

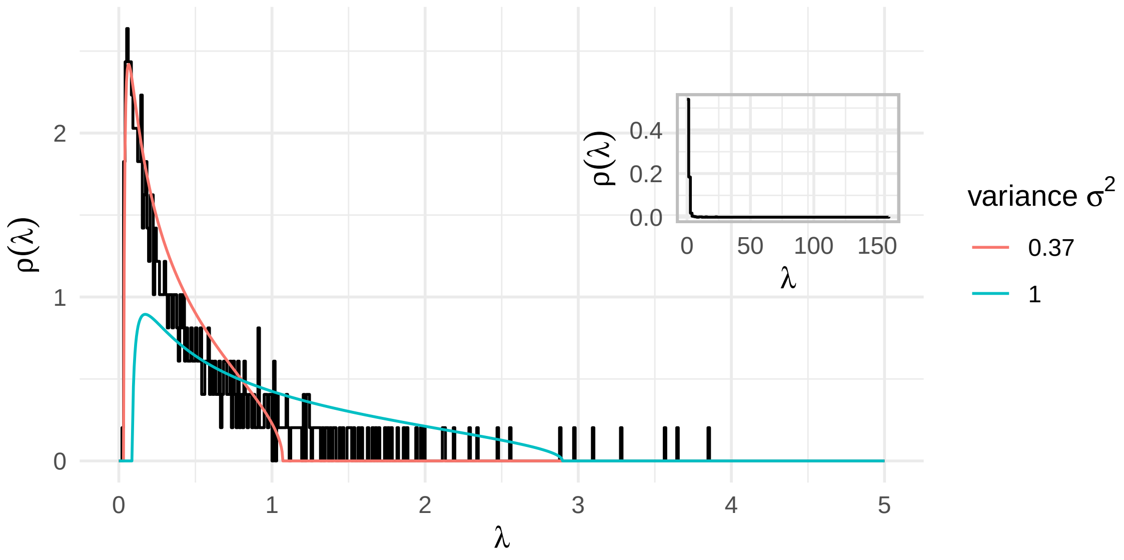


Figure 6: Empirical eigenvalue spectrum and fits with different values of σ^2. The blue line is the naive Marcenko-Pastur fit where no mode is removed. The red line is the fit obtained after removing the $p = 44$ largest eigenvalues which is determined in an unsupervised manner following our algorithm. The number of assets considered is N = 491 and Q = 2.03 . Inset shows the full spectrum and shows the largest eigenvalue which is much larger than the predicted upper eigenvalue threshold.

This procedure allows us to find an optimal theoretical $\lambda_{\max}$ to split the empirical eigenvalues into a random bulk composed of the eigenvalues less than $\lambda_{\max}$ and group modes which actually contain the useful information. This splitting of the eigenvalues now allows us to properly define the null-model for our clustering algorithm : $\mathbf{C} = \mathbf{C}^{(r)} + \mathbf{C}^{(g)} + \mathbf{C}^{(m)}$.

[12]The eigenvalue threshold $\lambda_{\max}$ is shifted to the left during the optimization process.

2.4 Application to redefine the null model

Using the tools from RMT, one can de-noise a given correlation matrix. As seen, the bulk of eigenvalues falling in the predicted interval $[\lambda_{\min}, \lambda_{\max}]$ can be attributed to random noise in the estimates of correlation coefficients; while eigenvalues falling outside of this theoretical interval represent deviations from the null hypothesis of purely random independent stock returns. Hence, only eigenvalues greater than $\lambda_{\max}$ are considered relevant and informative. The largest eigenvalue under the null-hypothesis $\lambda_{\max}$ is interpreted as a threshold which allows to split a correlation matrix $\mathbf{C}$ into a random part and a structured part

$$\mathbf{C} = \mathbf{C}^{(r)} + \mathbf{C}^{(s)} \tag{7}$$

where each component can be written using bra-ket notation

$$\begin{aligned} \mathbf{C}^{(r)} &= \sum_{i:\lambda_i \leq \lambda_+} \lambda_i \left|v_i\right\rangle \left\langle v_i\right| \\ \mathbf{C}^{(s)} &= \sum_{i:\lambda_i > \lambda_+} \lambda_i \left|v_i\right\rangle \left\langle v_i\right| \end{aligned} \tag{8}$$

The decomposition of the correlation matrix can then be pushed further by removing the mode supposedly common to all stocks in the market : the common market mode. Thus, $\mathbf{C}$ can be rewritten as $\mathbf{C} = \mathbf{C}^{(r)} + \mathbf{C}^{(g)} + \mathbf{C}^{(m)}$; where $\mathbf{C}^{(m)} = \lambda_1 \left|v_1\right\rangle \left\langle v_1\right|$ and λ_1 is the largest empirical eigenvalue. This is the null-model that is used in our modified Louvain algorithm, and the modularity can be rewritten as in equation (5) :

$$\begin{aligned} Q(\vec{\sigma}) &= \frac{1}{C_{\text{norm}}} \sum_{i,j} \left[C_{ij} - C_{ij}^{(r)} - C_{ij}^{(m)}\right] \delta\left(\sigma_i, \sigma_j\right) \\ &= \frac{1}{C_{\text{norm}}} \sum_{i,j} C_{ij}^{(g)} \delta\left(\sigma_i, \sigma_j\right) \end{aligned}$$

where $\mathbf{C}^{(g)}$ is called the filtered correlation matrix; from which random noise and common market correlation dynamics are filtered out.

$\mathbf{C}^{(g)}$ is expected to contain neither uncorrelated noise at the individual stock level, neither correlations at the market level. We refer to it as the group component which contains correlation information at the community level; which makes it an appealing candidate for community detection.

The decomposition suggested by MacMahon and Garlaschelli [2015] makes sense when considering stocks from the S&P 500 which are expected to have a strong common driver; but this hypothesis lacks substance when considering a much broader stock universe containing different asset classes and different markets. Under a more general setting, the presence of a unique market mode (or the existence of a market mode at all) becomes questionable. We will propose a redefinition of the null-model in section 3.2 which is consistent with a more general setting, and makes neither any assumption on the number of market modes, nor on the existence of such market mode.

3 Our implementation of community detection

We implemented a recursive hierarchical community detection algorithm following the work of MacMahon and Garlaschelli [2015]. Recursive means that communities as well as sub-communities (communities within communities) are resolved. This creates a hierarchical clustering and the depth of the recursive algorithm is an input parameter. A depth of 1 is a simple clustering of the N nodes. A depth of 2 means that the clustering procedure is repeated within each individual community and so on. If a target depth is not specified, our algorithm digs in the hierarchical structure until no sub-communities can be identified.

3.1 Hierarchical clustering

The hierarchical structure is revealed by iterating the clustering algorithm recursively within each community. Given an initial correlation matrix $\mathbf{C}$, the clustering procedure will output a partition of the nodes into communities. This is the first clustering level. At the second level, a community of size s is represented by the correlation matrix $\mathbf{C}_*$, a $s \times s$ sub-matrix of $\mathbf{C}$. Based on this sub-matrix, the null model $\langle \mathbf{C}_* \rangle$ is constructed again by finding the eigenvalue threshold λ_{max} which separates the random bulk of eigenvalues from the larger eigenvalues.

An issue arises at this point. If the initial correlation matrix could be considered high-dimensional with $N = 500$ stocks - hence results from RMT were applicable - it may not be the case at deeper levels as the sizes of sub correlation matrices shrink going down the hierarchy. We noticed that our community detection algorithm usually finds in the order of 4 communities per parent-community at the higher level. At the hierarchical level 3, the number of communities could reach about $4^3 = 64$ which would correspond to communities of size $\approx \frac{500}{4^3} \approx 8$ stocks assuming homogeneously sized communities. With $N = 8$ stocks only, results from RMT cannot be applied anymore. We recall that results from Marcenko-Pastur in equation (6) hold as $N \to \infty$, $T \to \infty$.

To deal with the shrinking dimensionality problem, we do not compute hierarchical clusterings with depths greater than 3 since we cannot expect reliable results; at least not with $N = 500$ as is the case with the S&P 500. Furthermore, to estimate the eigenvalue threshold λ_{max} at depths greater than 1 we cannot reliably use the methodology presented in section 2.3.1 and in Figure 6. The reason being the procedure implemented to find the optimal number of largest eigenvalues to remove from the empirical eigenvalue spectrum tends to remove a lot of eigenvalues, about 40, and relies on a dense eigenvalue spectrum, i.e. N large. This is why at levels deeper than 1 where the eigenvalue spectrum becomes sparse due to the shrinking dimensionality problem, we use an empirical random threshold determined by bootstrapping, as per the procedure described below.

3.1.1 Empirical random threshold

Eigenvalue thresholds predicted by equation (6) correspond to maximum and minimum eigenvalues expected from the correlation matrix' spectrum when the underlying processes are assumed random and independent, in the limit of large N and T. Since the large N hypothesis fails as we move down the hierarchical clustering, we decided to use bootstrapped eigenvalue thresholds from random, i.i.d. processes.

A grid (N, Q) is created from a list of values for the parameters N and Q. For each pair of parameters (N, Q) on the grid, a large number of $NQ \times N$ matrices of random normal independent returns are sampled. From each sample's correlation matrix, the largest eigenvalue is saved; and the median value of the obtained distribution of eigenvalues is saved as the bootstrapped empirical random threshold for this pair (N, Q). We then build a dictionary of empirical random upper eigenvalue thresholds to be called from on the fly when needed for a given pair (N, Q).

The corresponding upper eigenvalue thresholds are shown in Figure 7 for different values of N. The limiting distribution $N \to \infty$ obtained from Marcenko-Pastur's results is shown in black as well. Note the deviation from the theoretical thresholds (equation (6)) at low N values. In small communities, results from RMT do not hold anymore and the clustering procedure which uses Marcenko-Pastur's results cannot be blindly applied anymore. Instead, one should turn to the dictionary of bootstrapped eigenvalue thresholds to split the empirical correlation matrix into random and structured components.

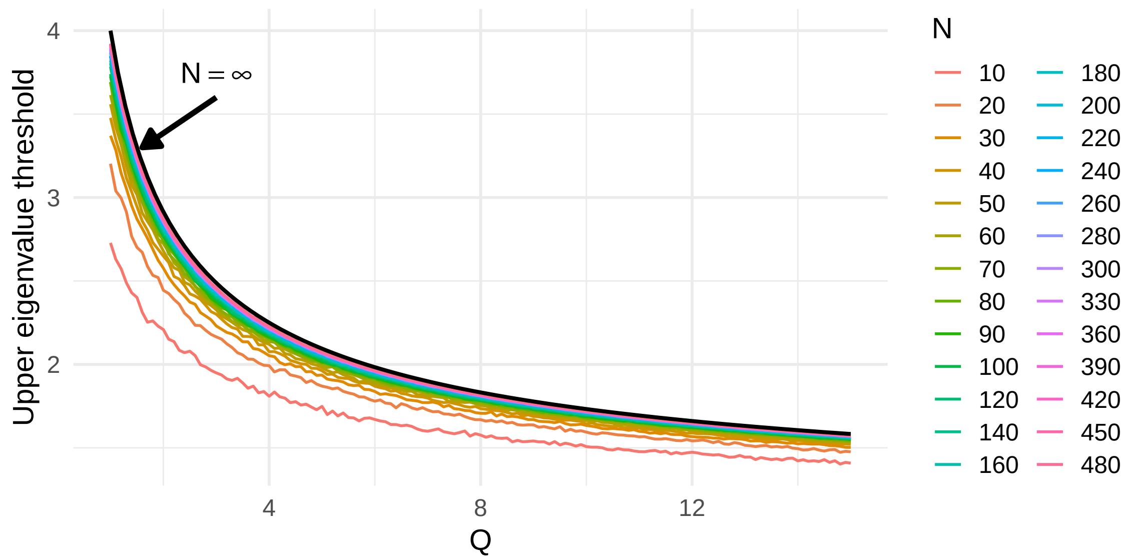


Figure 7: Empirical random upper thresholds against Q. The small size effect at low N is shown.

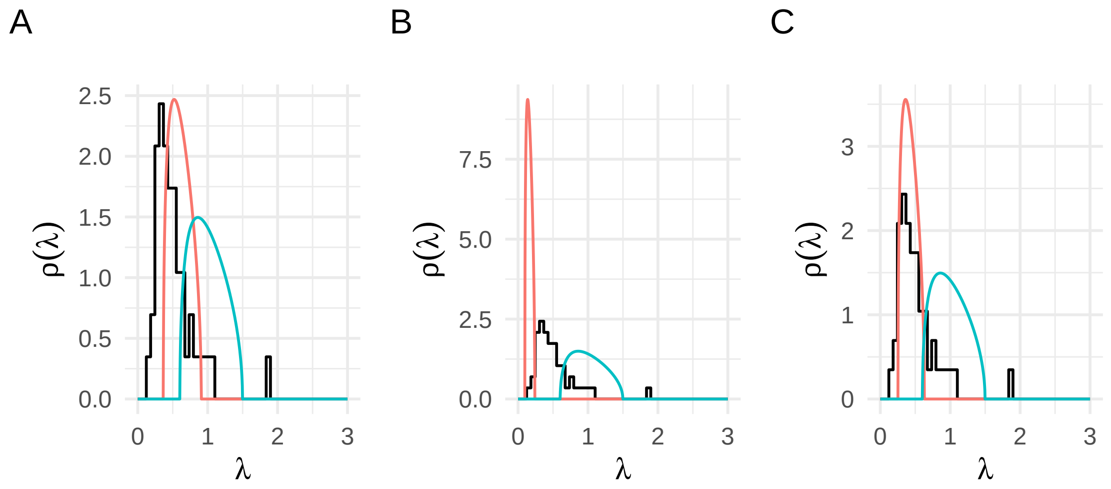
We decided to use those bootstrapped upper eigenvalue thresholds for depths greater than 1 to account for deviations from Marcenko-Pastur as N shrinks. We illustrate this choice with an example taken from our returns data. Figure 8 shows the different methods that we have at hand to fit the Marcenko-Pastur distribution. In particular, one can see how our unsupervised method to find the optimal threshold fails with small N. The use of bootstrapped random eigenvalue thresholds seems consistent at low N.

3.2 Consistent null-model

Our implementation of the clustering algorithm takes into account the fact that the null-model suggested by MacMahon and Garlaschelli [2015] might not be valid when dealing with a broad asset universe, and that the largest eigenvalue λ_1 might contain some information on the community structure (i.e. is not a common market mode).

To avoid difficulties and make sure the procedure applies in a non supervised manner, for any pass of our clustering algorithm, we start with the null-model $\langle C_{ij} \rangle = C_{ij}^{(r)}$: the most general case. With this null-model, we include the market component previously referred to as $C_{ij}^{(m)}$ in the group component $C_{ij}^{(g)}$. By doing so, we allow the largest eigenmode to contain information on the community structure that could assist in differentiating two communities. Think of a universe made of the S&P500 and of the CSI 100[13]. It is not obvious that this universe has a common market mode. Instead, it could be that the largest eigenmode contains information which could help cluster the universe into the two separate indexes. Then, if the clustering procedure finds no communities with this null-model - i.e. the algorithm puts all stocks in a single community - it means that the largest eigenvalue was effectively associated with a common mode, the market mode. The null-model would then be updated to $\langle C_{ij} \rangle = C_{ij}^{(r)} + C_{ij}^{(m)}$ where only the largest eigenvalue is included in the market component. This procedure is more general than the one suggested by MacMahon and Garlaschelli [2015] and is applicable to a broader investment universe.

[13]The capitalization-weighted market index replicating the performance of the 100 largest stocks traded in the Shanghai and Shenzhen exchanges.



A :Remove only the largest mode. B : Our unsupervised method. C : Random eigenvalue threshold.

Figure 8: All upper threshold estimation methods available, for a community of size $N = 50$. One notes that for low N our unsupervised method fails; while the two other methodologies provide consistent fits. Although, the methodology with the bootstrapped eigenvalue threshold is preferred since it does not rely on an arbitrary rule. (blue) The original Marcenko-Pastur fit. (red) The modified variance Marcenko-Pastur fit.

In the context of hierarchical clustering, we implemented the following procedure to construct the null-model $\langle \mathbf{C}_* \rangle$:

Algorithm for consistent null-model selection

1. Consider the most general null $\langle \mathbf{C}_* \rangle = \mathbf{C}^{(r)}$.
2. If only one community is found, add the largest eigenvalue λ_1 to the market component and define the null-model $\langle \mathbf{C}_* \rangle = \mathbf{C}^{(r)} + \mathbf{C}^{(m)}$. Else, accept the clustering into at least two communities.
3. Iteratively remove the largest eigenvalues as long as only one community is found. For a level r clustering, the maximum number of largest eigenvalues to include in the market mode is r.

This procedure guarantees that the algorithm will try and cluster while only comparing to the pure-noise null; but will add more and more eigenvalues in the market-mode to help the procedure find a non-trivial partition.

3.3 The naive approach

The naive approach to clustering with our procedure consists in running the algorithm only once and obtaining a hierarchical clustering. We recall that even though Louvain clustering is deterministic - all the potential moves of one node from its community are evaluated -, the output will depend on the ordering of the nodes when evaluating such moves[14]. This means that running the algorithm twice, starting from two different initial nodes ordering will result in two different clusterings. We further enforced some randomness

[14]Nodes and their potential migrations towards other communities are evaluated sequentially; and the ordering of the nodes can be a source of instabilities. Suppose node i is evaluated before j. It could be that i is associated to the community of k, and that j is not associated to the community of i and k, because i "closed" the community. On the other hand, evaluating j before i could lead to associating j to the community of k while keeping the community "open" for i to join.

in our procedure by introducing a "random shuffling" parameter. At the start of each pass of the Louvain algorithm, a portion of the nodes controlled by the shuffling parameter is allocated to random communities. This allows to avoid local optimums by pushing the clustering out of equilibrium and exploring a larger volume in the space of partitions.

Table 4 presents the number of communities and the population of each community for two passes of the clustering procedure. Only the orderings of the nodes in the queue differ. This cross-sectional instability[15] shows that accepting the clustering outputted by one pass of the algorithm does not guarantee to find the "true" or the "best possible" partition. Also, the cross-sectional instability will lead to unstable clusterings over time, where some stocks might oscillate between different communities.

Stability of clusterings over time is a desirable attribute for any application. Indeed, stability means less frequent portfolio rebalancing and less transaction costs for trading applications; as well as assurance of informative clusterings and a potential predictive power. But stability over time cannot be achieved if not even cross-sectional stability can be guaranteed. We need to design an adaptive procedure to decide on one unique clustering for a given time stamp. One could simply accept the first outputted clustering; but this seems extremely arbitrary and do not help with the time stability. One could design a rule to decide on the ordering of the nodes when running the algorithm, enforcing cross-sectional stability at the cost of deciding on a rule. Finally, one could fix no rule and leverage the power of random sampling to estimate a statistics, or bootstrap the clustering.

Table 4: Two repetitions of the clustering algorithm on the same data. Only the ordering of the nodes differ. There is not one community that is the same between the two repetitions.

	Community names			
Clustering number	0001	0002	0003	0004
1	123	164	69	135
2	128	166	64	133

3.4 A first alternative to the naive approach

To circumvent this limitation of instability within the cross-section of clustering estimates, we introduce an alternative approach which consists in iterating the procedure M times and selecting the most "central" partition as the true partition. To this end we introduce the variation of information (VI) which is a measure of distance between two clusterings. The variation of information is related to the notions of mutual information and entropy.

3.4.1 Variation of information and the L1-median

We propose to use the variation of information as a measure of distance between all M sampled clusterings. The full set of clusterings can be characterized by a $M \times M$ variation of information matrix with components $V_{ij} = V_{ji} = VI(\mathcal{C}_i, \mathcal{C}_j)$; and a most central clustering can be identified using the L_1-median introduced later. We illustrate in Figure 9 this idea. The M clusterings can be represented by a fully connected network (only links to the most central partition are drawn in Figure 9) based on the variation of information matrix, which is a distance-like matrix. Finally, the most central partition is identified as the true partition for this time stamp.

We implemented the variation of information following Meila [2003] to compare two clusterings of the same dataset. The variation of information measures the amount of information gained or lost in changing from clustering $\mathcal{C}$ to clustering $\mathcal{C}'$. The variation of information is positive, symmetric and it obeys the triangle

[15] At a fixed time, with the same data, two clusterings will be different.

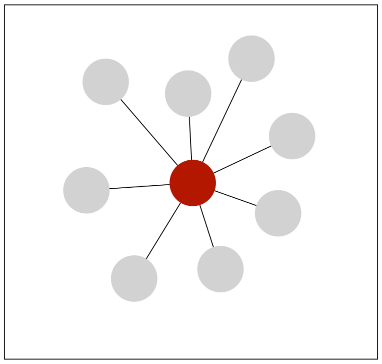


Figure 9: The most central partition out of M partitions is identified in the space of clusterings as the one which minimizes its VI distance to all other partitions.

inequality; which makes it a metric on the space of clusterings. This motivates the use of VI to choose the most central partition $\mathcal{C}$ out of M such partitions.

Given two clusterings $\mathcal{C}$ and $\mathcal{C}'$ of the same dataset, the variation of information is given by equation (9)

$$VI(\mathcal{C}, \mathcal{C}') = H(\mathcal{C}) + H(\mathcal{C}') - 2I(\mathcal{C}, \mathcal{C}') \tag{9}$$

where $H(\mathcal{C})$ is the entropy of the clustering $\mathcal{C}$ and $I(\mathcal{C}, \mathcal{C}')$ the mutual information between $\mathcal{C}$ and $\mathcal{C}'$. The entropy is given by

$$H(\mathcal{C}) = -\sum_{k=1}^{K} P(k) \log P(k) \tag{10}$$

with $P(k) = \frac{n_k}{N}$, where n_k is the number of nodes in cluster k and N the total number of nodes. Similarly, the mutual information is given by

$$I\left(\mathcal{C}, \mathcal{C}'\right) = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P\left(k, k'\right) \log \frac{P\left(k, k'\right)}{P(k) P'\left(k'\right)} \tag{11}$$

The variation of information can be computed on all pairs of clusterings and synthesized in a symmetric matrix. To illustrate how one such matrix looks like, we repeated our clustering 6 times to obtain 6 different partitions. From those 6 partitions, all coefficients of the variation of information matrix are computed using equation (9). The resulting matrix is shown as a heatmap in Figure 10. A large coefficient means that the two corresponding partitions are distant in the space of clusterings; while a low value means they are close. A value of 0 means that the two clusterings are identical, which happens in the diagonal of the VI matrix. Some partitions look more central than others just by looking at the color-coded VI matrix. We now need a criterion to decide on the most central partition.

To find the one partition which is the most central in the space of the sampled partitions, we propose to use the L_1-median reviewed by Fritz et al. [2012], which is also called the spatial median. The L_1-median has an efficient implementation in the R package `pcaPP`.

The L_1-median is a generalization of the univariate median to higher dimensions. For a dataset $X = \{\vec{x}_1, \cdots, \vec{x}_n\}$ with each $\vec{x}_i \in \mathbb{R}^p$, the L_1-median is

$$\hat{\vec{\mu}}(X) = \underset{\vec{\mu}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\| \tag{12}$$

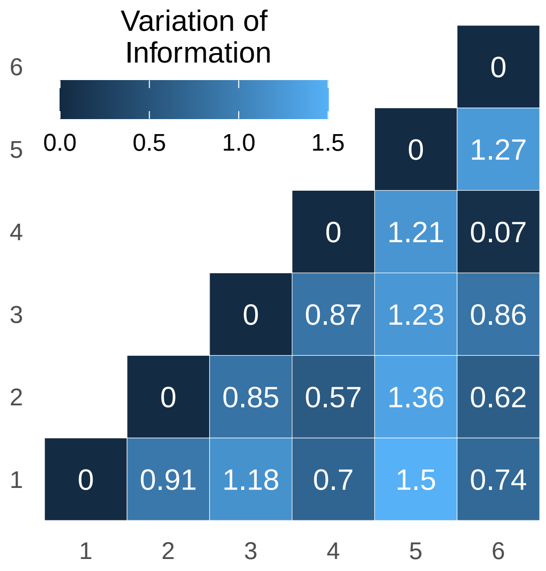


Figure 10: Heatmap of the variation of information matrix for 6 repetitions of the clustering procedure. A central partition is one that has a large number of low-valued coefficients; i.e. a lot of dark tiles. A partition with lots of light tiles is distant from a lot of other partitions and is not central.

where $\|\cdot\|$ is the Euclidean norm.

In the case of M samples $\mathcal{C}^*$ of the true clustering $\mathcal{C}$; the L_1-median of the variation of information matrix X is M-dimensional and each $\vec{x}_i \in \mathbb{R}^M$. $\vec{x}_i$ is the vector of variation of information of sampled clustering $\mathcal{C}^*_i$ with all other clusterings. The most central clustering is identified as the one clustering in $\mathcal{C}^*$ which minimizes its component of the L_1-median $\hat{\vec{\mu}}(X)$. More formally

$$\mathcal{C}^*_{\text{central}} = \mathcal{C}^*_j \quad \text{where} \quad j = \underset{i}{\operatorname{argmin}}\, \hat{\vec{\mu}}(X)_i$$

We sampled 60 partitions of the same dataset and ordered the clusterings by minimization of the L_1-median. Results on the sizes of the obtained communities are shown in Table 5 where only the 5 most and least centered partitions are shown. One can clearly see the gain in stability through repetition of the clustering procedure and selection of the most central partition based on L_1-median minimization. The “best” clusterings look alike, even though some instabilities are still noticeable. This procedure has the advantage of selecting in an unsupervised manner a “best” partition out of repetitions of the clustering algorithm.

At this point we have a consistent procedure which allows to estimate a hierarchical clustering of assets by iterating M times the clustering procedure into each of the higher level communities. The procedure is summarized in the box below.

Algorithm for iterated hierarchical clustering

1. Select the clustering depth n.
2. Repeat M times the first-level clustering procedure. Accept the most central partition.
3. Enter each community and repeat step 2 within each community.
4. Repeat steps 2-3 until depth n is reached.

Table 5: Clusterings ranked from most central to least central according to L1-median. Top 5 and bottom 5 are shown. Values represent the community sizes.

	Community names			
Clustering rank	0001	0002	0003	0004
Top 5				
1	120	165	69	137
2	120	168	69	134
3	122	166	70	133
4	120	169	69	133
5	121	168	69	133
Bottom 5				
56	123	188	40	140
57	129	68	222	72
58	126	67	188	110
59	127	67	187	110
60	126	68	182	115

3.4.2 Time evolution

The next step is to run the clustering algorithm over time, because stationarity is for sure not applicable to financial markets. Two desirable properties when clustering a system over time are

1. The matching of clusters over time. It is likely that from one time stamp to the next some clusters' compositions will look alike, with maybe few additions or deletions of stocks. We would like to be able to match time t clusters to time $t-1$ clusters.
2. Tracking the clusters' dynamics over time. We would like to be able to detect clusters' birth, death, merging, splitting or any other interactions.

To illustrate the kind of dynamics that can exist between two time separated clusterings, we ran the repeated procedure at two distinct time stamps, separated by 1 month. 11 shows the flows between the identified clusters at each time stamp; in the form of a Sankey diagram. This simple setting already illustrates the challenges associated to tracking clusters over time :

1. The number of assets and the assets themselves are not constant over time : only stocks which belong to the S&P 500 on each time stamp are included. On a 10-year period a turnover as high as 50% can be expected.
2. Most of communities' mass is conserved between two time stamps close in time. But it can happen that small groups of stocks detach from a community and join another. This will make the tracking of clusters through long periods of time difficult.
3. Some stocks may oscillate between clusters in a sporadic manner over time, while some groups of stocks consistently remain together in the same cluster.

We show how the matching between time stamps can be made over time in the chapter on applications in section 4.3, introducing the notion of dynamic communities; following the work of Liechti and Bonhoeffer [2019].

With this repeated clustering procedure and the rule of selecting the most central partition out of the M samples, we considerably increased the robustness of the initial procedure with respect to random fluctuations and local minima following the Louvain greedy optimization. Our only regret at this point is to waste lots of resources by using only one clustering result out of M clusterings sampled at each iteration of the repeated

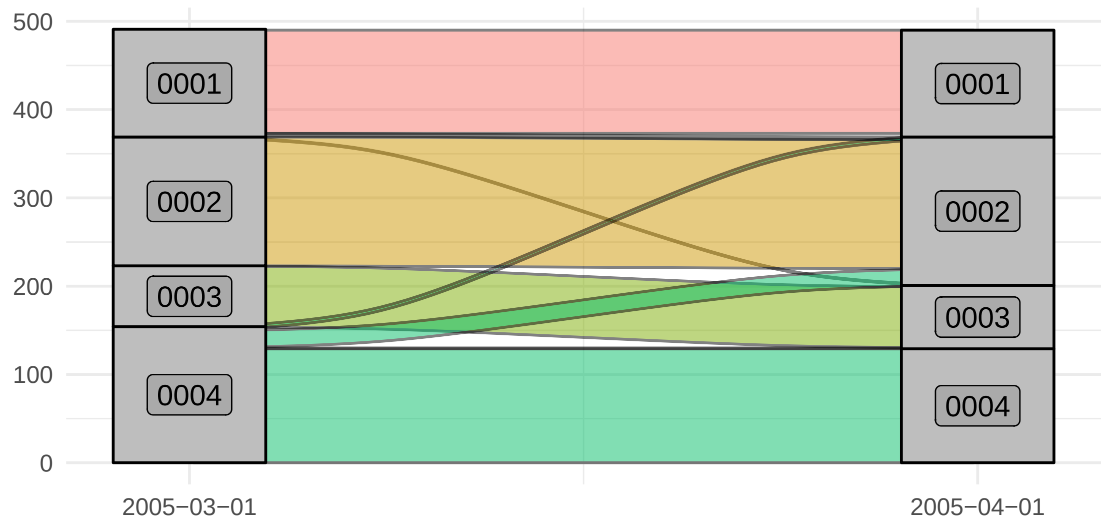


Figure 11: Sankey diagram showing the flows of assets between two time-separated cluterings. Only first hierarchical level is shown. Clusterings on dates 2005-03-01 and 2005-04-01 are obtained by running 12 times the clustering procedure and selecting the most central partition in the L_1-median sense.

procedure. This leads us to our second proposition to robustify the clustering algorithm : the notion of probabilistic clustering.

3.5 Probabilistic clustering

In this section we leverage the power of random sampling and bootstrapping of statistics. Having realized that our previously described iterative procedure wastes lots of resources by discarding $M - 1$ of the M clusterings; we propose to rely now on the notion of bootstrapped clustering. A bootstrapped clustering is a clustering estimated from a series of sampled clusterings; in a similar way that a bootstrapped statistics for a population is estimated from a series of samples' statistics (Efron [1992]).

Introducing bootstrapped clustering allows us to obtain probabilistic clusterings, where each stock is defined by a distribution over a basis of communities. Bootstrapped clustering also leads us to introduce a powerful statistics : the co-occurrence matrix; which is estimated from bootstrapped clusterings. This new statistics is found to be a transform of the initial correlation matrix into an adjacency-like matrix which can be a direct input to the vanilla Louvain algorithm. It is a simple $N \times N$ matrix which stores information from tens of bootstrapped clusterings and which can be inputted to any further application. Finally, the notion of co-occurrence matrix leads to our utmost robustification of the clustering procedure attained in this work.

3.5.1 Motivation for a probabilistic clustering

As mentioned in section 1.1, Louvain algorithm produces some non-overlapping communities, meaning that one stock is associated with only one community; and that a community is defined as a vector of binary entries over the stocks. Suppose we have 5 stocks $\{1, 2, 3, 4, 5\}$ and 3 communities $\{A, B, C\}$. A clustering could be $\{A, B, A, A, C\}$. Then, community A would be defined by the binary vector $\{1, 0, 1, 1, 0\}$ over the stocks. This property of Louvain algorithm allows for clearly defined and separated communities, but certainly hinders useful information. For instance, suppose that stock 1 which was found to be in community A is only weakly attached to community A, but just slightly more than it is to community B. Then, the Louvain procedure would most certainly put it in community A. But now the binary affiliation to A may lack substance; and a weighting over the three communities may be more informative. This information of affiliation for stock 1 may read $\{0.6, 0.4, 0\}$, which better represents the fact that stock 1 is not absolutely related to community A, but may rather find a probabilistic inclusion in all three communities.

This concept of probabilistic clustering or overlapping clustering is more general than the non-overlapping clustering provided by our naive procedure. Note that an overlapping clustering can always be collapsed into a non-overlapping clustering by majority voting, or any other rule. It also finds applications for automated trading strategies. We recall that a trading strategy is a weighting over different underlying assets. If a trading strategy were evaluated on each community for diversification purposes, the weights put onto each stock could be re-weighted by the stock's weight in the community.

Finally, this view has economic interpretation too. The idea of clustering stocks is not new. Companies are already clustered into sectors, industry groups, industries and sub-industries. This is the Global Industry Classification Standard developed and maintained by MSCI and Standard & Poor's. These standards define a unique sector, industry group, industry and sub-industry per company. It thus corresponds to a non-overlapping clustering; and both companies review at least annually the classification of companies and base their decision on the definition of the companies' principal business activity as determined by MSCI and S&P. Let us take the example of Apple Inc. (AAPL). It is classified in sector 'Information Technology' and sub-industry 'Technology Hardware, Storage and Peripherals'. This sub-industry is defined in the GICS by :

> Manufacturers of cellular phones, personal computers, servers, electronic computer components and peripherals. Includes datastorage components, motherboards, audio and video cards, monitors, keyboards, printers, and other peripherals. Excludes semiconductors classified in the Semiconductors Sub-Industry.

But still, as of March 27, 2021 Apple's three months ended net sales were at 18.87% from their Services business line. Bearing this in mind, it would seem natural to define Apple's classification as a weighting over 'Technology Hardware, Storage and Peripherals' and 'Internet Services & Infrastructure' for their cloud storage solutions which are included in their Services category in terms of sales.

Data is shown in Table 6 and comes from Apple's 10-Q.

Table 6: Apple's 3 months ended net sales by category as of 2021-03-27

net sales per category	Dollars in millions
iPhone	47938
Mac	9102
iPad	7807
Wearables, Home and Accessories	7836
Services	16901
Total net sales	**89584**

3.5.2 Our bootstrapped clustering procedures

In what follows, a bootstrapped clustering simply refers to a repetition of the clustering procedure to obtain a series of clusterings for a given time t. It is exactly the same object we introduced under section 3.4. What differs here is how we use this series of clusterings to accept a clustering for the corresponding time stamp. In section 3.4, we designed a procedure to accept one clustering of this series as the true clustering. In this section, we develop two procedures which use all the clusterings in the series to output a new, robustified clustering for this time stamp.

Below we describe two distinct ways of using bootstrapped clusterings to estimate more robust clusterings. The first alternative is the one based on co-occurrence matrices. It does not output a probabilistic clustering;

but rather outputs a strictly non-overlapping clustering like the vanilla Louvain method would.
Interestingly, this approach focuses on the dynamics of pairs of stocks during the bootstrap of the clusterings leading to a very interesting property, as well as unexpected applications for pairs-trading strategies. The major result from this new procedure is the mapping of the initial empirical correlation matrix used to estimate clusterings to an adjacency-like matrix. We recall that we had to consistently redefine Louvain method for correlation matrix precisely because correlation matrices are no adjacency matrices. We name the adjacency-like matrix that can be estimated from bootstrapped clusterings a co-occurrence matrix, where each entry represents the probability of two stocks ending up in the same community. The co-occurrence matrix can then be collapsed into a strict non-overlapping clustering using the vanilla Louvain method, efficiently implemented in the R function `igraph::cluster_louvain`. Hence, the co-occurrence matrix is a very powerful yet simple object which contains information from all the bootstrapped clusterings and that can be collapsed into a non-overlapping clustering. This approach has never been discussed in academic papers to the best of our knowledge.

Our second alternative procedure outputs a probabilistic clustering where each community is defined as a weighting over all the stocks in the stock universe. The price of obtaining such information is that the information on pairs of stocks or co-occurrence of stocks is lost. Such clustering information allows more general trading strategies. As mentioned before, it can be used to re-weight trading strategies on individual communities by the probabilities of each stock to belong to these communities. It is also a more consistent approach from the economic point of view since it allows for weighted classifications and may convey information about companies' business segmentation.

3.5.2.1 Alternative 1 : Co-occurrence matrices The first alternative bootstraps clusterings vertically[16]. The procedure is graphically depicted in Figure 12.

The procedure to estimate the co-occurrence matrix from bootstrapped clusterings is described in the following box.
Suppose M repetitions of the hierarchical clustering.

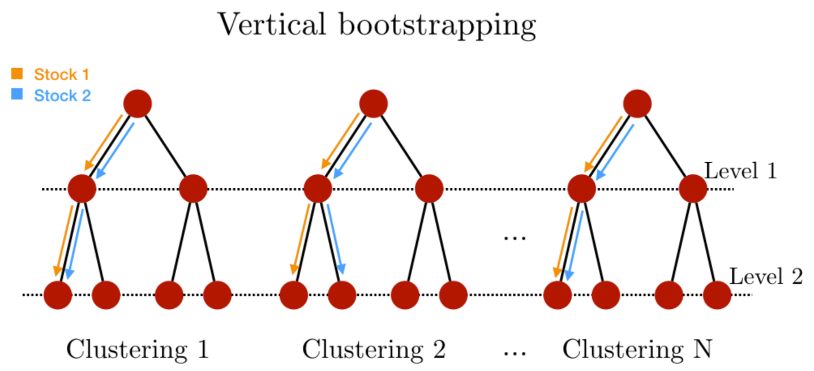
Algorithm for vertical bootstrapping of hierarchical clusterings.

1. Estimate a full hierarchical clustering.
2. Record all communities and sub-communities.
3. Repeat steps 1-2 until M clusterings are obtained.
4. Turn the information into a co-occurrence table which records for each pair of stocks the frequency at which they ended up in the same community, or sub-community.
5. Either use the co-occurrence matrix as is; or collapse it into a strict clustering by running vanilla Louvain on it. Note that a co-occurrence matrix is adjacency-like and original Louvain method can be used.

An example of such co-occurrence matrix is shown in Table 7. Entries represent the frequency of co-occurrence of pairs of stocks - i.e. the frequency at which they end up in the same community when bootstrapping communities. Co-occurrence matrices can be interpreted as weighted adjacency matrices with entries indicating the intensity of the link between pairs. This means that the original Louvain method can be applied to obtain a non-overlapping clustering.

To wrap up, this first alternative bootstraps the clusterings, but no matching is made between communities; only stocks' co-dynamics are considered. The output of the method is not a clustering per se; it is a co-occurrence matrix. We can then collapse the co-occurrence matrix into a strict non-overlapping clustering by applying the original Louvain method. Sizes of communities obtained at the highest hierarchical level are shown in Table 8. Note that communities' sizes, when collapsed into a strict clustering, are similar to the most central partitions found when optimizing the L_1-median in Table 5.

[16]Here vertical bootstrapping means that full clusterings are estimated at each pass. This is opposed to horizontal bootstrapping where clusterings are bootstrapped level by level. Initially, only the first level is bootstrapped; then when all the passes



Co-occurrence matrices

	Stock 1	Stock 2
Stock 1	1	1
Stock 2	1	1

Level 1

	Stock 1	Stock 2
Stock 1	1	$2/3$
Stock 2	$2/3$	1

Level 2

Figure 12: Illustration of the vertical bootstrapping procedure. The method records the dynamics of pairs of stocks during the bootstrapping of the clusterings. M clusterings are estimated. At each level, the frequency at which two stocks end up in the same community is stored as an entry of the co-occurrence matrix. The numbers used here represent the three clusterings depicted in the example. The co-occurrence matrix can then be interpreted as a weighted adjacency matrix for further applications. Note that one co-occurrence matrix is estimated per hierarchical level.

Table 7: Example of a subset of a co-occurrence matrix

	007d3258	00ce3e92	016d5512	0179bdd5	0251dcf4	02685522	031429b2
007d3258	1.000	1.000	0.317	1.000	0.033	0	1.000
00ce3e92	1.000	1.000	0.317	1.000	0.033	0	1.000
016d5512	0.317	0.317	1.000	0.317	0.717	0	0.317
0179bdd5	1.000	1.000	0.317	1.000	0.033	0	1.000
0251dcf4	0.033	0.033	0.717	0.033	1.000	0	0.033
02685522	0.000	0.000	0.000	0.000	0.000	1	0.000
031429b2	1.000	1.000	0.317	1.000	0.033	0	1.000

Table 8: Highest hierachical level communities' sizes obtained by collapsing the co-occurrence matrix into a strict clustering.

Community name	Size
0001	138
0002	122
0003	70
0004	161

To conclude this section on non-overlapping clusterings, we propose an interesting data visualization which is recurrent in the asset clustering literature : the reordering of the correlation matrix into a block diagonal matrix. Indeed, we can use our hierarchical clustering data to reorder the correlation and correlation-like matrices into block diagonal matrices, highlighting the hierarchical structure with the communities and the sub-communities.
One can see in Figure 13 the reordering of correlation/filtered correlation and co-occurrence matrices with highlighting of the hierarchical structures[17].

3.5.2.2 Alternative 2 : probabilistic clustering The second alternative that we developed bootstraps clusterings horizontally. This means that hierarchical communities are bootstrapped level by level. The procedure is graphically depicted in Figure 14.

The bootstrapping procedure is described in the following box.

Algorithm for horizontal bootstrapping of hierarchical clusterings.

1. Define a depth n for the probabilistic clustering.
2. Bootstrap the first hierarchical level. Rename clusters according to a reference clustering and identify for each stock its frequency of occurrence within each community.
3. For each identified community consider all stocks which ended up in this community at least once at previous step; and repeat steps 2-3 until the desired depth is reached.

The output of such procedure is shown in Table 9. The `data.table` identifies for each stock all the communities it belonged to during the bootstrap; and the frequency of occurrence is shown as well, interpreted as the probability of belonging to the community.

Finally, the communities are defined as weightings over stocks. Hence, each community is composed of all the stocks with varying weights. A sample is shown in Table 10 to illustrate the shape of the output.

are done for the first level, the second level is bootstrapped and so on.

[17]The plotting function was developed by Thierry Bochud, many thanks!

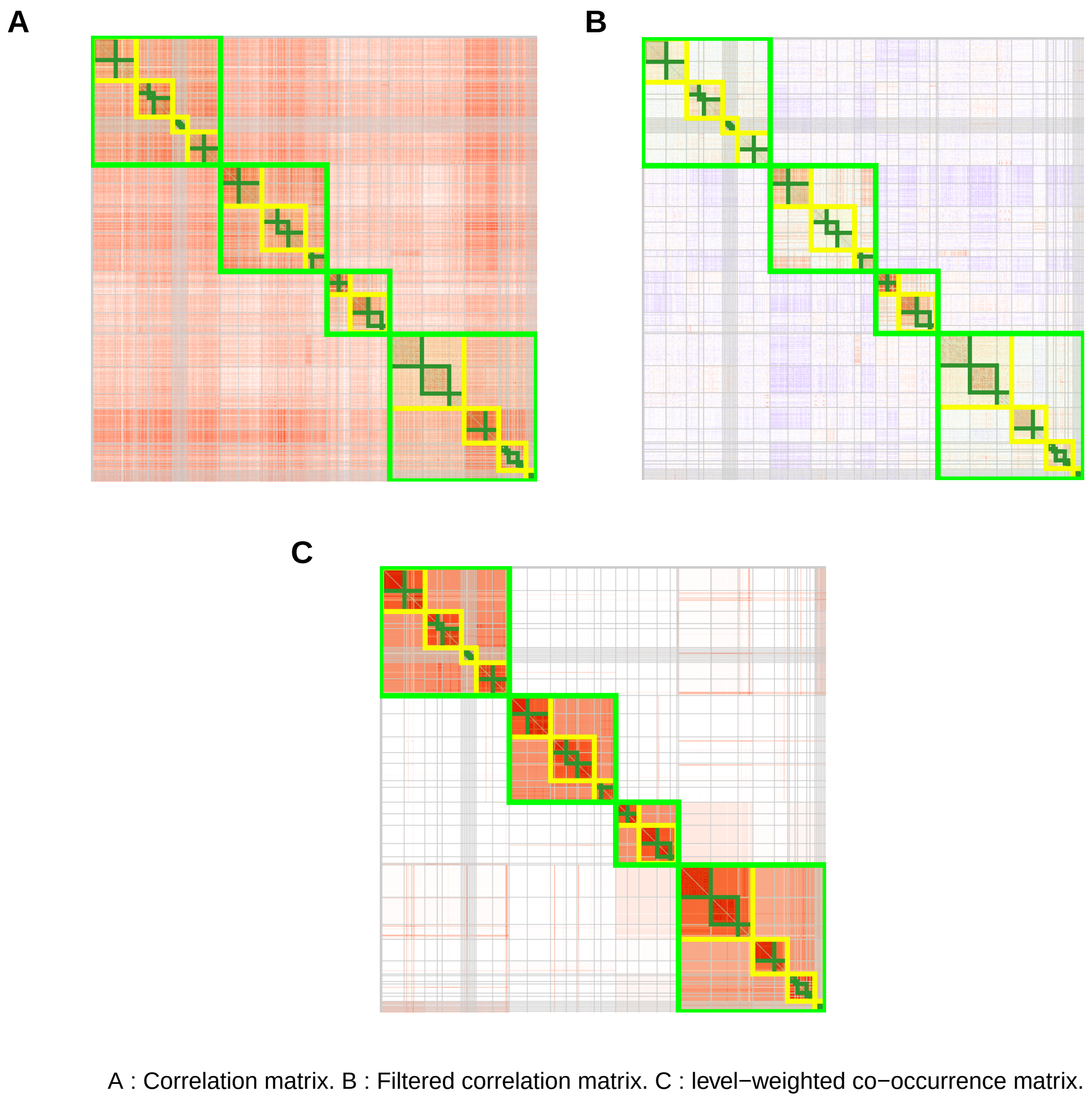


Figure 13: Vizualization of communities and hierarchical structures on top of correlation matrix, or correlation-like matrices. The grey lines are artefacts from the construction of the cluster structure on top of the matrices. One can see how nicely the blocks structure is revealed in the filtered correlation matrix which contains correlation structure at the communities levels.

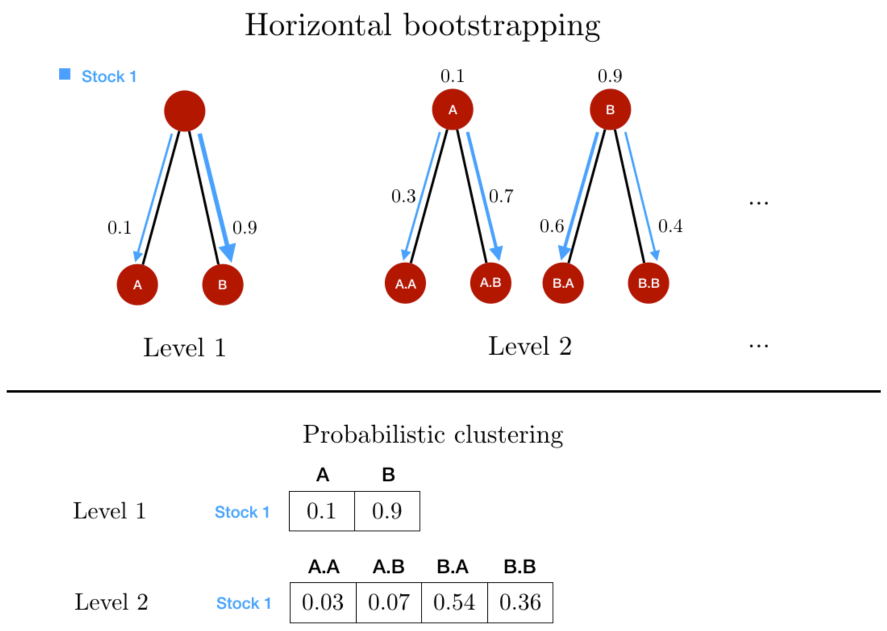


Figure 14: Illustration of the horizontal bootstrapping procedure. In this method, clusterings are bootstrapped level by level.

Table 9: Output of the probabilistic clustering procedure. 10 first rows

Ticker names	Hierarchic level	Community names	Frequency of occurrence
007d3258	Level.1	0,0002	0.0166667
007d3258	Level.1	0,0004	0.9833333
007d3258	Level.2	0,0002,0001	1.0000000
007d3258	Level.2	0,0004,0003	1.0000000
00ce3e92	Level.1	0,0004	1.0000000
00ce3e92	Level.2	0,0004,0001	1.0000000
016d5512	Level.1	0,0001	0.6666667
016d5512	Level.1	0,0004	0.3333333
016d5512	Level.2	0,0001,0001	1.0000000
016d5512	Level.2	0,0004,0002	1.0000000

Table 10: Definition of communities as weightings over stocks. The 10 first rows are shown for the first community only.

Ticker names	Community names	Weights
007d3258	0,0001	0.0000000
00ce3e92	0,0001	0.0000000
016d5512	0,0001	0.6666667
0179bdd5	0,0001	0.0000000
0251dcf4	0,0001	0.9500000
02685522	0,0001	0.0000000
031429b2	0,0001	0.0000000
03840ba2	0,0001	0.0000000
046841cc	0,0001	0.4333333
049d72de	0,0001	0.0000000

4 Applications

In this section we provide examples of applications where our two clustering procedures are applied to trading strategies, as well as further improvements to our clustering algorithm. We study the effect of the different correlation methods on clustering results and introduce the notion of dynamic communities to track clusters over time to finally propose a procedure even more robust than our original method.

4.1 Review of the parameters at hand

The possible applications are broad, and listing the parameters that can be tuned in our clustering procedure is the first step to carry out.

The parameters that can be tuned when running the clustering procedures are:

- **Input data**
  - The shape of the returns matrix N and T. This will impact the value of $Q = \frac{T}{N}$ the dimensionality ratio. N would typically be fixed by the underlying investment universe. Throughout this work the universe was the S&P500 with $N \approx 500$, constant over time. The time parameter T will have a crucial impact on the output of the clustering procedure : it needs to be as large as possible for results from random matrix theory to hold; but it should be small enough to find interesting applications and handle non stationarities. A large value of T means a stable clustering and a smoothed-out correlation dynamics due to the large history used. A smaller value of T will better capture the correlation dynamics but may lead to unstable clusterings; as well as to the breakdown of the hypothesis from random matrix theory.
  - The time series used can be log-returns or signed returns (suggested by Almog et al. [2015]) which consists in replacing returns data with their sign -1 or $+1$ to robustify the correlation matrix estimates. For most of the following applications signed returns are used.
- **Correlation matrices estimates**
  - The type of correlation considered can be Spearman, Kendall, Pearson or Gaussian rank (Boudt et al. [2010]). For computation time considerations Spearman and Kendall correlations were discarded; but Pearson and Gaussian rank correlation are investigated. Nevertheless, for most applications Pearson is used by default for computation time considerations.
  - The amount of admissible missing data for a stock to be included in the correlation matrix is tunable and defaults to 10% of the desired number of data points per stock. Allowing for more missing data would keep N closer to its original chosen value because less stocks would be dropped. But this would induce more inconsistencies in the correlation matrix estimates. For example, correlation data from varying time periods within the lookback window would be aggregated to the same time stamp : the rebalancing date considered. Suppose a stock joined the investment universe at the very end of the lookback window; this stock may have missing data at the beginning of the lookback window[18]; and only its latest correlation dynamics will be captured on the rebalancing date; correlation that will de facto be compared to that of other stocks which belonged to the universe during the whole period; hence aggregating asynchronous data to the same time stamp.
- **Correlation matrix cleaning**
  - The eigenvalue threshold $\lambda_{\max}$ can be obtained by various methods, as seen in Chapter 3. The default method will be our unsupervised procedure to find a best $\lambda_{\max}$ when the number of assets in the universe is greater than 300. For all values lower than 300, the empirical eigenvalue threshold is used. Basically, for a recursive clustering on the S&P500, the first level $N \approx 500$ uses our unsupervised procedure; while deeper levels will most of the time use the empirical

[18]For example, Facebook joined the S&P500 in December 2013, a few months after its IPO on May 2012.

eigenvalue threshold from our dictionary since communities' sizes will usually be smaller than 300 with S&P500 data.
  - The null-model as a decomposition of the empirical correlation matrix can also be tuned through the number of modes included in the market component. Again the default behavior of our procedure is fixed. At level i, the 0 to i-th largest eigenmodes are successively included in the market mode as long as the algorithm cannot output a partition sufficiently different from the null-partition. This procedure guarantees that no large eigenvalues are sacrificed to the market-mode if clustering information can be extracted from such largest eigenvalues.

- **Clustering procedure**
  - The percentage of randomization of the communities during the clustering procedure can be tuned. Randomization was designed to overcome local optima and to explore a larger volume during the optimization process. A larger value of the shuffling percentage will induce more variability in the bootstrapped clusterings, but will allow for broader exploration of the clustering space.
  - The depth of the clustering as well as some conditions on hierarchical clustering can be imposed; like some conditions on the minimum community size to keep on iterating within a community. In what follows, the clustering procedure was stopped at the first hierarchical level for computation time considerations and to keep simpler settings.
  - Finally, applications can be based on any of the two possible clustering alternatives. We recall that the first alternative outputs co-occurrence probabilities which represent the probabilities that any two stocks end up in the same community; for all possible pairs. Those co-occurrence probabilities can then be collapsed to a strict clustering using the vanilla Louvain algorithm. The second alternative outputs overlapping communities, where each obtained community is a weighting over all assets in the investment universe; and reciprocally, any asset is a weighting over the different communities.

- **Returns data frequency**
  - The frequency at which returns are sampled is the finest frequency than can be used in our clustering procedure. But a time series of returns can be aggregated to obtain returns data at lower frequencies. We recall a basic property of log-returns : $r^{\log}_{0\to1} = \frac{P_1}{P_0} = \log P_1 - \log P_0$ and $r^{\log}_{1\to2} = \frac{P_2}{P_1} = \log P_2 - \log P_1$; hence $r^{\log}_{0\to2} = \frac{P_2}{P_0} = \log P_2 - \log P_0 = r^{\log}_{0\to1} + r^{\log}_{1\to2}$; where P_i is the price of the asset at time i. A time series of daily log-returns can be easily aggregated into a time series of weekly log-returns. Throughout this work we have been working with daily log-returns. Having access to higher frequency data is an efficient way of increasing T while keeping N constant, hence increasing Q, while keeping the lookback period the same.
  - The portfolio rebalancing frequency for trading applications can also be chosen; summing the higher frequency log-returns between two rebalancing dates to obtain the realized return. Weekly rebalanced portfolios were designed for the presented applications. This helped reduce the full-sample computation time down since clusterings were computed only once a week over the full data sample, a total of 1181 weeks. Any frequency can be chosen for computing clusterings or for rebalancing portfolios. For instance, clusterings can be estimated everyday but used only once a week for portfolio rebalancing; and clusterings data accumulated between two rebalancing dates can be used for other purposes, like for market monitoring or to detect more subtle changes in dynamics.

4.2 The effect of the different correlation methods on clustering

Kendall and Spearman rank correlations were discarded because their computation takes too long with such high-dimensional input as the S&P500 at daily frequency. At this point, only Pearson and Gaussian rank correlation estimators are considered.

To study the effect of the different correlation measures and the different types of inputs (either log-returns or signed log-returns) on our clustering procedure, we sampled some bootstrapped clusterings over the years

2018-2019 at a weekly frequency. On each Wednesday of the years 2018 and 2019, about 100 clusterings were bootstrapped for each combination of the parameters (correlation measure, type of input). The mean number of communities found with each combination of the parameters over the years 2018-2019 is shown in red in Figure 15. Each bootstrapped clustering was then collapsed into a strict clustering by using co-occurrence frequencies tables; this is the first alternative of our bootstrapped clustering procedure. The number of communities obtained for each combination of the parameters after collapsing the co-occurrence matrix into a strict clustering is also shown in blue in Figure 15.

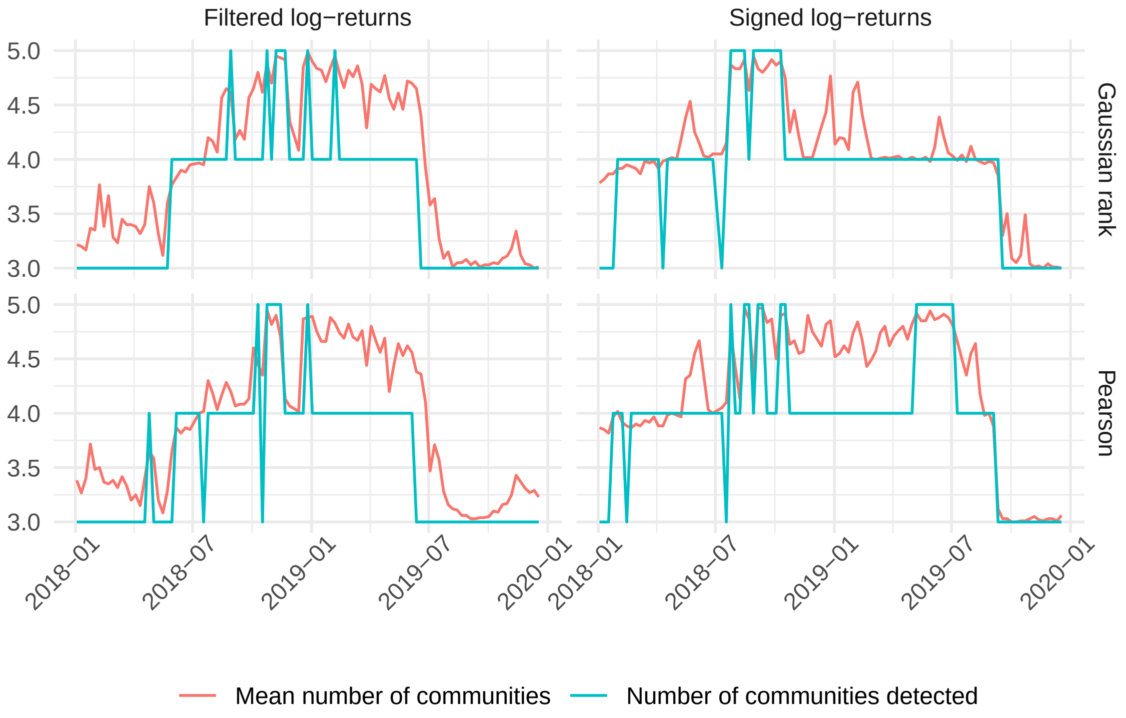


Figure 15: Mean (red) and actual (blue) number of communities found over the years 2018-2019 at a weekly frequency, for each combination of the parameters (correlation measure, type of input). The first insight is the gain in stability over time obtained by collapsing the bootstrapped clusterings into a strict clustering; which captures the change from 4 to 3 communities in the summer/fall of 2019 for all pairs of the parameters. Another insight is the time lag between the two types of inputs. The drop from 4 to 3 communities happens much earlier with log-returns than with the less informative signed log-returns.

An interesting observation is that when input data is fixed, the correlation method has little effect on the number of communities detected. Indeed, when collapsing the bootstrapped clusterings into strict clusterings, the time series of how many communities are detected is very much alike; except for one-off peaks and a persistent regime at 5 communities in mid 2019 with the Pearson correlation estimator and signed returns. Therefore, the type of correlation measure used does not have a great impact on the clustering data; and it is not clear whether one should use Pearson or Gaussian rank correlation to estimate empirical correlation matrices.

Another observation is that, in the second half of 2019, all time series show a sharp transition from a regime of around 4-5 communities; to a regime where only 3 communities are detected. The sharp drop from 4 to 3 happens synchronously at fixed type of input data; and it happens earlier with filtered log-returns and later with signed log-returns. We interpret this as a drawback from using signed log-returns : there may be a loss of information and regime transitions take more time to be detected.

There is not a single outstanding pair of parameters which seems definitely better than the others. All

4 configurations show stable regimes and rather synchronous transitions; and all configurations showcase instabilities with sudden peaks in the detected number of communities. Bearing this in mind, the most robust and conservative approach is to use signed log-returns which filter out extreme events and only focus on the synchronicity of up and down moves in stock prices; together with Pearson correlation which saves a lot of computation time for full-sample applications.

In Figure 16 we show the distribution of the communities' sizes in bootstrapped partitions obtained for 27-11-2019. This date is chosen because all combinations of the parameters lead to 3 detected communities as can be seen from Figure 15. For each combination of the parameters, the 30 most central bootstrapped clusterings are selected[19]. One can see from this particular date that Gaussian rank correlation with signed log-returns present the most variability in communities' sizes, while the three other combinations seem rather equivalent. This is an additional incentive to use the combination of Pearson correlation and signed log-returns which provides robustness to outliers and efficient computation.

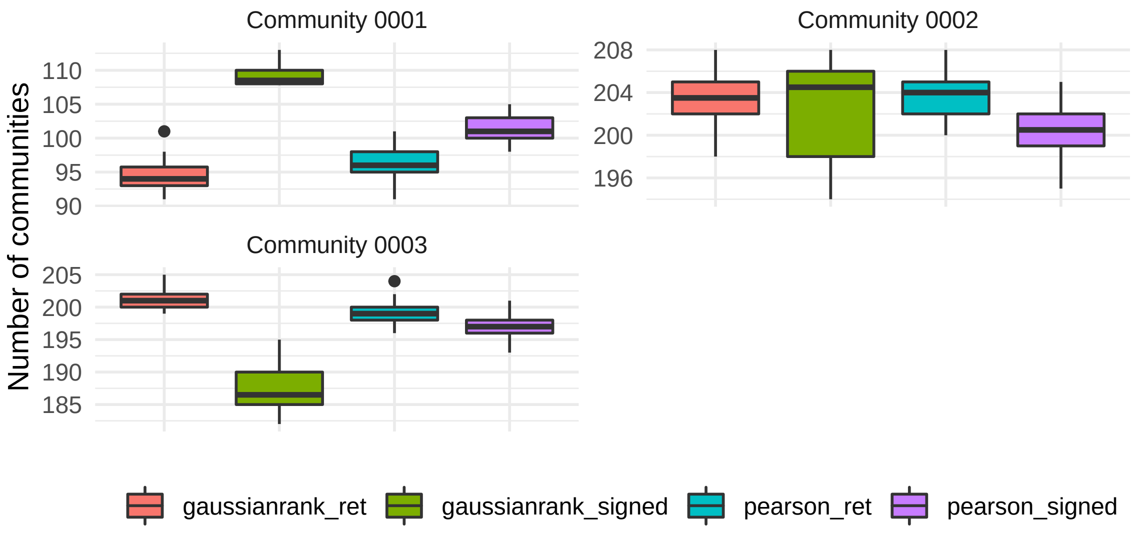


Figure 16: Boxplot for the distribution of the number of assets in each community on 2019-11-27. For each combination of the parameters, the 30 most central bootstrapped partitions are selected using the L1-median over the Variation of Information matrix. The distribution of the number of assets in each community for each combination of the parameters is then computed from those 30 best partitions.

4.3 Dynamic communities

Our algorithm can compute clusterings or bootstrapped clusterings at different time stamps; it is a rolling algorithm which is carried out over the time dimension of the dataset. It is important to stress that clusterings found at dates t and $t + 1$ use the same naming convention for clusters[20], but two clusters with the same name at different time stamps might not match with each other; this is exemplified in Figure 17 where the labels of communities 0001 and 0002 are exchanged on 01-04-2005.

Clusters can undergo many different events throughout their life. Clusters are born, can persist over time, can split or merge, and eventually die. Due to these many different life paths, being able to follow and identify clusters over time and to match present clusters to past clusters is not a trivial task.
Some applications may neither require to identify clusters through time, nor to characterize the evolution of a cluster over time; but some will. We propose to implement an algorithm developed by Liechti and

[19]Here centrality is measured using the L1-median over the Variation of Information matrix, as it was defined in 3.4.1
[20]0001; 0002; 0003 and so on for first level clusters. Second level clusters follow the convention "name of the parent cluster, name of the cluster" to track the affiliation to higher levels clusters : 0001,0001; 0001,0002; 0002,0001 and so on.

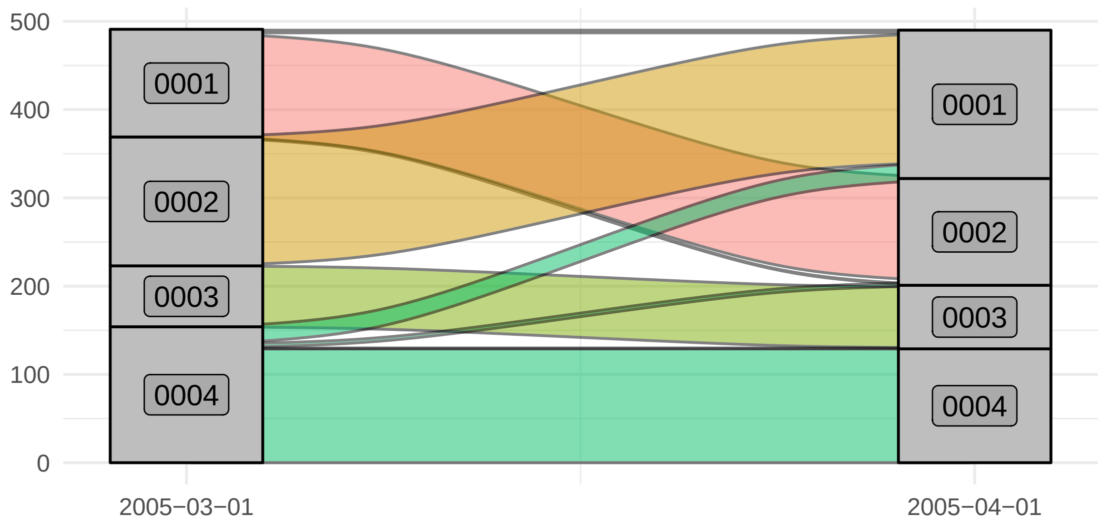


Figure 17: Same Sankey plot as shown before where labels of communities 0001 and 0002 are exchanged on the later date. Labels are clearly not informative over time and they should be updated to match previous time stamp clusters. Here, the cluster labelled 0001 at time t matches almost entirely with the cluster labelled 0002 at time $t+1$, and 0002 at time t matches to 0001 at $t+1$. It is necessary to have a procedure renaming clusters to match them accordingly to clusters from the prior time stamp, allowing to track clusters over time.

Bonhoeffer [2019] which detects dynamic clusters from a time series of clusterings. Specifically, the method allows for detection of persistent structural elements with internal dynamics; which is the kind of object we are dealing with. Indeed, the communities we observe over time are expected to persist[21] with internal dynamics[22].

The algorithm is based on the identification of majority overlaps between clusters at different time stamps. The algorithm is rolled over the time series of clusterings and at each time stamp t individual clusters are either matched with a dynamic cluster in the past; or are assigned to a new dynamic cluster, representing a birth. The parameter $t_{\max}$ is used to control how far in the past the algorithm can go to try and find a matching cluster. More precisely, on time stamp t the algorithm will check at time stamps $\{t-1, \cdots, t-t_{\max}\}$ recursively until it finds a matching cluster. If none is found, the cluster is assigned a new dynamic cluster.

Note that this algorithm which rolls over a time series of clusterings has some retroactive action on its output. At time t it can modify its output up to $t-t_{\max}$. This means that the whole output of the algorithm on the final time stamp (which is a time series of dynamic clusterings) is only known on the final time stamp. Using the output of the algorithm without proper care to backtest applications would induce some forward-looking bias.

To avoid inducing a bias in the way one sees such Sankey diagram (Figures 11 and 17); we should be careful with the labeling of clusters and with the colors used to highlight the flows. Such neutral diagram is shown in Figure 18. This diagram only shows flows of stocks between the different communities for the first 10 clusterings of 2018 at weekly frequency.

From this blind time series of clusterings we can detect dynamic clusters using our implementation of the algorithm by Liechti and Bonhoeffer [2019]. Figure 19 displays the Sankey diagram from Figure 18 but color coded with the different dynamic clusters. Note how the labeling of the clusters is now consistent over time; and how the small clusters that split are now highlighted as new dynamic clusters of their own. The

[21] A group of stocks with similar dynamics is expected to be identified as such over the following time stamps.
[22] Splitting, merging, dying and so on.

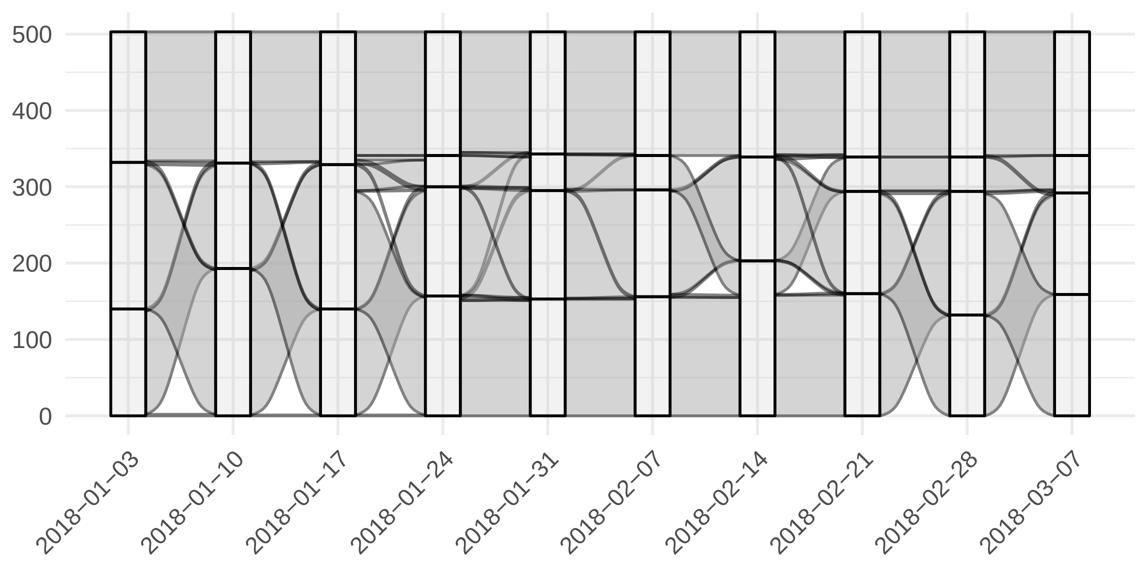


Figure 18: Neutral Sankey diagram where clusters' labels are not shown. One cluster at the top seems persistent over time and the labeling of two clusters seems to oscillate (see the clusters supposedly labelled 0002 and 0003 at the beginning of the period). A small cluster detaches itself from a larger one between 17-01-2018 24-01-2018 and merges back on 14-02-2018.

parameter $t_{\max}$ is set to $t_{\max} = 3$; which means that at each time stamp the algorithm looks for a matching cluster up to 3 time stamps in the past. It is now clearly identifiable that the orange cluster detaches from the purple cluster on 14-01-2018 and merges back on 14-02-2018.

We can tune the parameter $t_{\max}$ which defines how far in the past the algorithm can go to find the cluster's source set. When setting a larger $t_{\max}$ like $t_{\max} = 4$; the algorithm understands that the orange cluster which lived on its own for a while in Figure 19 merges back on 14-02-2018; and hence interprets the orange cluster as belonging to the purple cluster when sitting on date 14-02-2018. The algorithm still interprets the orange cluster as a dynamic cluster of its own when sitting on date 07-02-2018 since at this date it has not merged back yet. It is important to note that this algorithm which is a rolling algorithm can modify previous time stamps results, as demonstrated here in Figure 20

This algorithm allows to consistently follow and match clusters over time, and the only parameter is the so-called history parameter $t_{\max}$. Nevertheless, it is not clear what value of $t_{\max}$ should be chosen and dynamic clusters can substantially differ depending on the value of $t_{\max}$, like we just showed. This matching procedure is not used in our other applications but we believe it could be used as input to market monitoring or more advanced trading applications.

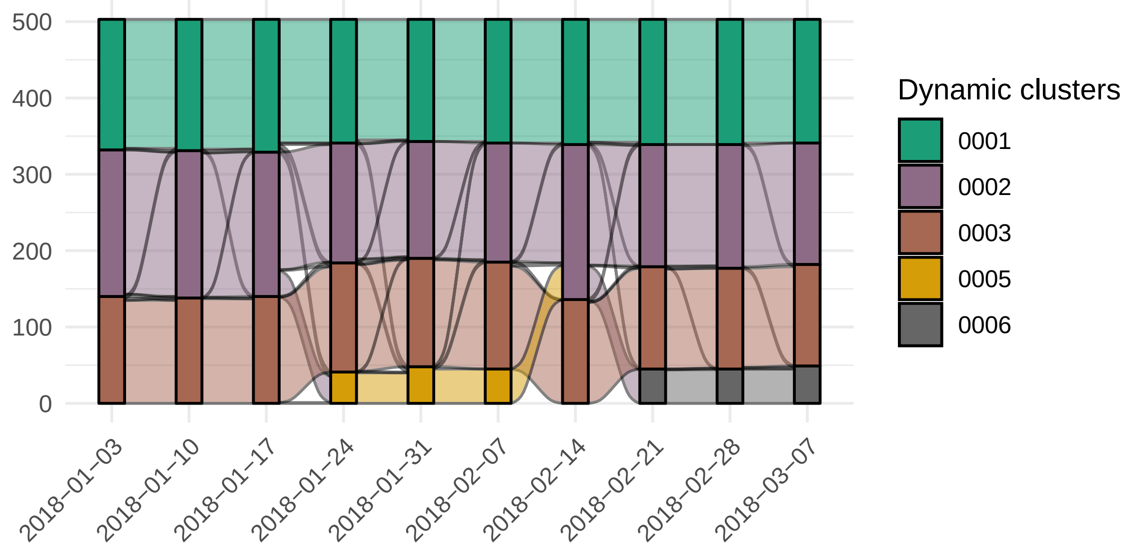
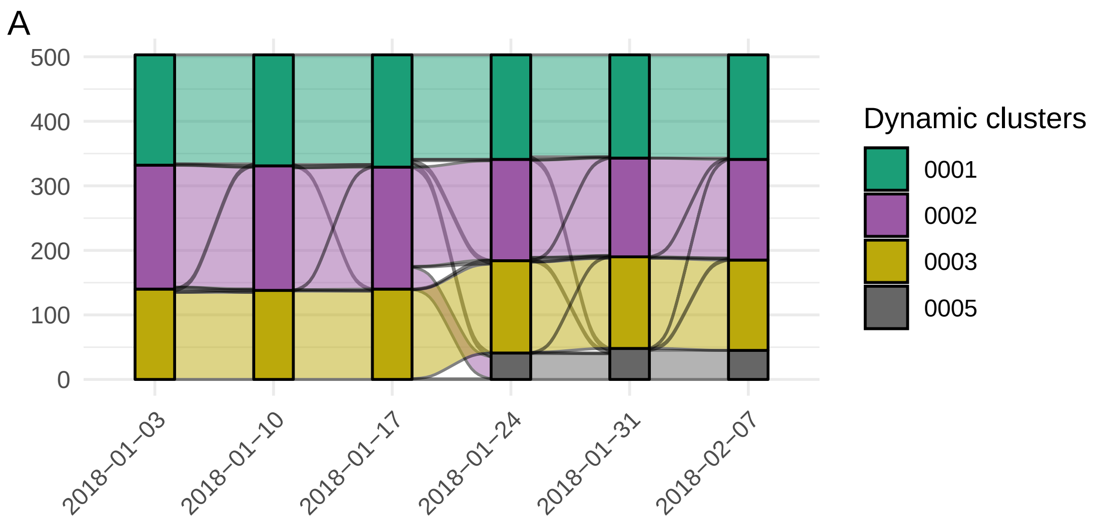
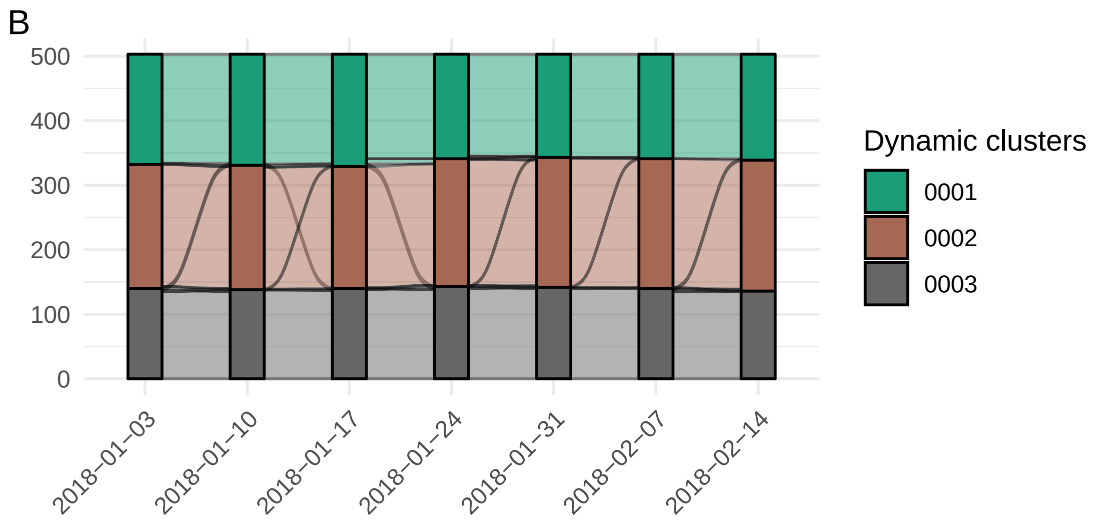


Figure 19: Dynamic clusters as seen from 07-03-2018. This output is only known on the date 07-03-2018.





A : Dynamic clusters on date 02−07−2018. B : Dynamic clusters on date 02−14−2018.

Figure 20: Dynamic clusters identified with the history parameter set to $n_{\max} = 4$. A : On 07-02-2018 dynamic cluster 0005 does not make a bijective match with any older cluster except itself. It is still identified as a cluster on its own on this date. B : On 14-02-2018 cluster 0005 merged back with cluster 0002 (see Figure 19). The algorithm then merges cluster 0005 to 0002 on all previous time stamps when 0005 was on its own.

4.4 Robustifying procedure : bootstrap across parameters

In section 3.5 we introduced the notion of bootstrapped clustering; and we studied its stability to varying parameters in section 4.2 where we concluded that there is no obvious better choice for the pair of parameters (correlation measure, input data).

In addition to bootstrapping clusterings at each time stamp with fixed parameters like it was done before; we propose to also bootstrap across parameters. For instance, we could use all the bootstrapped clusterings obtained when comparing the correlation measures and the input data; and aggregate them into one bootstrapped clustering. Since not one set of input parameters seems favored, why not use all of them? Or at least a subset of them. This aggregation could once more robustify our co-occurrence matrix and lead to more robust strict clusterings once the co-occurrence matrix is collapsed using the vanilla Louvain algorithm.

In Figure 21 we did such aggregation and plotted the mean number of communities detected when collapsing the co-occurrence matrix as well as the mean number of communities detected at each time stamp in the aggregate of bootstrapped clusterings. This Figure is to be compared with Figure 15. The gain in stability is impressive, and the general behavior remains the same with a transition from 3 to 4 communities at the beginning of the period, and the transition from 4 to 3 communities in the summer of 2019.

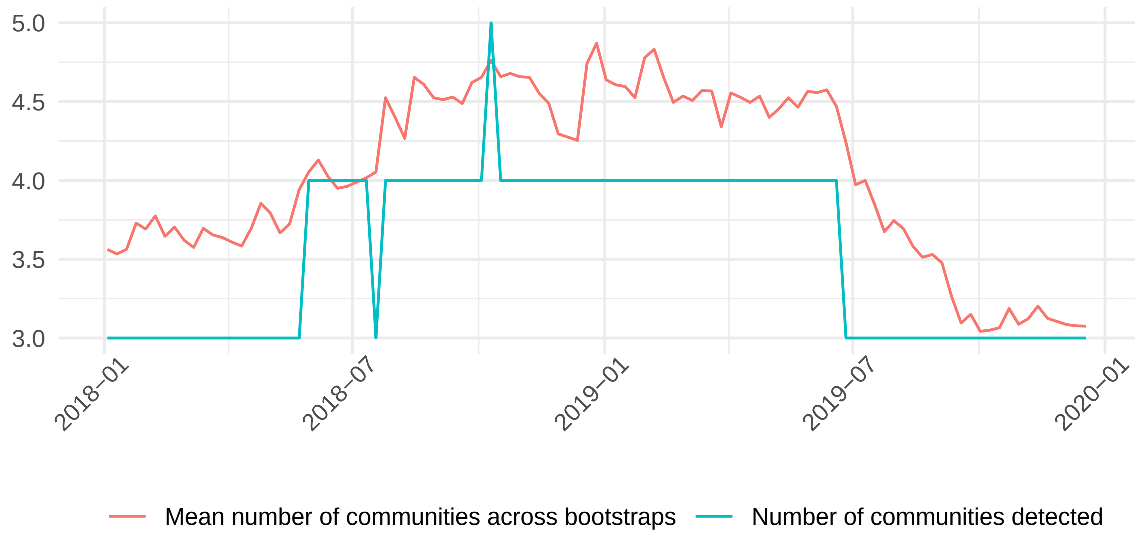


Figure 21: (red) Mean number of communities detected in the aggregate of bootstrapped clusterings with different input parameters; and (blue) the number of communities detected when collapsing the robustified co-occurrence matrix with vanilla Louvain algorithm. The gain in stability from aggregating all bootstraps is sizable and regimes of low and high number of communities are clearly identifiable. Over the time window, only two one-off deviations from stable regimes happened.

In Figure 22 we show the corresponding blind Sankey diagram over a time window which contains the transition from the 3 to 4 communities regimes as well as the one-off jump to 3 communities which happened in mid 2018 (see Figure 21). The corresponding color coded diagram using our matching procedure from section 4.3 is shown in Figure 23.

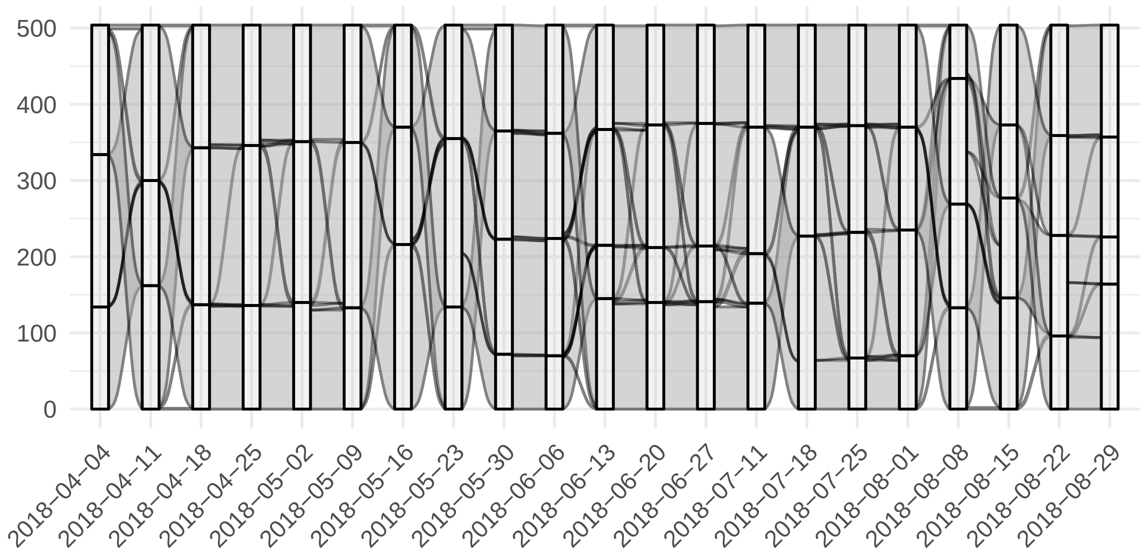


Figure 22: Neutral Sankey diagram where clusters' labels are not shown. The selected time period comprises the change of regime from 3 to 4 communities taking place in summer 2018 as well as the one-off jump from 4 to 3 communities in July/August 2018.

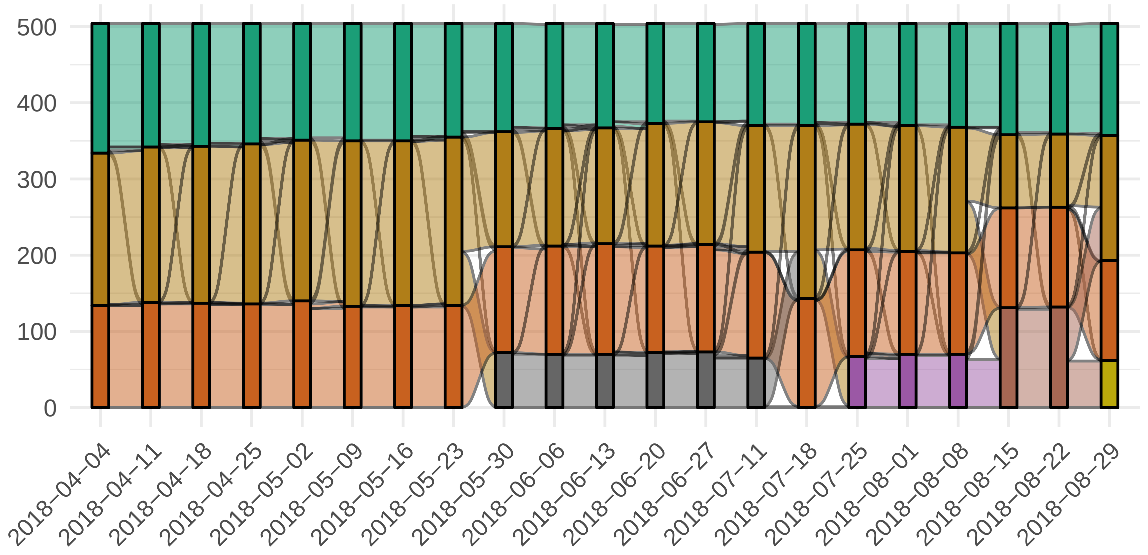


Figure 23: Corresponding Sankey diagram color coded with the dynamic clusters labels. The clusters' labels are not shown in the legend to allow for a better comparison to Figure 22 by eye. The selected time period comprises the change of regime from 3 to 4 communities taking place in summer 2018 as well as the one-off jump from 4 to 3 communities in July/August 2018.

4.5 Trading and portfolio construction applications

Two kinds of applications are evaluated. The first type is based on the partition of the investment universe into communities of assets, within which traditional strategies will be implemented (momentum and Markowitz portfolio selection) and aggregated into one portfolio.
The other type of strategies implemented is not based on actual partitions of assets into communities, but rather builds on the properties of our procedure in its first alternative which provides insights on pairs of assets. More precisely, a pairs-trading strategy is implemented; as well as a strategy which creates portfolios of uncorrelated stocks in the sense that assets in such portfolios are paired[23] to the least number of other assets.

4.5.1 Momentum and Markowitz optimization inside communities

A direct application of clustering within traditional trading strategies is to bring diversification by forcing multi-sector (or in our case multi-community) allocation. We will illustrate this property with two vanilla trading strategies : momentum trading and global minimum variance (GMV) portfolio trading.

4.5.1.1 Momentum Momentum strategy consists in buying the top quantile and selling (shorting) the bottom quantile of the trading universe; where assets are ranked by their past performance. The use of our clustering procedure should reduce concentration in the final portfolio of top performing sectors and avoid losses from sector-wise mean reversion[24]. The proposed procedure for each rebalancing date is detailed in the following box.

Procedure for momentum portfolio construction using clustering data

1. Obtain a clustering for this trading day, based on past returns data.
2. Apply the momentum strategy within the communities, creating as many portfolios as there are communities.
3. Aggregate the different portfolios into one large portfolio, and weighting each community by its inverse volatility $\frac{1}{\sigma}$[a].

[a]Such a weighting scheme is called risk-parity, with each component contributing equally to the volatility of the aggregated portfolio.

Figure 24 shows the cumulative log-return of both the community-wise and the naive momentum strategies for different strategy settings. Both strategies were backtested for different values of the fraction of assets used, from 1% to 20% of assets from the investing universe for both long and short positions. A value of 50% would mean that 50% of assets would be bought and 50% sold, which amounts to trading the whole universe. Note that all portfolios were leveraged to yield a constant 10% annualized volatility based on rolling past volatility; which allows for comparison of the different portfolios' return on a risk-adjusted basis.
The naive and the community-wise strategies performed similarly until 2015; and the community-wise momentum strategies outperformed the naive setting between 2015 and 2020.

The 50-week rolling Sharpe ratios for both the naive and the community-wise strategies are shown in Figure 25. Even though the community-wise strategies outperformed the naive strategies on a risk-adjusted basis, their rolling Sharpe ratios are similar and extremely noisy.

Further full-sample risk measures are shown in Table 11 for portfolios with 15% of assets in both long and short legs. The downside standard deviation is a measure of downside risk which is computed as the standard deviation of negative returns. It measures the volatility in the negative part of the returns distribution. The worst drawdown is the maximum loss from a peak to a trough of a portfolio, before a new peak is attained. It is also an indicator of downside risk over the backtested period of time.

[23]A proper definition of a pair of assets at a point in time will be defined later.
[24]When an over performing sector becomes under performing, leading to a sector-wise mean reversion.

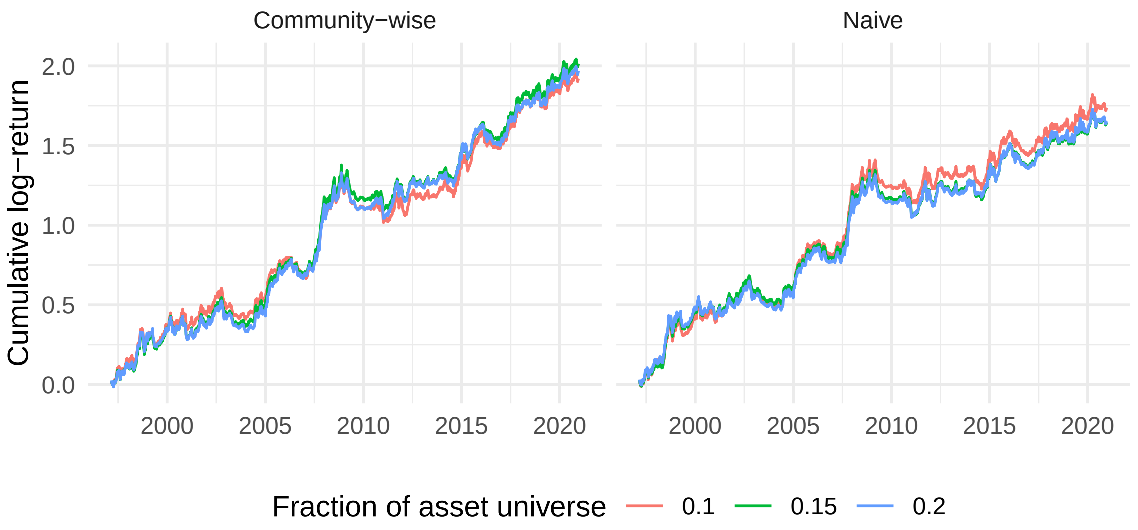


Figure 24: Cumulative log-return of the application to momentum strategies. Different fractions of the investment universe are used. Portfolios are standardized to 10% annualized volatility. The community-wise strategies outperform the naive ones on the full-sample simulation; but they performed similarly until 2015.

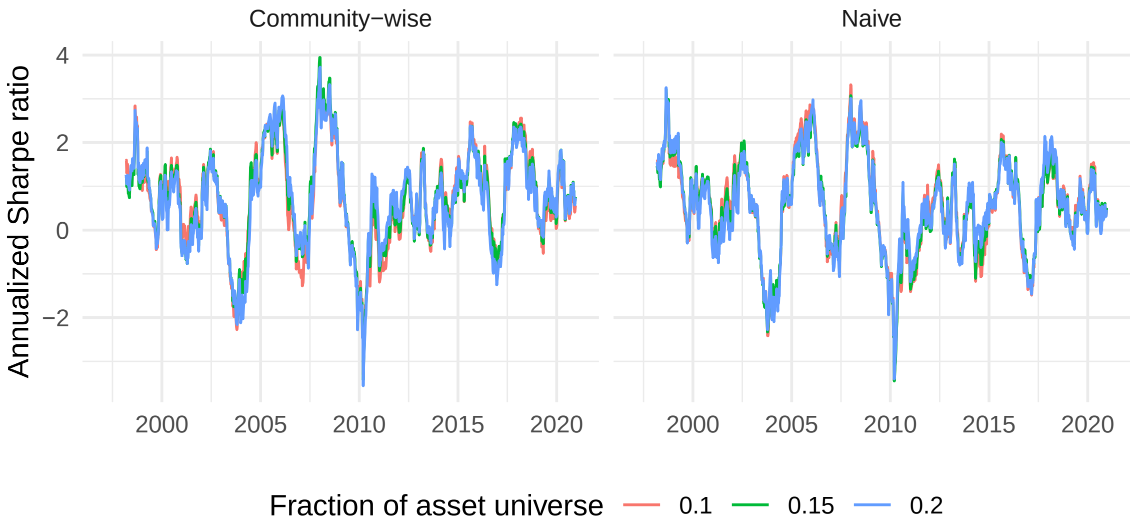


Figure 25: 50-weeks rolling annualized Sharpe ratio for momentum strategies. Only ratios for large allocation fractions are shown; where the use of communities significantly outperforms the naive momentum. The rolling Sharpe ratio for both settings look very much alike and are highly unstable. There is no evidence that one setting is better than the other from the Sharpe ratio point of view.

Table 11: Portfolio risk measures for the 15% momentum strategies. Overall, the community-wise strategy yielded a better annualized mean return over the full sample while taking on less variance of returns. Other metrics are similar between the two, except the worst drawdown which is largest for the community-wise strategy.

Risk measures	Community-wise	Naive
Annualized standard deviation (%)	10.741	10.904
Annualized downside standard deviation (%)	7.502	7.759
Annualized mean return (%)	8.913	7.285
Max weekly return (%)	5.277	5.516
Min weekly return (%)	-5.595	-5.916
Worst drawdown (%)	-19.979	-17.188

Finally, the 50-week drawdown is shown in Figure 26 for both the community-wise and the naive strategies. There is absolutely no clear gain from the community-wise strategy in terms of downside risk as measured by the size of drawdowns. Indeed, the community-wise strategy had the largest worst drawdown just prior to 2010.

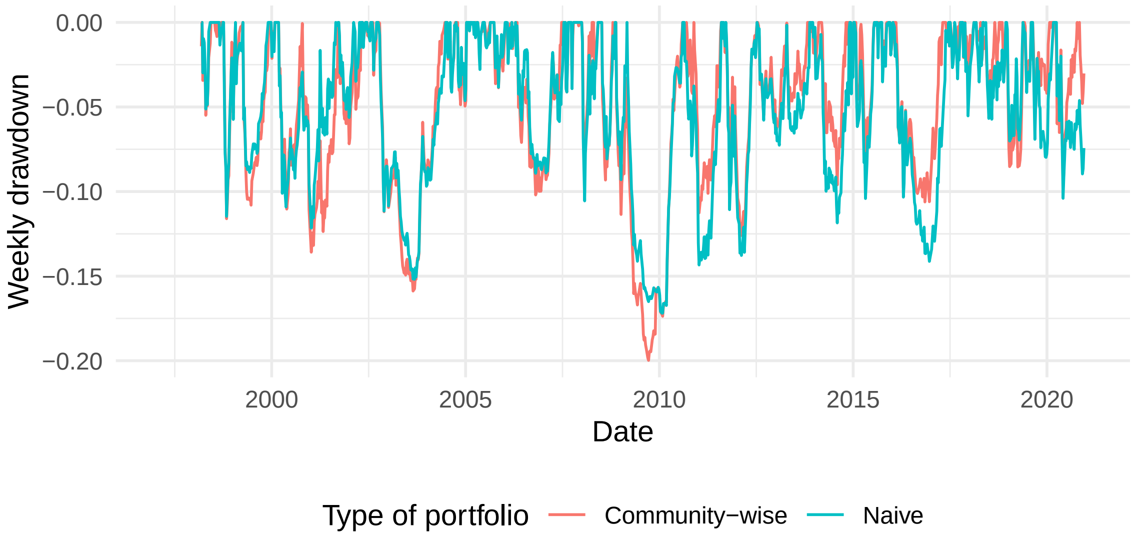


Figure 26: 50-weeks rolling weekly drawdown for momentum strategies. Both strategies show significant drawdown.

4.5.1.2 Markowitz The Markowitz optimization application that we propose should help enforce diversification not between sectors but between assets. We also expect it to bring more stable portfolios. Because Markowitz allocation will give a large weight to the least correlated asset, if this asset's correlation with the market increases even slightly the whole portfolio allocation will dramatically change between two adjacent periods. Hence, we aim at reducing concentration of assets by applying Markowitz between communities. There are two distinct approaches here : either apply Markowitz within each community and aggregate all sub-portfolios into one portfolio; or create as many synthetic assets as there are communities (using any weighting scheme) and apply Markowitz on the different portfolios.

Below we illustrate a strategy where on each rebalancing date, as many synthetic portfolios as there are communities are created. Communities' portfolios are created with a risk-parity approach, where all assets contribute equally to the volatility of the portfolio. Suppose a community with C assets $\{X_1, \cdots, X_C\}$ and a matrix of time series of past returns $\{(r_1)_{t-\tau}^t, \cdots, (r_C)_{t-\tau}^t\}$. Then the vector of weights is $\frac{1}{\sum_i \frac{1}{\sigma_i}}\{\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_C}\}$ where σ_i is the standard deviation of returns of asset X_i over the past time window $[t-\tau, t]$.

The obtained full-sample cumulative returns for 10% annualized volatility trading strategies are shown in Figure 27. The naive strategy consists in the usual Markowitz portfolio selection procedure applied to the whole investment universe.

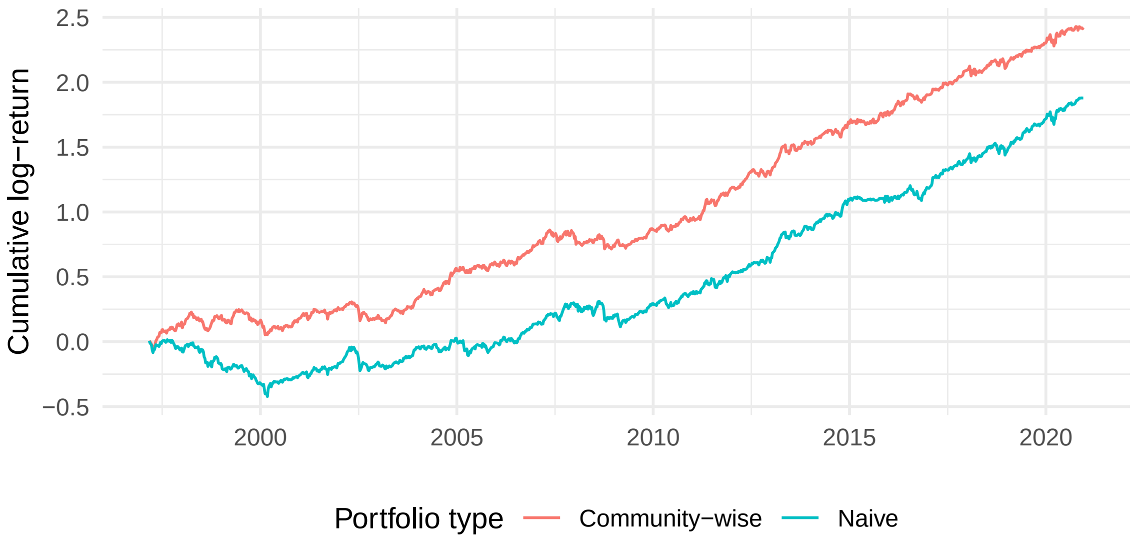


Figure 27: Cumulative log-return for minimum variance strategies. The community-wise strategy which applies Markowitz optimization to communities synthetic assets significantly outperforms the naive Markowitz optimization over the full-sample. Nevertheless, the two strategies performed similarly from 2000 onwards; and the community-wise strategy outperformed the naive one from initiation to 2000.

The 50-week rolling Sharpe ratios for both strategies are shown in Figure 28. There are some clear periods of time when the community-wise strategy outperformed the naive one on a risk-adjusted basis. From initiation to 2000, and between 2004 and 2006 did the community-wise strategy achieve better risk-adjusted performance.

Further full-sample risk measures are shown in Table 12. One can check that the targeted 10% annualized volatility was achieved. Once again, the community-wise strategy achieved a much higher annualized mean return over the full sample, in excess of the naive strategy by about 2.4%. It also achieved a better min and max weekly return as well as a much lower worst drawdown.

Finally, the 50-week drawdown is shown in Figure 29 for both the community-wise and the naive strategies. The community-wise strategy had very small drawdown at the beginning of the period when the naive

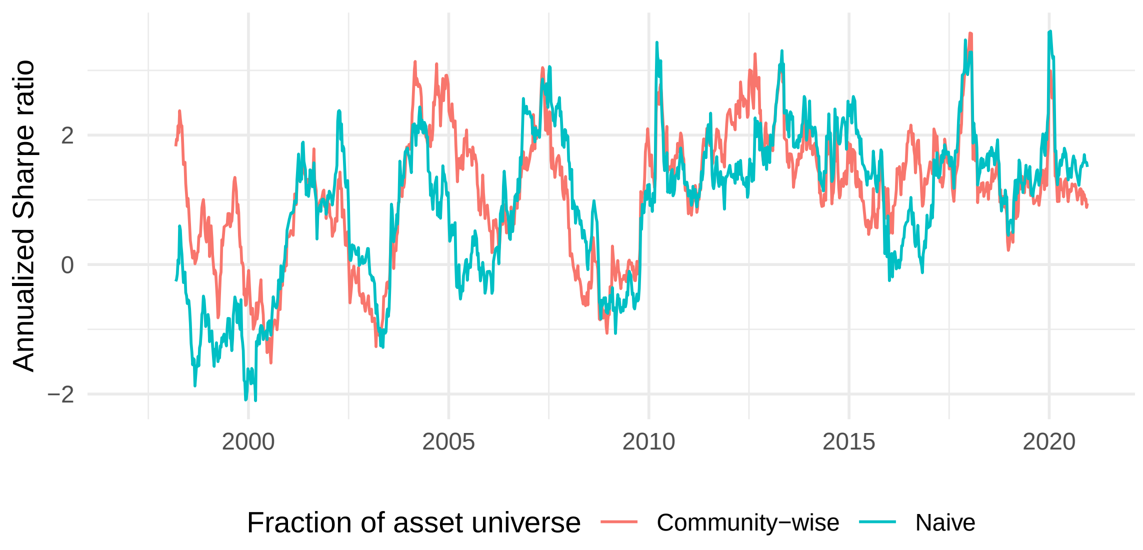


Figure 28: 50-weeks rolling Sharpe ratio. The dynamics of the Sharpe ratios are similar between the two strategies, but still there are some clear periods of time where the community-wise strategy outperformed the naive one.

Table 12: Portfolio risk measures for the minimum variance strategies. The community-wise strategy outperforms the naive strategy in all risk measures.

Risk measures	Community-wise	Naive
Annualized standard deviation (%)	10.317	10.486
Annualized downside standard deviation (%)	7.474	7.838
Annualized mean return (%)	10.724	8.320
Max weekly return (%)	6.539	5.414
Min weekly return (%)	-6.537	-7.281
Worst drawdown (%)	-17.703	-22.017

application of Markowitz optimization did have its largest drawdown. This period of time corresponds to the period when our proposed strategy indeed performed better than the naive strategy (see Figure 27).

Both methods can be tuned by the dimensionality ratio Q. One can use larger values of Q for more stable clusterings and portfolios. A smaller value of Q will add noise in the portfolio construction and increase the assets turnover but will better incorporate changes in correlation dynamics. Another parameter that can be tuned here is the length of the returns time series to use to find portfolio weights in the strategies. Again a longer history should provide more stable strategies, but maybe lag the market dynamics.
Overall, the use of our clustering procedure to vanilla trading strategies did contribute to attaining better risk metrics. Note that the present work is not dedicated to discovering profitable trading strategies but rather to implementing our clustering procedure within potential real life applications.

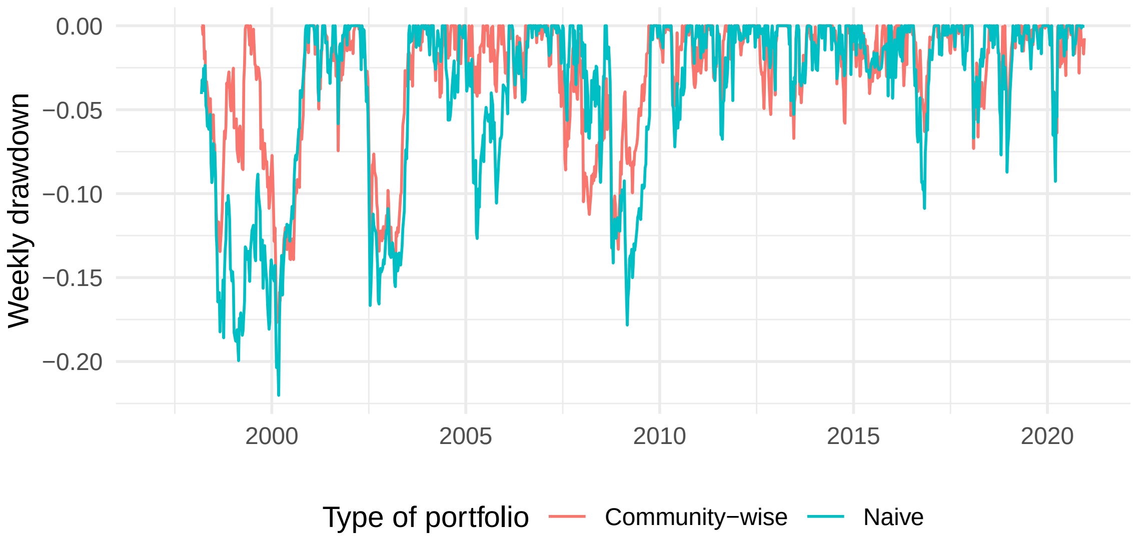


Figure 29: 50-weeks rolling weekly drawdown for minimum variance strategies. The community-wise strategy shows less significant drawdowns prio to 2000, which is the period over which it outperformed the naive setting.

4.5.2 Pairs-trading

Another application could be in a pairs-trading framework. Pairs-trading is a general trading strategy based on identifying pairs of similar assets. A suitable pair would be any two stocks which historically had a similar dynamics, very often in the same business segment. A suitable pair being thought of as sharing a common driver for its respective stock prices, a pairs-trader would wait for a divergence to happen (one stock goes up, the other goes down) and would bet on the future convergence of the two stock prices by buying the less performing stock and selling the most performing one. As such, it is considered a statistical arbitrage and convergence trading strategy.

Indeed, our procedure used in its first alternative records asset's correlation dynamics pair by pair, for all pairs of stocks; which is suitable data for pairs detection. We recall that our procedure in its first alternative does not rename clusters to link them between different clusterings during the bootstrap. Instead it only records how often two assets ended up together in the same community. The basic pairs-trading framework is detailed in the following box.

Pairs-trading framework using clustering co-occurrence data

1. Identify suitable pairs. A suitable pair could be any two stocks which are consistently clustered together; thus correlated in a sense.
2. Trade pairs based on a strategy. It could be selling the best performer and buying the least performing expecting mean-reversion.
3. Monitor pairs with our procedure and exit trades when a pair is dying; and entering new positions as pairs are created and persisting.

4.5.2.1 Identifying and monitoring pairs For each time stamp our procedure outputs the probability that the two assets end up together in the same community across all bootstraps. This probability is shown over time for a random pair of assets in Figure 30 as black dots. Those probabilities are not stable and are not consistently above or below an arbitrary threshold. To robustify the pair creation detection, the rolling 10-weeks mean co-occurrence probability for the pair is computed; the red line in Figure 30. This can be used as a signal as its is more stable than point estimates; but the price to pay is to lag the real dynamics. It could be coupled with a shorter term signal to identify sharp increases and decreases in co-occurrence probability. Those two signals together identify persisting pairs and also sudden pair creation or destruction; which can be used to automatically enter or exit trades.

The 10-weeks rolling signal has been used to identify pairs to trade in our pairs-trading framework. With the signal oscillating around an arbitrarily chosen 80% threshold, our algorithm will enter or exit a trade in the pair depending on the signal being above or below the threshold.

Another information that can be extracted from our first clustering alternative is the total fraction of pairs of assets identified as suitable pairs for pairs-trading, meaning that their 10-weeks rolling mean co-occurrence probability lies above the threshold. The fraction of suitable pairs[25] detected in the dataset over time is shown in Figure 31. Applications to market surveillance can be imagined.

The information on pairs can be used to design a pairs-trading strategy. But another application could be to use the largest possible number of assets within the S&P500 universe while minimizing the number of detected pairs within the sample to create a least correlated portfolio; a portfolio with the smallest possible number of detected strong pairs within its components. The developed trading strategy is described in the following box.

[25] Any pair of stocks with rolling 10-weeks co-occurrence probability above the threshold 80%.

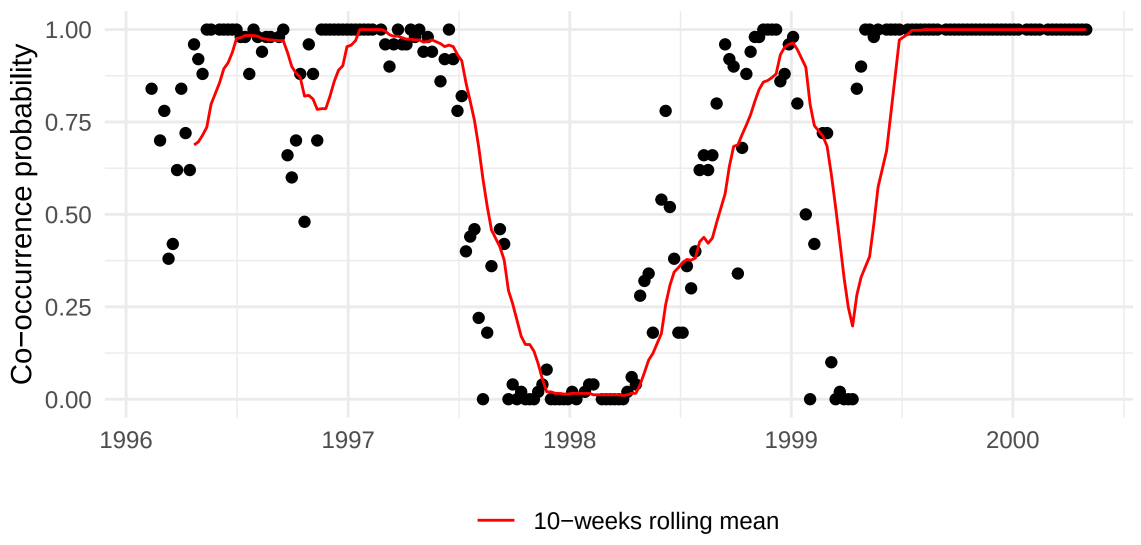


Figure 30: Co-occurrence probabilities over time (black dots) and smoothed out rolling 10-weeks mean probabilities (red line), for an arbitrary pair of stocks in our dataset.

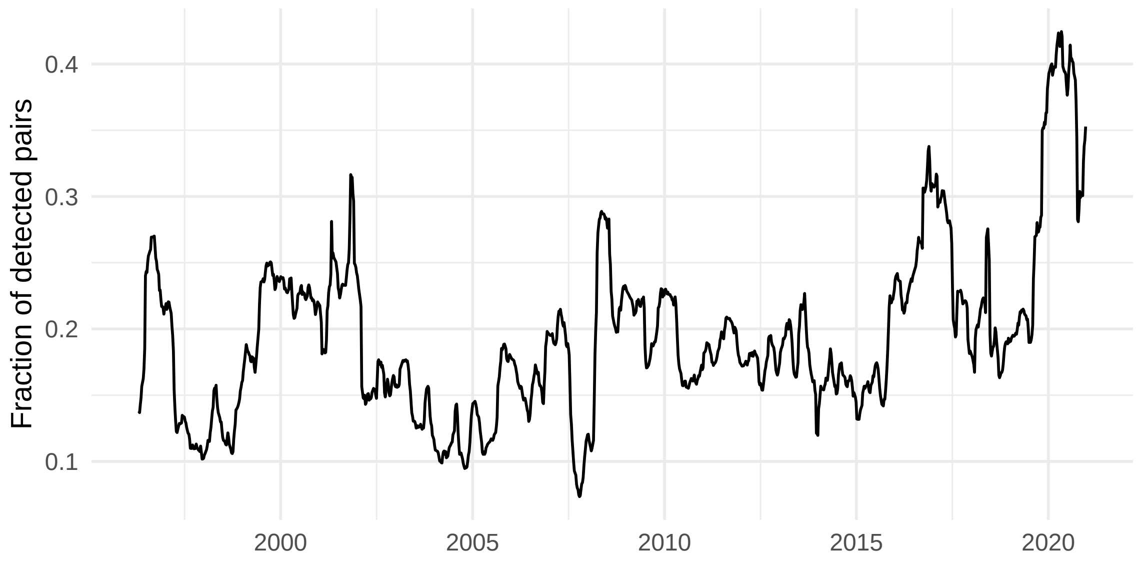


Figure 31: Fraction of pairs of assets detected as strong pairs with 10-weeks rolling mean co-occurrence probability above threshold 80%.

Construction of the least-pairs strategy using co-occurrence data

1. On rebalancing date, identify strong pairs.
2. Order assets by the number of strong pairs they participate to. Assets are thus ordered by their desirability for the portfolio which is constructed from assets which participate in the least possible number of strong pairs.
3. Select those assets which are in a number of strong pairs less than the average number of strong pairs per asset in the sample.
4. Weight those assets with $\frac{1}{\sigma}$: risk-parity; such that each asset contributes equally to the volatility of the portfolio.
5. Repeat for all portfolio rebalancing dates.

The obtained cumulative log-return is shown in Figure 32; the strategy is termed the least-pairs strategy. Note again that the portfolio is normalized to 10% annualized volatility for comparison to other trading strategies. Its risk-adjusted full-sample performance is less than that of the applications to momentum strategies and Markowitz portfolio selection.

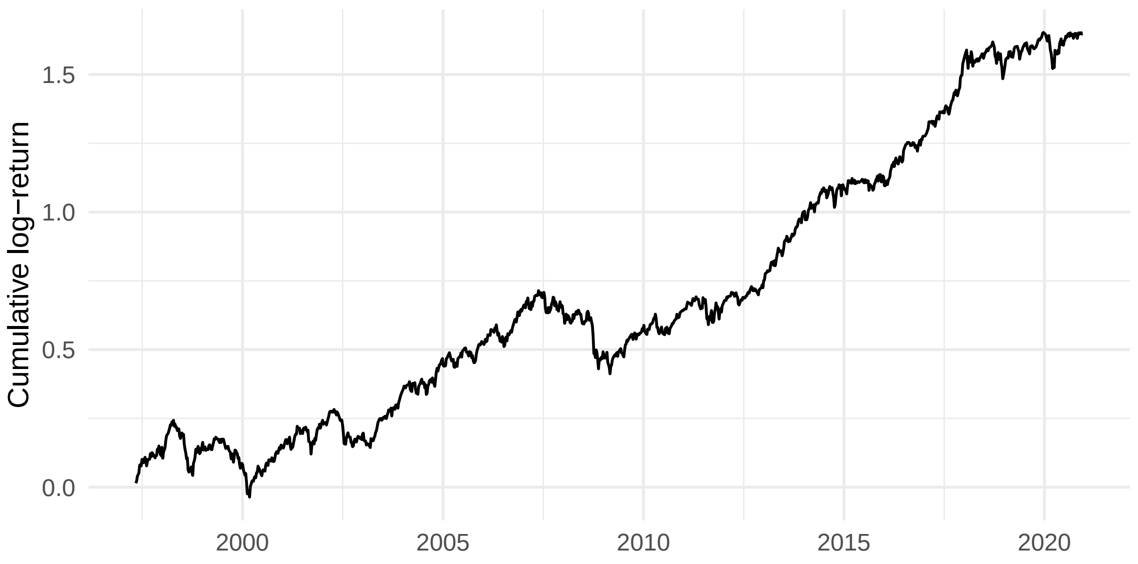


Figure 32: Cumulative log-return for the application to the least-pairs strategy.

Again, the 50-weeks rolling Sharpe ratio for this strategy is shown in Figure 33. It oscillates between -1 and 3. Further risk-metrics for this strategy are shown in Table 13. Overall, it under-performed both Markowitz and momentum applications. The 50-weeks rolling weekly drawdown is shown in Figure 34.

Table 13: Portfolio risk measures for the least-pairs strategy.

Risk measures	Least-Pairs
Annualized standard deviation (%)	10.373
Annualized downside standard deviation (%)	7.238
Annualized mean return (%)	7.359
Max weekly return (%)	6.297
Min weekly return (%)	-6.481
Worst drawdown (%)	-21.654

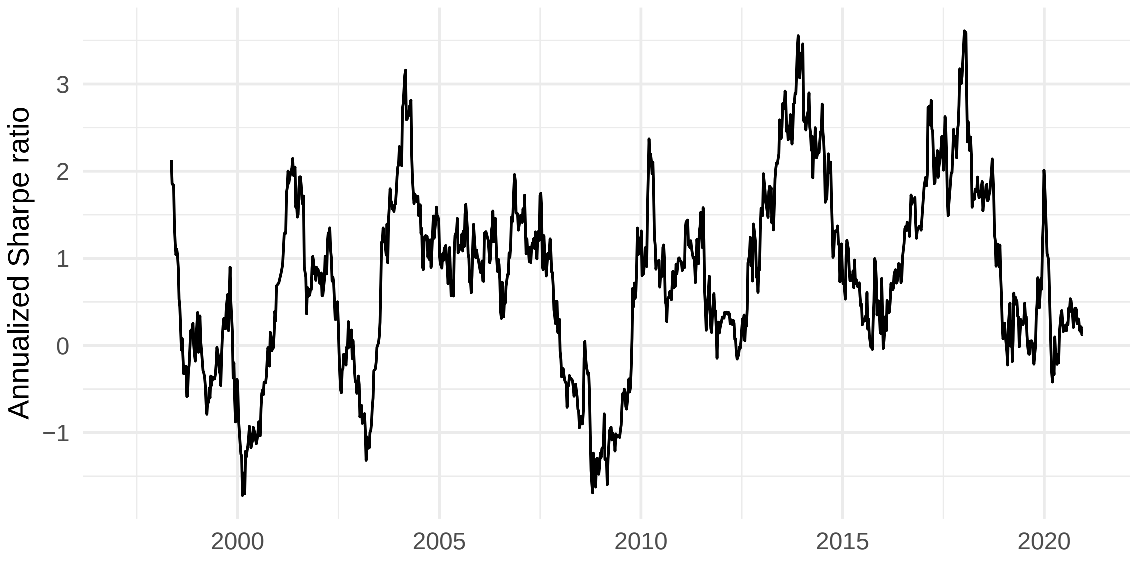


Figure 33: 50-weeks rolling Sharpe ratio for the least-pairs strategy.

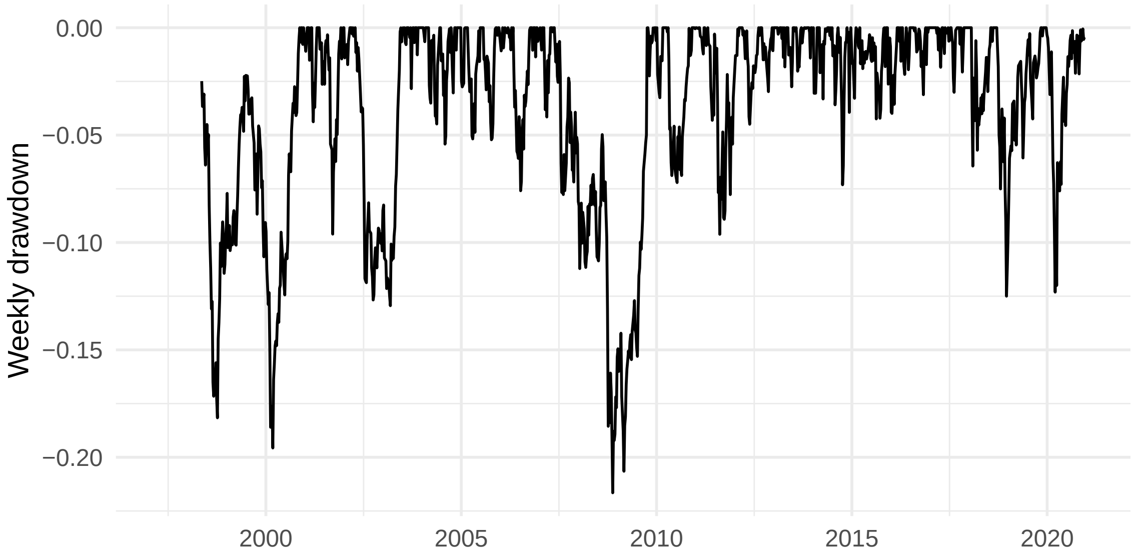


Figure 34: 50-weeks rolling weekly drawdown for the least-pairs strategy.

4.5.2.2 Trading strategy and bet sizing We developed a simple trading strategy on pairs. On each rebalancing date the list of strong pairs on that day is obtained by taking all pairs of assets such that their rolling 10-weeks mean co-occurrence probability is greater than the threshold (arbitrarily fixed to 80%).
For each strong pair, the mean standardized excess return over the previous year is computed. For a pair of assets $\{X_i, X_j\}$, the time series of excess return of asset X_i over asset X_j is $s_t^{ij} = (r_i - r_j)_{t-\tau-1}^{t-1}$. Standardized excess spread is then defined as $\frac{s_t^{ij}}{\sigma(s_t^{ij})}$, which is the average return of trading the pair (long asset X_i and short asset X_j) per unit of volatility.
Strong pairs are then ranked by decreasing value of absolute mean standardized excess spread. Absolute values are used because it was an arbitrary choice to go long the first asset and to go short the second asset. We will decide which side of the trade to take based on relative excess return. If the excess return is indeed large and positive we will take a position short-long betting on mean-reversion (the bet is that asset X_j will outperform asset X_i since the two are considered a strong pair and one is outperforming the other); while a short-long position will be entered if the spread is negative and large in absolute terms; again betting on the mean-reversion.

By computation-time consideration, on each date at most 600 strong pairs were considered and only the top 300 strong pairs in terms of absolute excess spread are kept for trading. 300 portfolios of long-short or short-long positions are created, one per strong pair. The time series of past returns of those portfolios are used to aggregate the 300 portfolios into 1 portfolio by risk-parity. This means that each strong pair returns are weighted by their inverse past standard deviation so each pair contributes equally to the aggregate portfolio's volatility. Once again the portfolio is leveraged to attain a 10% annualized volatility over the full sample. The portfolio's cumulative return is shown in 35. The portfolio performed well in the first half of the period but had some extreme losses in 2008-2009. This is illustrated in the portfolio's 50-week rolling Sharpe ratio in Figure 36 which peaked above 3 but reached -4 during the year 2008. This extreme loss is also identified in the rolling 50-weeks weekly drawdown in Figure 37. The worst drawdown was about -35%. Further risk-measures for the strategy are shown in Table 14. This strategy has definitely the worst risk-adjusted performance of all the strategies benchmarked in this work.

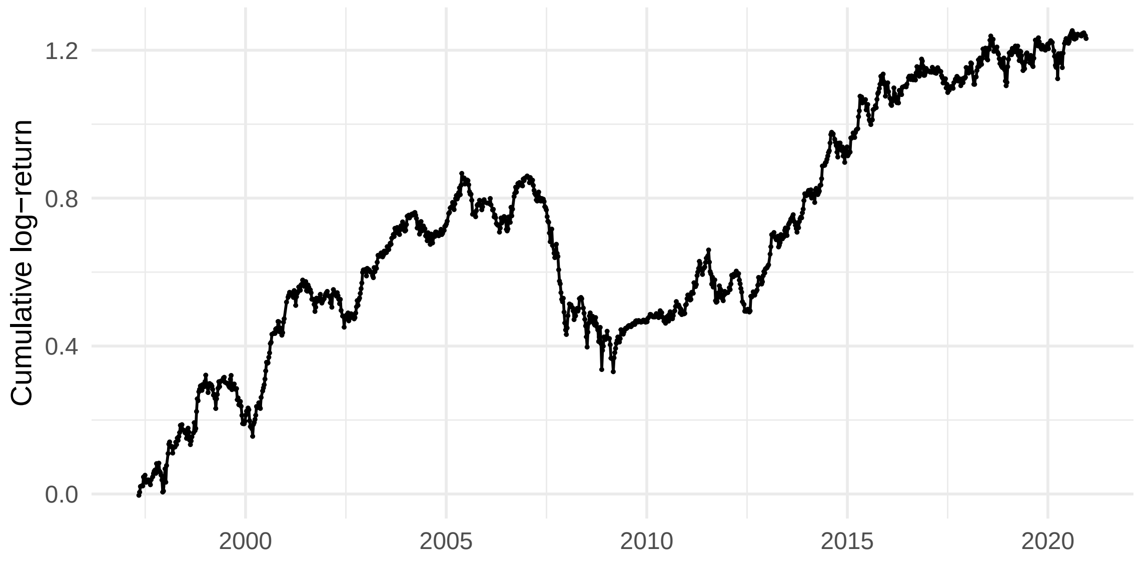


Figure 35: Cumulative log-return for the application to the pairs-trading strategy. The strategy performs poorly over 2008-2009.

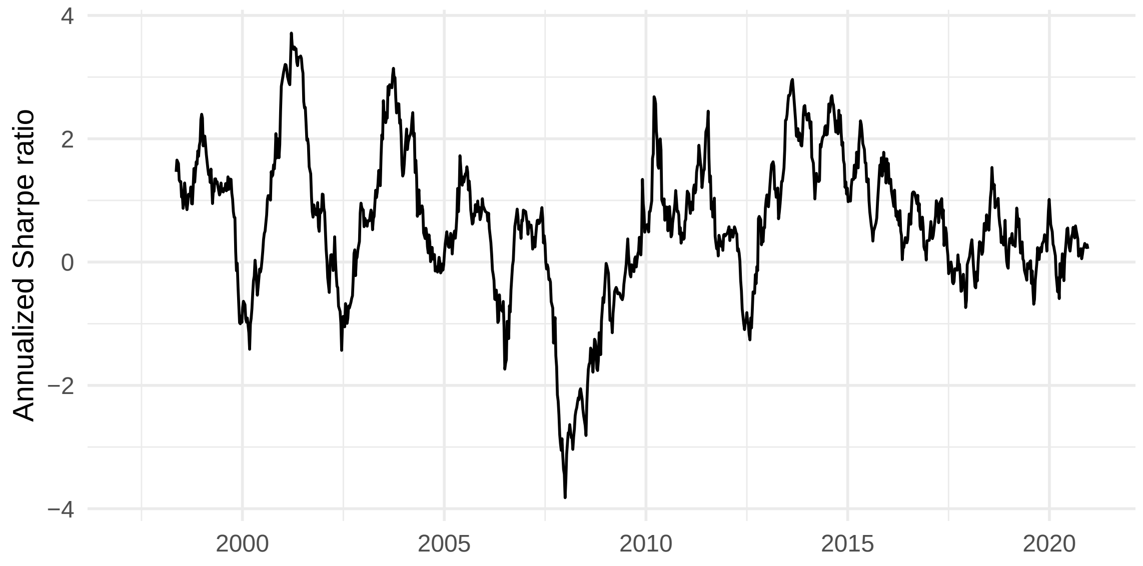


Figure 36: 50-weeks rolling Sharpe ratio for the pairs-trading strategy.

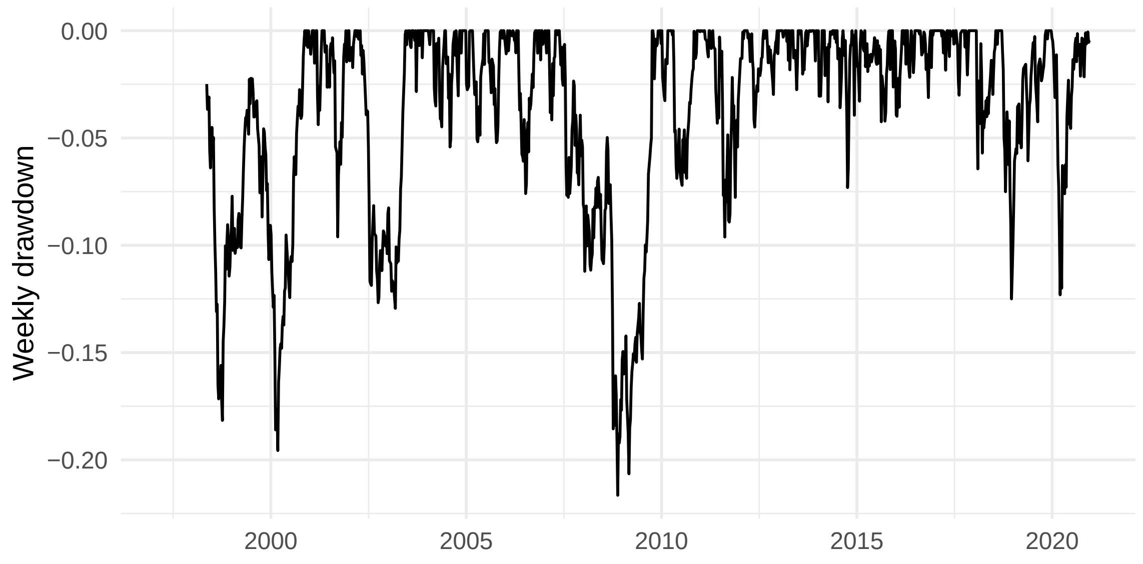


Figure 37: 50-weeks rolling weekly drawdown for the pairs-trading strategy.

Table 14: Portfolio risk measures for the pairs-trading strategy.

Risk measures	Pairs-trading
Annualized standard deviation (%)	10.552
Annualized downside standard deviation (%)	6.815
Annualized mean return (%)	5.498
Max weekly return (%)	6.759
Min weekly return (%)	-7.014
Worst drawdown (%)	-34.860

5 Conclusion

In this work we developed a fully unsupervised, rolling framework for the clustering of assets from a specific asset universe.

The unsupervised clustering algorithm we implemented is based on the redefinition of the celebrated Louvain method (Blondel et al. [2008]), a redefinition needed for consistency with the use of empirical correlation matrix (MacMahon and Garlaschelli [2015]) instead of the usual adjacency matrix from network theory. The rolling framework we propose allows for real-life applications of the clustering procedure; and we benchmarked a few of these applications. The applications that we benchmarked do not constitute the heart of this work, but rather showcase the wide range of possibilities offered by our rolling framework for unsupervised clustering; and as such they should serve as motivation for further applications.

From the clustering algorithm proposed by MacMahon and Garlaschelli [2015], and the mathematical toolbox borrowed from Random Matrix Theory which supports this algorithm; we introduced the notion of bootstrapped clustering. A bootstrapped clustering is defined as a repetition of our building block clustering algorithm to obtain a sample of clusterings at each time stamp. Our clustering procedure introduces randomization during the clustering process, always pushing the solutions out of equilibrium and allowing for the exploration of a broader volume of the clusterings space. A bootstrapped clustering is a collection of unique clusterings. The enforced randomness is then leveraged to introduce the notions of probabilistic clustering, and of co-occurrence matrix. Even though probabilistic clustering has some appealing properties, the central result of this work is the concept of co-occurrence matrix from bootstrapped clusterings.

The co-occurrence matrix is simply explained as the square matrix whose entries represent the frequency at which two stocks end up in the same community for different instances of the same bootstrap. Two stocks with exact same dynamics would have co-occurrence 1, meaning that they end up in the same community 100% of the time; while two completely uncorrelated stock would have a zero co-occurrence frequency. This simple, yet powerful, statistics obtained by aggregating bootstrapped clusterings can be interpreted as a weighted adjacency matrix; with coefficients indicating the strength of the link between two nodes. Adjacency matrices being the one element that we did not have initially to apply the vanilla Louvain algorithm, we managed to sample it from the redefined Louvain algorithm for correlation matrices. Finally, the co-occurrence matrix can be used as an adjacency matrix representing the links between nodes (stocks) in the market at a time t, and inputted to the vanilla Louvain algorithm, closing the loop.

Finally, some final robustification and trading applications were proposed, building on the introduced concept of co-occurrence matrix.

6 Appendix

6.1 Appendix A : Random Matrix Theory

We discuss a technique based on RMT to clean noise from correlation matrices. This result was introduced by Laloux et al. [2000] to understand the statistical structure of empirical correlation matrices in the context of financial returns time series. Empirical correlation matrices have been used extensively in finance over the past decades and are at the heart of Markowitz [1952]'s optimal portfolio selection theory.

It turns out that the reliable estimate of an empirical correlation matrix is subject to a trade off between stock universe's size and informativity of the statistics. If one considers a set of N assets, the correlation matrix contains $N(N-1)/2 \approx N^2$ non-trivial entries to estimate from N time series of T observations each, for a total of $N \times T$ observations. If $T \leq N$ then the estimate of the correlation matrix will be noisy ($NT \leq N^2$). A sound rule of thumb is to select $T \geq N$. Suppose $N = 500$ as in the S&P 500, then $T \geq 500$ with 252 trading days per year represents at least two years worth of data, at daily frequency. As the universe's size increases, the time window increases as well and dynamics is smoothed out.

We denote the dimensionality ratio $Q = \frac{T}{N}$. For a constant number of assets N the number of observations per series T should be as large as possible to have an accurate estimate, but not too large and smooth the dynamics of stocks' returns out. A solution is to increase the sampling frequency of stock returns to inflate T while keeping the same time window; or to reduce the stock universe's size N.

Another solution is to use tools from RMT and clean noise out from empirical correlation matrices. The general idea of using RMT to de-noise a correlation matrix is to split the correlation matrix itself into a noise term which accounts for the random part of the empirical correlation matrix and an informative term which contains actual structural information.

In the following we recall the derivation of the most central result from RMT used in this work : the Marcenko-Pastur distribution. We first recall results from portfolio theory which motivate the cleaning of correlation matrices and then we recall the mathematical derivation of the Marcenko-Pastur distribution. We follow the paper by Potters et al. [2005].

6.1.1 Portfolio theory : basics results

Suppose a portfolio of N assets with weight ω_i on the i-th asset. Call C_{ij} the true correlation matrix of assets' returns and σ_i^2 the true variance of asset i returns. Then, the variance of the returns of the portfolio is

$$R^2 = \Sigma_{ij}\omega_i\sigma_i C_{ij}\sigma_j\omega_j$$

Call g_i the expected gain of asset i; then the expected gain of the portfolio is $G = \Sigma_i \omega_i g_i$.

C_{ij} is never observed and needs to be estimated. The correlation matrix has of the order of $N^2/2$ coefficients to be estimated from $N \times T$ data points; where T is the length of each time series of returns and N is the number of such time series, i.e. the number of assets in the portfolio. The dimensionality ratio $Q = \frac{T}{N}$ is replaced by its inverse in the paper, and we will stick to this notation : $q = \frac{N}{T}$. An accurate estimate of C_{ij} the correlation matrix will require $q \ll 1$, or equivalently $T \gg N$. Let us call r_t^i the daily return of stock i at time t. Then the empirical variance of each stock is given by

$$\sigma_i = \frac{1}{T}\Sigma_t^T (r_t^i)^2$$

where the daily mean return is neglected for simplicity as it is small compared to daily fluctuations. The empirical correlation matrix over the same period is obtained by

$$E_{ij} = \frac{1}{T} \Sigma_t^T x_t^i x_t^j$$

where $x_t^i = r_t^i / \sigma_i$ are the normalized returns [26].

If $T < N$, then the empirical correlation matrix $\mathbf{E}$ will have rank T and will have $N - T$ zero eigenvalues. The variance of the portfolio is estimated by

$$\langle R_E^2 \rangle = \frac{1}{T} \Sigma_{ijt} \omega_i \sigma_i \langle x_t^i x_t^j \rangle \sigma_j \omega_j \approx \Sigma_{ij} \omega_i \sigma_i C_{ij} \sigma_j \omega_j$$

The portfolio with minimum variance for a given return level G is obtained from a Markowitz optimization scheme and is given by

$$\omega_i \sigma_i = G \frac{\Sigma_j C_{ij}^{-1} g_j / \sigma_j}{\Sigma_{ij} g_i / \sigma_i C_{ij}^{-1} g_j / \sigma_j}$$

which rewrites in matrix notation

$$\mathrm{w}_C = G \frac{\mathbf{C}^{-1} \mathbf{g}}{\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g}}$$

They show that the out-of-sample risk of an optimized portfolio is larger than its in-sample risk; which is an underestimate of the true minimal risk. This is explained by the optimization adapting to the particular realization of the noise which is unstable in time.

The conclusion is that out-of-sample risk[27] is underestimated and is only accurately estimated in the limit $q \to 0$ where the measurement noise disappears.

6.1.2 Matrix Cleaning and RMT

The question is now how to clean the empirical correlation matrix to avoid such downward biases in the estimation of future risk.

The Markowitz solution rewrites in terms of the eigenvalues λ_k and eigenvectors V_i^k of the correlation matrix. By writing $\mathbf{C} = \mathbf{V}^T \, \mathbf{V}$, then $\omega_i \approx \Sigma_j C_{ij}^{-1} g_j$; and

$$\begin{aligned} C_{ij} &= \Sigma_k (\mathbf{V}^T \ )_{ik} \mathbf{V}_{kj} \\ &= \Sigma_{kl} \mathbf{V}_{il}^T \ _{lk} \mathbf{V}_{kj} \\ &= \Sigma_k \mathbf{V}_{ik}^T \lambda_k \mathbf{V}_k j \\ &= \Sigma_k \lambda_k V_i^k V_j^k \end{aligned} \tag{13}$$

Then, $C_{ij}^{-1} = \Sigma_k \lambda_k^{-1} V_i^k V_j^k$; and finally

$$\omega_i \approx \Sigma_{jk} \lambda_k^{-1} V_i^k V_j^k g_j = g_i + \Sigma_{jk} (\lambda_k^{-1} - 1) V_i^k V_j^k g_j$$

This result illustrates a shortfall of the Markowitz optimized portfolio. Obviously, the weight ω_i is proportional to the expected risk-adjusted return g_i; this is the first term of the sum. The correction term suppresses allocation to eigenvectors (portfolios) with $\lambda > 1$, and enhance the allocation to eigenvectors with

[26] Once again the mean daily return is neglected.
[27] i.e. future risk

$\lambda < 1$. Hence, a very large portion of the portfolio could be allocated to small eigenvalued eigenvectors; which may be entirely dominated by unstable measurement noise.

A method proposed to clean eigenvalues is to replace all eigenvalues below a certain threshold $k^* $with a unique value; and to keep all the high eigenvalues which are above the threshold and which are believed to contain economical information. The unique value to assign to small eigenvalues is chosen so as to conserve the trace of the correlation matrix. The problem of finding the appropriate eigenvalue threshold λ^* remains.

The idea proposed in papers by Laloux et al. [2000] and Laloux et al. [1999] is to use RMT to determine the threshold λ^* which separates the random part of the eigenvalue distribution from the insightful part. This theoretical threshold for the spectrum of a random correlation matrix is known under certain hypothesis thanks to the work by Marcenko and Pastur Marčenko and Pastur [1967].

We will recall the derivation of the major result by Marcenko and Pastur known as the Marcenko-Pastur distribution which provides a theoretical maximum value for the eigenvalue spectrum of a random correlation matrix. We follow the derivation provided by the authors Potters et al. [2005].

Let us consider an empirical correlation matrix $\mathbf{E}$ obtained from N time series (N assets) of T observations. Furthermore, suppose that N and T are very large with $q = \frac{N}{T}$ finite; and that the true correlations are given by

$$\langle x_t^i x_{t'}^j \rangle = C_{ij}\delta_{tt'}$$

. The resolvent is introduced to study the eigenvalue spectrum of $\mathbf{E}$:

$$G(z) = \frac{1}{N}\,\mathrm{Tr}\left[(z\mathbf{I} - \mathbf{E})^{-1}\right]$$

and the eigenvalue density is given by:

$$\rho(\lambda) = \lim_{\epsilon\to 0}\frac{1}{\pi}\Im(G(\lambda - \mathrm{i}\epsilon))$$

The simplest case which is considered throughout this work is $\mathbf{C} = \mathbf{I}$, the null-model where all assets are uncorrelated. Then, $\mathbf{E}$ is a sum of rotationally invariant matrices $\delta E_{ij}^t = (x_t^i x_t^j)/T$; such that $\mathbf{E} = \Sigma_{t'=t-T}^{t}\, \mathbf{E}^{t'}$. The matrix $\delta E_{ij}^t = (x_t^i x_t^j)/T$ has one eigenvalue equal to q and $N-1$ zero eigenvalues; such that its trace is $\mathrm{Tr}(\delta E_{ij}) = \Sigma_i \lambda_i = q$.

Then, $z\mathbf{I} -\ \mathbf{E}$ has eigenvalues $z - q$ once and z $N-1$ times. Finally, $(z\mathbf{I} -\ \mathbf{E})^{-1}$ has eigenvalues $\frac{1}{z-q}$ once and $\frac{1}{z}$ $N-1$ times. Then, $\delta G_t(z) = \frac{1}{N}\,\mathrm{Tr}[(z\mathbf{I} -\ \mathbf{E})^{-1}] = \frac{1}{N}\left(\frac{1}{z-q} + \frac{N-1}{z}\right)$.

The trick is then to consider the Blue function which is the inverse function of the resolvent G, i.e. such that $B(G(z)) = z$; and to use the R-transform of G given by $R(x) = B(x) - 1/x$.

Inverting $\delta G_t(z)$ to first order in $1/N$ gives:

$$\delta B_t(x) = \frac{1}{x} + \frac{q}{N(1-qx)}$$

The R-transform of $\delta G_t(z)$ is then $\frac{q}{N(1-qx)}$. From Feier [2012], as E_{ij} is the free convolution of the random matrices δE_{ij}, its R-transform is

$$R_E = TR_{\delta E} = \frac{qT}{N(1-qx)} = \frac{1}{1-qx}$$

and finally:

$$B_E(x) = \frac{1}{x} + \frac{1}{1-qx}$$

By inverting $B_E(x)$ one finds:

$$G_E(z) = \frac{(z+q-1) - \sqrt{(z+q-1)^2 - 4zq}}{2zq}$$

which is the Marcenko-Pastur distribution:

$$\rho(\lambda) = \frac{\sqrt{4\lambda q - (\lambda + q - 1)^2}}{2\pi\lambda q}$$

This theoretical distribution depends on the value of the parameter q and is defined for $\lambda \in [\lambda_-, \lambda_+]$; where $\lambda_\pm = (1 \pm \sqrt{q})^2$. Any eigenvalue in the empirical correlation matrix larger than λ_+ will be considered as resulting from a divergence of the correlation matrix from the null hypothesis where $\mathbf{C} = \mathbf{I}$. The solution provides a consistent procedure to clean eigenvalues and separate the random bulk from the economically meaningful set of eigenvalues.

From now on we will use the notation $Q = \frac{1}{q} = \frac{T}{N}$. In Figure 38 one can see different such Marcenko-Pastur's distributions for increasing values of Q (i.e. decreasing values of q) which correspond to increasing values of T. As Q increases the distribution of eigenvalues tends towards a delta function centered on $\lambda = 1$ as measurement noise vanishes and the empirical correlation matrix tends towards the true correlation matrix $\mathbf{C} = \mathbf{I}$; which has all eigenvalues equal to 1.

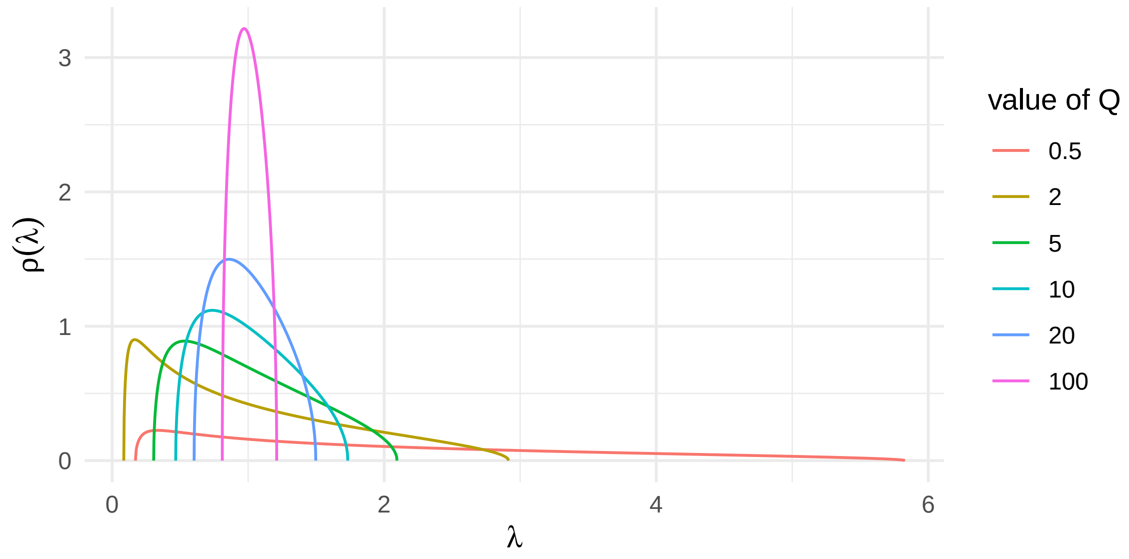


Figure 38: Marcenko-Pastur's theoretical distribution for increasing values of Q. The. distribution converges to a Delta Dirac centerd on 1 as Q increases.

6.2 Appendix B : Modified Louvain method

The modified Louvain method is the one proposed by MacMahon and Garlaschelli [2015] which is based on their redefined null-model. In particular, they show that the two requirements of the Louvain procedure are satisfied by the new definition. The two requirements are that:

- the network can be properly renormalized. This means that the aggregation of the nodes belonging to same communities into hypernodes should remain consistent with the meaning of the modularity.
- the change in modularity obtained from moving one node from its community to another should be efficient to compute.

Here we show that those two properties are satisfied using the new null-model

$$Q(\vec{\sigma}) \equiv \frac{1}{C_{norm}} \sum_{i,j} C_{ij}^{(g)} \delta\left(\sigma_i, \sigma_j\right)$$

where $C_{ij}^{(g)} = C_{ij} - C_{ij}^{(r)} - C_{ij}^{(m)}$; also called the filtered correlation matrix. We follow the derivations and the notations of the authors.

6.2.1 Consistent renormalization

Let us consider two standardized time series X_i and X_j; such that $C_{ij} = \text{Cov}[X_i, X_j]$. Then, the renormalized interaction between two hypernodes[28] A and B is given by

$$\begin{aligned}\sum_{i\in A}\sum_{j\in B} C_{ij} &= \sum_{i\in A}\sum_{j\in B} \text{Cov}\left[X_i, X_j\right] \\ &= \text{Cov}\left[\sum_{i\in A} X_i, \sum_{j\in B} X_j\right]\end{aligned}$$

where the bilinearity of the covariance function was used. After introducing the renormalized time series of community A, $\tilde{X}_A = \sum_{i\in A} X_i$; the renormalized interaction rewrites

$$\tilde{C}_{AB} \equiv \sum_{i\in A}\sum_{j\in B} C_{ij} = \text{Cov}\left[\tilde{X}_A, \tilde{X}_B\right]$$

which shows that renormalized interactions are consistently defined in terms of covariances.

The next step is to show whether the modularity function remains consistent with the null-model at the level of renormalized nodes.

We can write the following for $\langle \tilde{C}_{AB} \rangle$

$$\begin{aligned}\langle \tilde{C}_{AB} \rangle &= \tilde{C}_{AB}^{(r)} + \tilde{C}_{AB}^{(m)} \\ &= \sum_{i\in A}\sum_{j\in B} (C_{ij}^{(r)} + C_{ij}^{(m)}) \\ &= \sum_{i\in A}\sum_{j\in B} \langle C_{ij} \rangle\end{aligned}$$

which means that the filtered renormalized correlation matrix writes

[28]Hypernodes are nodes from a higher level. They can be the initial nodes (assets) or a community of nodes after renormalization.

$$\tilde{C}_{AB}^{(g)} = \sum_{i \in A} \sum_{j \in B} C_{ij}^{(g)}$$

The renormalized modularity thus rewrites

$$\begin{aligned}
\tilde{Q}(\vec{\sigma}) &\equiv \frac{1}{\tilde{C}_{norm}} \sum_{A,B} \tilde{C}_{AB}^{(g)} \delta\left(\tilde{\sigma}_A, \tilde{\sigma}_B\right) \\
&= \frac{1}{\tilde{C}_{norm}} \sum_{A,B} \sum_{i \in A} \sum_{j \in B} C_{ij}^{(g)} \delta\left(\tilde{\sigma}_A, \tilde{\sigma}_B\right) \\
&= \frac{1}{\tilde{C}_{norm}} \sum_{i,j} C_{ij}^{(g)} \delta\left(\sigma_i, \sigma_j\right) \\
&= Q(\vec{\sigma})
\end{aligned} \tag{14}$$

where $\tilde{C}_{norm} = \sum_{A,B} \tilde{C}_{AB} = \sum_{A,B} \sum_{i \in A} \sum_{j \in B} C_{ij} = \sum_{i,j} C_{ij} = C_{norm}$. Equation (14) shows that the modularity is invariant under renormalization of the network. This result shows that the redefined Louvain algorithm with the new null-model is consistent with the renormalization of the network.

6.2.2 Change in modularity : efficient computation

Another crucial attribute of the vanilla Louvain algorithm is the fact that the gain in modularity associated with moving one node from its community to its neighbors' communities can be efficiently calculated. We need to show that there exists a simple formula with the modified algorithm as well.

Let us start from the definition of the modularity at any aggregation level : $\tilde{Q}(\vec{\sigma})$. The modularity gain obtained by moving hypernode I to the community J is $\Delta\tilde{Q}^{(I \to J)}$. It is calculated as the difference between $\tilde{Q}(\vec{\tilde{\sigma}}')$ for a partition $\vec{\tilde{\sigma}}'$ where I is in community J and $\tilde{Q}(\vec{\tilde{\sigma}}'')$ for a partition $\vec{\tilde{\sigma}}''$ where I is in its own community. This reads respectively $\tilde{\sigma_I}' = J$ and $\tilde{\sigma_I}'' \neq J$.

Since $\tilde{\sigma_A}' = \tilde{\sigma_A}''$ for all $A \neq I$ (only I changes community between $\vec{\tilde{\sigma}}'$ and $\vec{\tilde{\sigma}}''$) and $\delta(\tilde{\sigma_I}'', \tilde{\sigma_A}'') = 0$ for all $A \neq I$ the difference writes

$$\begin{aligned}
\Delta\tilde{Q}^{(I \to J)} &= \tilde{Q}\left(\vec{\sigma}'\right) - \tilde{Q}\left(\vec{\sigma}''\right) \\
&= \frac{1}{C_{norm}} \sum_{A,B} \tilde{C}_{AB}^{(g)} \left[\delta\left(\tilde{\sigma}'_A, \tilde{\sigma}'_B\right) - \delta\left(\tilde{\sigma}''_A \tilde{\sigma}''_B\right)\right] \\
&= \frac{1}{C_{norm}} \sum_{A} \tilde{C}_{IA}^{(g)} \left[\delta\left(\tilde{\sigma}'_I, \tilde{\sigma}'_A\right) - \delta\left(\tilde{\sigma}''_I, \tilde{\sigma}''_A\right)\right] \\
&= \frac{1}{C_{norm}} \sum_{A \in J} \tilde{C}_{IA}^{(g)} \\
&= \frac{\tilde{C}_{IJ}^{(g)}}{C_{norm}}
\end{aligned}$$

which shows that the change in modularity is proportional to the renormalized interaction between hypernode I and community J; which is the sum of the filtered correlations between time series within I and those within J.

The change in modularity obtained from moving a hypernode I from a community J' to its own isolated community is $-\Delta\tilde{Q}^{(I \to J')}$.

Finally, the change in modularity when moving I from J' to another community J is given by equation (15).

$$-\Delta\tilde{Q}^{(I\rightarrow J')}+\Delta\tilde{Q}^{(I\rightarrow J)}=\frac{\tilde{C}^{(g)}_{IJ}-\tilde{C}^{(g)}_{IJ'}}{C_{norm}} \tag{15}$$

where $\tilde{C}^{(g)}_{IJ}$ is simply the sum of the filtered correlations between time series in I and in J; which is the sum of all the elements in a submatrix of the filtered correlation matrix.

This demonstrates that the reformulation of Louvain proposed by MacMahon and Garlaschelli [2015] satisfies the requirements of the vanilla Louvain algorithm at all aggregation levels.

6.3 Appendix C : Review of the classical measures of correlation

6.3.1 Pearson correlation coefficient

The Pearson correlation, also know as the product-moment correlation is the most elementary measure of linear correlation between two random variables. The formula for Pearson correlation between variables X and Y is given in equation (16).

$$\begin{aligned}\rho_{X,Y} &= \frac{cov(X,Y)}{\sigma_X \sigma_Y} \\ &= \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{(\mathbb{E}[X^2] - \mathbb{E}^2[X])(\mathbb{E}[Y^2] - \mathbb{E}^2[Y])}}\end{aligned} \tag{16}$$

As can be seen from the definition of Pearson correlation coefficient, it is well defined only for random variables X, Y with finite first and second moments. This makes this statistic not robust to the presence of outliers in the data set which can significantly impact the mean and standard deviation of the considered samples. Some distributions like the Cauchy distribution have undefined variance; hence product-moment correlation cannot be defined if X or Y follows such a distribution. These considerations are of great importance when working with data suspected to follow some heavy-tailed distributions like stock returns. Pearson linear correlation coefficient has the great disadvantage of assuming linear dependence between the two variables; hence not capturing non linear relationships. On the other hand, Pearson estimator has the advantage of being extremely fast to compute. Indeed, we have that

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = cov(\tilde{X}, \tilde{Y}) \tag{17}$$

where $\tilde{X}$, $\tilde{Y}$ are the standardized random variables. If $\mathbf{M}$ is the $T \times N$ matrix of standardized returns (or log returns), then $\mathbf{C} = \frac{1}{T}\mathbf{M}^T\mathbf{M}$ is the corresponding correlation matrix.

6.3.2 Rank correlation coefficient

Alternatives to linear correlation estimators are rank correlation estimators. Rank correlation measures ordinal association between two variables, where we are interested not in the data itself but in the ranking of the observations. With rank correlation estimators we abstract from the actual values random variables take, but we focus on the ordering of the observations. One can intuitively feel that these measures are more appropriate for heavy-tailed distributions; since an arbitrarily large realization of the random variable is not going to contaminate the estimator, but will rather be mapped to N where N is the number of observations used. In the following we are going to introduce 3 rank correlation estimators, namely Spearman, Kendall and Gaussian rank correlation coefficient.

6.3.2.1 Spearman correlation coefficient Spearman correlation coefficient is defined as the Pearson correlation of the rank variables, described in equation (18)

$$\rho^S_{X,Y} = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}} \tag{18}$$

Where ρ^S denotes Spearman correlation and ρ is the usual Pearson correlation. Spearman correlation is nonparametric in the sense that absolute perfect Spearman correlation is achieved when X and Y are related by any monotonic function; while absolute perfect Pearson correlation is only achieved when X and Y are related by a linear function.

6.3.2.2 Kendall correlation coefficient

Kendall correlation coefficient is another measure of rank correlation like Spearman's. Kendall's τ is defined in terms of concordant and discordant pairs of observations. Suppose $(X, Y) = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is a sequence of observations; where values of $\{x_i\}, \{y_j\}$ are unique (we neglect ties for simplicity). Then a pair (x_i, y_i), (x_j, y_j) is said to be concordant if either $x_i \geq x_j$ and $y_i \geq y_j$ or $x_i \leq x_j$ and $y_i \leq y_j$. Kendall correlation coefficient is then defined in equation (19)

$$\tau^K_{X,Y} = \frac{(\text{ number of concordant pairs }) - (\text{ number of discordant pairs})}{\begin{pmatrix} N \\ 2 \end{pmatrix}} \tag{19}$$

where $\begin{pmatrix} n \\ 2 \end{pmatrix}$ is the number of ways to choose 2 observations in N observations.

Bibliography

Assaf Almog, Ferry Besamusca, Mel Macmahon, and Diego Garlaschelli. Mesoscopic community structure of financial markets revealed by price and sign fluctuations. *PLOS ONE*, 10, 04 2015. doi: 10.1371/journal.pone.0133679.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088/1742-5468/2008/10/p10008.

Kris Boudt, Jonathan Cornelissen, and Christophe Croux. The gaussian rank correlation estimator: Robustness properties. *Statistics and Computing*, 22:471–483, 10 2010. doi: 10.1007/s11222-011-9237-0.

Kris Boudt, Jon Danielsson, and Sébastien Laurent. Robust forecasting of dynamic conditional correlation garch models. *International Journal of Forecasting*, 29(2):244–257, 2013.

Joël Bun, Jean-Philippe Bouchaud, and Marc Potters. Cleaning large correlation matrices: tools from random matrix theory. *Physics Reports*, 666:1–109, 2017.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

Adina Roxana Feier. *Methods of proof in random matrix theory.* PhD thesis, Harvard University, 2012.

Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2009.11.002. URL https://www.sciencedirect.com/science/article/pii/S0370157309002841.

Heinrich Fritz, Peter Filzmoser, and Christophe Croux. A comparison of algorithms for the multivariate l1-median. *Computational Statistics*, 27(3):393–410, 2012.

Laurent Laloux, Pierre Cizeau, Jean-Philippe Bouchaud, and Marc Potters. Noise dressing of financial correlation matrices. *Physical review letters*, 83(7):1467, 1999.

Laurent Laloux, Pierre Cizeau, Marc Potters, and Jean-Philippe Bouchaud. Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*, 03(03):391–397, 2000. doi: 10.1142/S0219024900000255. URL https://doi.org/10.1142/S0219024900000255.

Jonas Liechti and Sebastian Bonhoeffer. A time resolved clustering method revealing longterm structures and their short-term internal dynamics. 12 2019.

Mel MacMahon and Diego Garlaschelli. Community detection for correlation matrices. *Phys. Rev. X*, 5: 021006, Apr 2015. doi: 10.1103/PhysRevX.5.021006. URL https://link.aps.org/doi/10.1103/PhysRevX.5.021006.

Vladimir A Marčenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457, 1967.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2975974.

Marina Meila. *Comparing Clusterings by the Variation of Information*, volume 16, pages 173–187. 11 2003. ISBN 978-3-540-40720-1. doi: 10.1007/978-3-540-45167-9_14.

Marc Potters, Jean-Philippe Bouchaud, and Laurent Laloux. Financial applications of random matrix theory: Old laces and new pieces. *arXiv preprint physics/0507111*, 2005.

Akihiko Utsugi, Kazusumi Ino, and Masaki Oshikawa. Random matrix theory analysis of cross correlations in financial markets. *Physical Review E*, 70(2), Aug 2004. ISSN 1550-2376. doi: 10.1103/physreve.70.026110. URL http://dx.doi.org/10.1103/PhysRevE.70.026110.