Market State Discovery

Unarine Singo

Supervisor: A/Prof. T. Gebbie

A minor dissertation submitted in partial fulfillment of the requirements

for the degree of Master of Science specialising in Data Science at the

Department of Statistical Sciences, University of Cape Town, Rondebosch

2022

We explore the concept of financial market state discovery by assessing the robustness of two unsupervised machine learning algorithms: Inverse Covariance Clustering (ICC) and Agglomerative Super Paramagnetic Clustering (ASPC). The assessment is carried out by: simulating market datasets varying in complexity; implementing ICC and ASPC to estimate the underlying states (using only simulated log-returns as inputs); and measuring the algorithms' ability to recover the underlying states, using the Adjusted Rand Index (ARI) as a performance metric. Experiments revealed that ASPC is a more robust and better performing algorithm than ICC. ICC is able to produce competitive results in 2-state markets; however, ICC's primary disadvantage is its inability to maintain strong performance in 3, 4 and 5-state markets. For example, ASPC produced ARI numbers that were up to 800% superior to ICC in 5-state markets. Furthermore, ASPC does not rely on the art of selecting good hyper-parameters such as, the number of states a priori. ICC's utility as a market state discovery algorithm is limited.

Keywords: Financial market states, time-series clustering, unsupervised machine learning.

Plagiarism Declaration

I am presenting this dissertation in partial fulfilment of the requirements for my degree. I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own. I hereby grant the University of Cape Town free licence to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever of the above dissertation.

Acknowledgements

I would like to express my sincerest gratitude to my supervisor A/Prof. Tim Gebbie for creating a safe space for me to learn and explore my interests interest in finance, for continually motivating me and remaining patient as I navigated this challenging journey.

Lionel Yelibi, thank for being open to sharing comments and guidance on how to navigate the complexity of these clustering algorithms.

To Irene, Moses, and Mantele, thank you for your unwavering support.

Contents

List of Figures

List of Tables

1 Introduction

Financial markets are an example of a complex adaptive system whereby a perfect understanding of the individual components will not lead to a sufficient understanding of the system's (or market's) behaviour. Asset prices emerge due to the collective behaviour of millions of purposeful agents, who compete with each other on information and expectations of their peer behaviour for personal gain. The nature of these interactions frequently feedback into the system, changing the market structure over time. Several authors (Wilcox et al. 2014; Authur 1995) have discussed the financial market through the lens of complexity theory with the view that this will lay the foundation to developing models that better encapsulate the real-world behaviour of financial markets. Under the framing of a complex adaptive system, a pertinent question is whether there exist states through which a financial market passes. Secondarily, if these states exist, can they be detected, modelled and predicted? The ability to identify and predict states would be a significant step in understanding how financial markets work and enable us to enhance the application of models in fields such as High-Frequency Trading and Asset Allocation, to name a few.

The concept of state discovery is not new. It has been studied under different names such as regime-switching (Hamilton 2005), temporal clustering (Hendricks et al. 2016), and volatility modelling (Bollerslev et al. 1992). However, the underlying objective is the same: to find a lower-dimensional representation of the complex structure of a financial market. Historically, market participants have developed their own language to summarise the behaviour of financial markets into discretised states. Bull, bear, recession and upturn are all common terminology analogous to a state representation of a market. Formally, the objective of state discovery is to cluster T time-series observations, $\{x_i\}_{i=1}^{T}$, defined by N dimensional assets, into equivalence classes. These equivalence classes are indices and are termed "market states".

Although several state discovery algorithms are in the literature, three key issues are yet to be reconciled. First, popular algorithms require pre-selection of the number of states to be estimated. Second, distributional assumptions are often imposed on the price dynamics. And third, many algorithms lack the ability to handle high dimensional datasets without compromising on computationally efficiency and performance. Inverse Covariance Clustering (ICC) and Agglomerative Super-Paramagnetic Clustering (ASPC) are two algorithms of interest that attempt to address these problems directly. ICC, proposed by Procacci et al. (2019), aims to solve the distributional assumption and big dimensional dataset problem by leveraging: Information Filtering Networks to estimate sparse precision matrices and filter out noise; And the Viterbi algorithm, which helps the algorithm achieve linear time complexity in a high dimensional setting. While the second problem of selecting states a priori remains unsolved, ICC offers additional flexibility through a

temporal-cohesion parameter (γ), which penalises frequent state switching. ASPC, proposed by Yelibi et al. (2019), addresses all three problems by postulating the market as a Potts-Spin model and uses an Agglomerative based Super-Paramagnetic Clustering algorithm (Blatt et al. 1996) to speed up computation.

The focus of this paper is a comparative study between the Inverse Covariance Clustering (ICC) and the Agglomerative Super-Paramagnetic Clustering (ASPC) algorithms. We first apply the ICC and ASPC methods to simulated datasets of varying market states, volatility and stock dimensions to explore their robustness. We then explore how these models can be applied to a simple asset allocation problem. We implement a vanilla mean-variance portfolio optimisation to S&P 500 data and compare the performance against a state-dependent mean-variance portfolio optimisation using states detected by ICC and ASPC.

Section 2 will serve as a literature review where we highlight the key historical pieces of work relating to market state discovery. In Section 3, we explain the methodology for all experiments, including: a review of the underlying theory and model formulation of ICC, ASPC, and mean-Variance portfolio optimisation; methodology to simulate state-dependant market data; and exploratory data analysis for the S&P 500 datasets. Our findings for the robustness tests and applications to real-world asset allocation are presented in Section 4. Lastly the conclusion and discussion in Section 5.

All code for this project is accessible on Github. We have also included the essential functions, source code and descriptions on how to implement them in the Appendices.

2 Literature Review: History of Financial Time Series Clustering

King (1996)'s dissertation paper is one of the earliest pieces of work to test the idea that large time-series of stocks can be clustered into groups that can help explain the behaviour of a market. In the introduction, he highlights that it was not common for market participants to consider the concept of prices moving together. This was substantiated further by a simple argument: when stocks are reported in averages of different industries (i.e., industrials, rails, and utilities), it is implied that participants have a latent clustering of stocks based on a measure of performance and similarity. In this study, factor analysis was applied to an observed covariance matrix. This matrix is built-up of monthly logged prices changes of 316 stocks of six SEC defined industries, namely: tobacco products, petroleum products, metals, railroads, utilities and retail stores. The intent was to separate the large datasets into clusters where the price movements of stocks in each cluster tend to move as a homogeneous group. Benjamin's key argument is that it is possible to break down price movements into market and industry groups that can explain the larger groups of stocks. However, he does warn that the process is highly susceptible to the estimate covariance matrix. It relies on the assumption of a stationary process, and it is commonly known that stationarity is not a feature of financial market data. He does conclude that more convincing results could be obtained by using methods that do not impose distributional assumptions on the data.

In the 1980s, Economics researchers studied the concept of market regimes related to the business cycle. Several studies looked to model macro-economic statistics such as Gross National Product (GNP) using auto-regressive approaches (Campbell et al. 1987; Beveridge et al. 1981). A common shortfall of these methods assumed stationary processes. Hamilton proposed a method that addresses this issue of stationarity (Hamilton 1989). In the study, US post-war GNP growth rates were assumed to result from an unobservable discrete Markov process where the state path of the process is referred to as market regimes. The market regimes that Hamilton identified were consistent with the National Bureau of Economic Research's (NBER) dating of market recessions suggesting this could be a feasible method for modelling and forecasting economic business cycles. However, a shortcoming of this method is that it is unable to handle many dimensions and requires the number of regimes (states) needs to be defined in advance.

In the econophysics domain, Mantegna (1999) was the first to propose a distribution-free method to identify stock sectors and market states which has gone to influence further works (over 2000 citations as of December 2021) in portfolio optimisation, trading strategies, to name a few. Most importantly, this seminal paper has influenced ASPC and ICC, the focus of this document. The algorithm consists of 4 main steps:

1. **Computing log-returns of a portfolio of stocks:**
   (a) Let N be the number of assets.
   (b) Let $P_i(t)$ be the price at time T.
   (c) Let $r_i(t)$ be the log-return at time t of asset i:

$$r_i(t) = logP_i(t) - logP_i(t-1). \tag{1}$$

2. **Compute pair-wise correlations:**
   (a) for each pair i, j assets, compute their correlation:

$$p_{ij} = \frac{\langle r_i r_j \rangle - \langle r_i \rangle \langle r_j \rangle}{\sqrt{(\langle r_i^2 \rangle - \langle r_i \rangle^2)(\langle r_j^2 \rangle - \langle r_j \rangle^2)}}. \tag{2}$$

3. **Convert pair-wise correlations to distance metrics:**
   (a) Covert the correlation coefficients, p_{ij}, into distances

$$d_{ij} = \sqrt{2(1 - p_{ij})}. \tag{3}$$

4. **Compute a Minimum Spanning Tree.**

The method above produces a tree with $N-1$ unique connections and no loop. From this tree, a clustering algorithm such as Single Linkage Clustering can be applied to produce a hierarchical dendrogram. From there, one can isolate groups of stocks that make economic sense. Mantegna tested this algorithm on a portfolio of 30 stocks stemming from the Dow Jones Industrial Average. By analysing the resultant Minimum Spanning Tree (Figure 1) and associated dendrogram (Figure 2), three groupings were observed. The most convincing group consisted of energy companies, namely: Chevron (CHV), Texaco (TX) and Exxon (XON). The second interesting group was Alcoa (AA) and International Paper (IP); both companies supply and beneficiate raw materials. The last group consisted of consumer companies, namely Procter & Gamble (PG) and Coca-Cola (KO).

Mantegna proceeds to apply this to a larger portfolio of stocks (443 stocks) used to construct the S&P 500 index and finds much more detailed and compelling clusters compared to the Dow Jones Industrial Average analysis. Given the size of the possible clusters, it is difficult to show the MST and associated dendrogram on a single image; however, parts of the MST with strong connections are shown in Figures 3 to 6. Four main groups of stocks with economic meaning were discovered, consisting of: (i) a group of financial services, capital goods, retailing, food and drink and tobacco companies; (ii) a group of

oil and energy companies; (iii) a group of communication and electrical utility companies; and (iv) a group of raw material companies. In the associated dendrogram (Figure 7), the branches of the tree are shown down to the level where groups begin to split. This means that the lines in the image always end in groups that contain two or more stocks belonging to the same industry or sub-industry, thus supporting the idea that the stock time series carry useful and detectable economic information.

Mantegna's work has influenced 20 years of research in the space. Marti et al. (2017) have documented in one place how this topic has evolved. They summarise some of the shortcomings of the original work and how other researchers have attempted to solve these challenges. Two main shortcomings of the original work have been identified:

1. **Unstable clusters caused by the choice of algorithm:**
   (a) Clusters produced by MST / Single Linakage Clustering Algorithm (SLCA) are known to drastically change when the input data is changed slightly (Marti et al. 2015).
   (b) It has been suggested that unstable clusters could be driven by the chaining effect (Gunnar et al. 2010) that comes with the SLCA. Meaning that SLCA tends to separate observations that intuitively belong to the same cluster, whereas Average Linkage and Complete Linkage has a preference towards building clique based clusters.
2. **Unstable clusters caused by the choice of distance metric:**
   (a) Using the Pearson correlation coefficient to define distances can be problematic as it is known for being sensitive to outliers and not being well suited to non-Gaussian distributions (Donnat et al. 2016).
   (b) It also turns out that high correlation edges in the MST does not necessarily mean that the the link between the two objects is reliable. Tumminello et al. (2007) proposed a bootstrap based measure of reliability of links within a correlation based network and found through empirical tests that some links with correlation as high as 0.7 had low reliability values.

Some enhancements to the original work come in the form of algorithms and adapting the distance metric:

1. **Alternative algorithms.**
   (a) **Average Linkage Minimum Spanning Tree**: To address the chaining problem that arises when using SLCA, the authors use the Average Linkage Clus-

tering Algorithm (ALCA) (Tumminello et al. 2007).

(b) **Planar Maximally Filtered Graph (PMFG)**: The authors proposed an alternative filtering algorithm to the MST that fully encompasses a MST whilst simultaneously maximising the amount of information in the tree (Tumminello et al. 2005) (Aste et al. 2005).

(c) **Triangulated Maximally Filtered Graph (TMFG)**: The TMFG was intended to be an improvement on PMFG's time complexity, as a result TMFG is better suited for big datasets. TMFG is also a core feature of the ICC algorithm studied in this document. (Massara et al. 2016).

(d) **Potts Super-Paramagnetic Transitions**: The authors in (Kullmann et al. 2000) propose a q-state Potts model approach to model the hierarchical structure of in S&P 500 and DJIA indices. This method was particularly interesting as it is able to handle anti-correlations, (negatively correlated stocks) which have been shown to change the structure of clusters. The idea of using Potts models for clustering is a key feature of the ASPC algorithm studied in this document.

2. **Alternative distance metrics.**

(a) **Shared information**: Granger causality (Billio et al. 2012) and Partial correlations (Kenett et al. 2010) have been used to understand whether a financial instrument provides information about another.

(b) **Capturing non-linearity**: Information theoretic distances such as mutual information have been tested on a number of occasions (Fiedor 2014) (Baitinger et al. 2017) so have Copula-based. (Marti et al. 2016) and tail dependence (Durante et al. 2015) distances have also been studied.

(c) **Multivariate effects**: Authors in Lee et al. (2017) have used Escoufier's RV coefficient (a multivariate generalisation of the Pearson's correlation coefficient) to include more information about a stock such as the high, low, open and closing prices.

These algorithms primarily focus on clustering stocks of financial markets and these same algorithms can also be used to detect market states. Rather than constructing a covariance matrix of stocks, one can construct covariance matrix of days. Naturally the number of dimensions increase significantly and computation time becomes extremely important, this is what ASPC and ICC intend to solve.

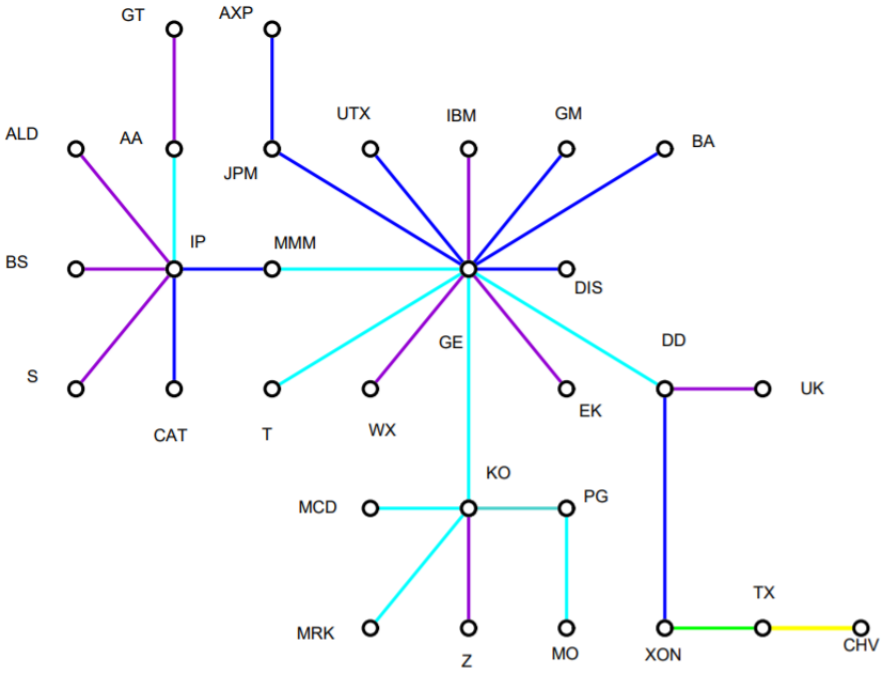


Figure 1: Minimum spanning tree to illustrate the stock clustering work of Mantegna (1999) as referenced in the Literature Review (Section 2). The tree connects 30 stocks of the Dow Jones Industrial Average from the period July 1989 to October 1995. The stocks are labelled by their trading symbols. The colours of the edges are proportional to the distance between the stocks: yellow $0.65< d(i,j) \leq 0.70$; green $0.70 < d(i,j) \leq 0.75$; turquoise $0.75 < d(i,j) \leq 0.80$; cyan $0.80 < d(i,j) \leq 0.85$; blue $0.85 < d(i,j) \leq 0.90$; violet $0.90 < d(i,j) \leq 0.95$ (Mantegna 1999).

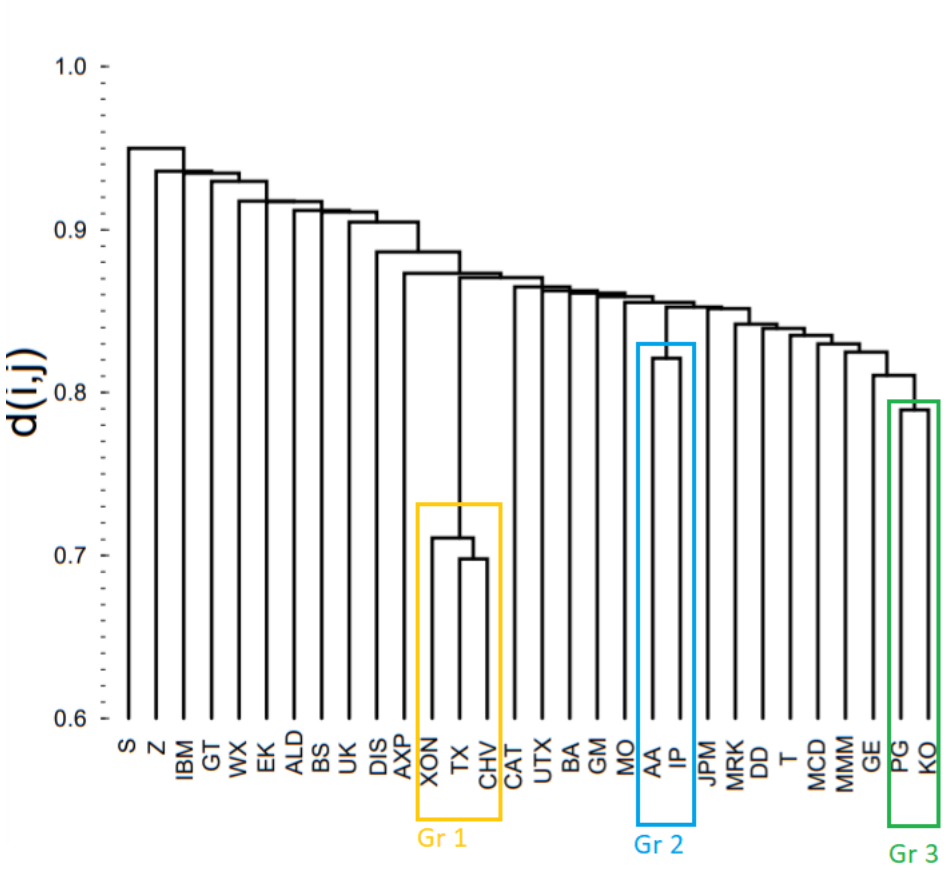


Figure 2: Hierarchical dendrogram associated with the MST in Figure 1 and is uncovered by a Single Linkage Clustering algorithm (Section 2). The similarity distances of tree branches are presented on the y-axis. Three interesting groups of which companies have similar activities are uncovered: energy companies (Exxon, Texaco and Chevron), raw materials producers (Alcoa, International Paper) and consumer product producers (Procter & Gamble, Coca-Cola) (Mantegna 1999).

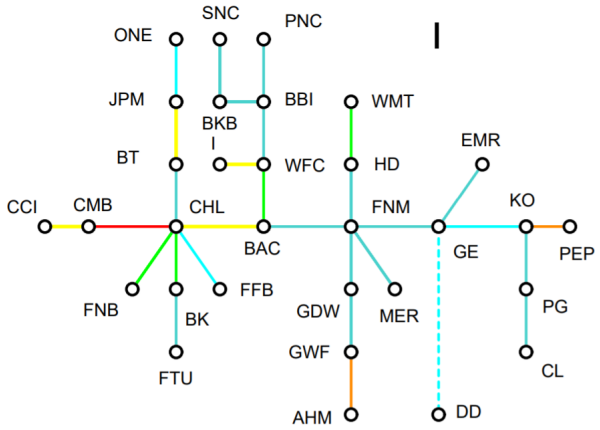


Figure 3: Second illustration of Mantegna (1999)'s seminal stock clustering work referenced in Section 2. Above is a partial minimum spanning tree (1 of 4) of strongly connected group of stocks used to compute the S&P 500 index over the period July 1989 to October 1995. The stocks are labelled by their trading symbols. The colours of the edges are proportional to the distance between the stocks: magenta $0.50 < d(i,j) \leq 0.55$; red $0.55 < d(i,j) \leq 0.60$; orange $0.60 < d(i,j) \leq 0.65$. This sub-tree consists of: financial services companies (AHM, BAC, BBI, BK, BKB, BT, CCI, CHL, CMB, FFB, FNB, FNM, FTU, GDW,GWF, I, JPM, MER, ONE, PNC, SNC and WFC), capital goods companies (EMR and GE), retailing companies (HD and WMT), consumer companies and food and drinks companies (KO and PEP).

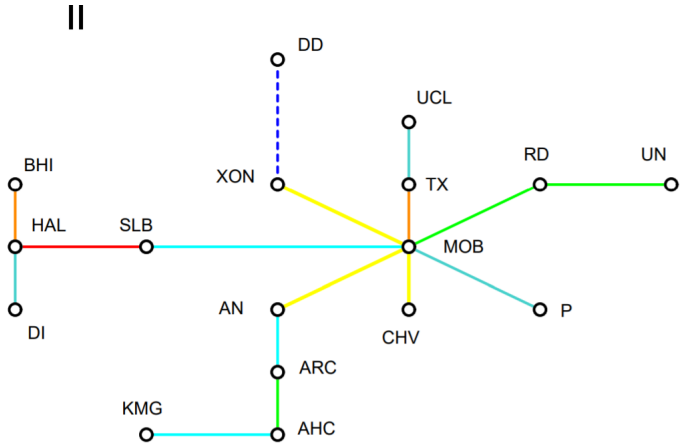


Figure 4: Second illustration of Mantegna (1999)'s seminal stock clustering work referenced in Section 2. Above is a partial minimum spanning tree (2 of 4) of strongly connected group of stocks used to compute the S&P 500 index over the period July 1989 to October 1995. The stocks are labelled by their trading symbols. The colours of the edges are proportional to the distance between the stocks: magenta $0.50 < d(i,j) \leq 0.55$; red $0.55 < d(i,j) \leq 0.60$; orange $0.60 < d(i,j) \leq 0.65$. This sub-tree consists of: primarily energy and oil companies, Unilever (UN) is the only company that does not align in an economic sense with the rest of the group.

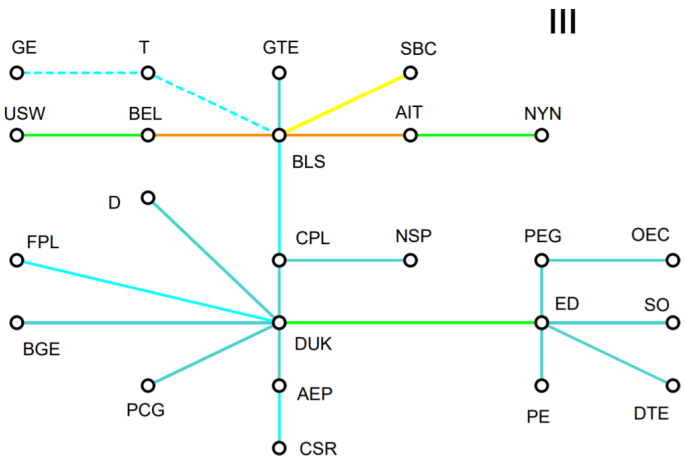


Figure 5: Second illustration of Mantegna (1999)'s seminal stock clustering work referenced in Section 2. Above is a partial minimum spanning tree (3 of 4) of strongly connected group of stocks used to compute the S&P 500 index over the period July 1989 to October 1995. The stocks are labelled by their trading symbols. The colours of the edges are proportional to the distance between the stocks: magenta $0.50 < d(i,j) \leq 0.55$; red $0.55 < d(i,j) \leq 0.60$; orange $0.60 < d(i,j) \leq 0.65$. This sub-tree consists of: electric utilities (AEP, BGE, CPL, CSR, D, DTE, DUK, ED, FPL, NSP, OEC, PCG, PE, PEG and SO) and in the sub-industry sector of telecommunications (AIT, BEL, BLS, GTE, NYN, SBC and USW). T, namely AT&T Corp., connects this group of stocks to other groups through General Electric (GE).

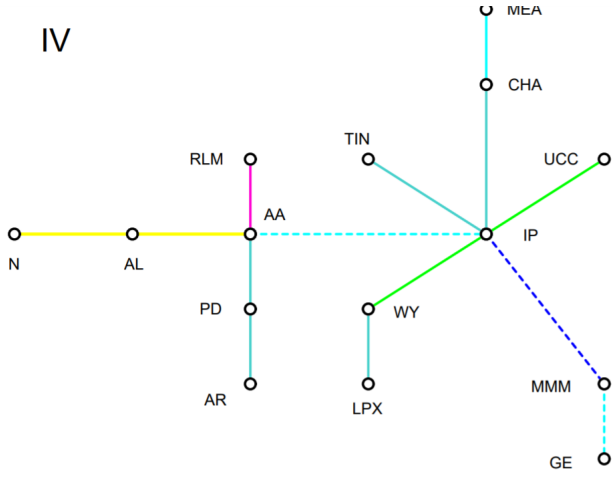


Figure 6: Second illustration of Mantegna (1999)'s seminal stock clustering work referenced in Section 2. Above is a partial minimum spanning tree (4 of 4) of strongly connected group of stocks used to compute the S&P 500 index over the period July 1989 to October 1995. The stocks are labelled by their trading symbols. The colours of the edges are proportional to the distance between the stocks: magenta $0.50 < d(i,j) \leq 0.55$; red $0.55 < d(i,j) \leq 0.60$; orange $0.60 < d(i,j) \leq 0.65$. This sub-tree consists of: raw materials. Two subgroups are observed: (i) companies working in the industry sector of metals and sub-industry nonferrous materials (AA, AL, AR, PD, N and RLM) and companies working in the industry sector of forest products and packaging (CHA, IP, LPX, MEA, TIN, UCC and WY).

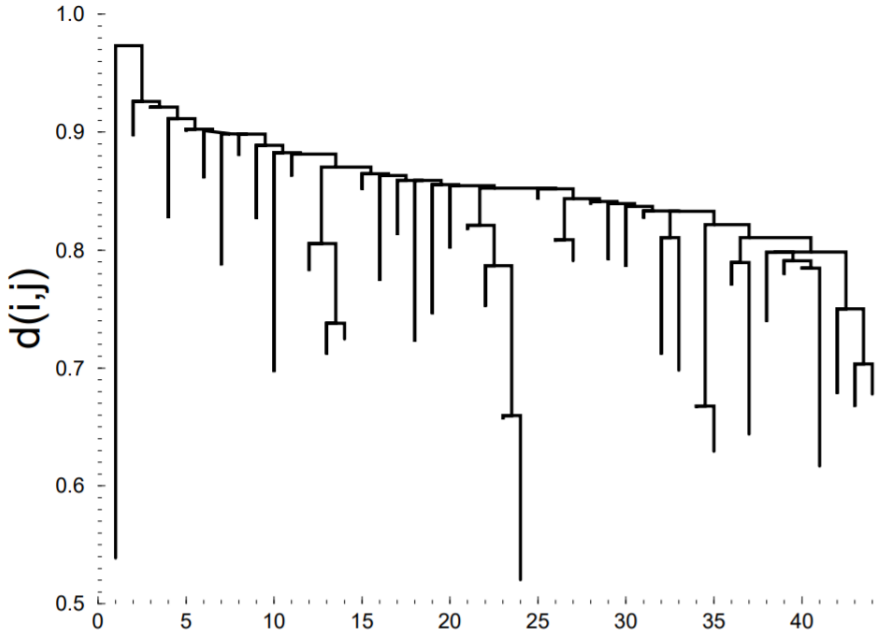


Figure 7: Hierarchical dendrogram associated with the partial MSTs in figures 3 to 6 and is uncovered by a Single Linkage Clustering algorithm (Section 2). Each line represents a group of two or more stocks and groups are labelled 1 to 44 (Mantegna 1999).

3 Research Methodology

3.1 Data I: Exploratory Data Analysis of Standard and Poor's (S&P) 500 Data

The S&P 500 is an index of the most commonly used stocks in the United States; we used stock samples of the S&P 500 for all simulations and modelling in this document. The utilisation period in the original dataset (sourced from Kaggle Nugent 2018) ranges from February 2013 to February 2018. It consisted of 504 daily open and close stock prices and the daily volume traded. We reduced the dataset to 470 stocks by retaining stocks that were traded for the entire period.

Figure 8 is a graphical summary of the surviving dataset. In frames one and two, we plot the whole market cumulative daily log-returns and daily log-returns, respectively. In frame three, we plot the number of stocks traded per day for the period. Lastly, frame four is a histogram of the daily log-returns for the period, which appears normally distributed with fat tails, particularly on the negative side around -0.04.

By simple visual analysis, the market structure between mid-2015 and mid-2016 is significantly different from the rest of the dataset. More specifically, we see the signs of a bear market state, with a downward trend in returns (frame one), high volatility (frame two) and a significant decrease in daily volume traded followed by a steep upward increase. One would expect either ICC or ASPC to detect these changes in market states.

Regarding how the data is used for implementation, we randomly sample 100 stocks with each iteration which serves as input to the models (Note: this is the same methodology followed in the ICC paper). See Figures 9 and 10 for an illustration of the sampled dataset. By simple visual analysis, we can see that the market trend of the sampled dataset is sufficiently aligned to the full dataset in Figure 8.

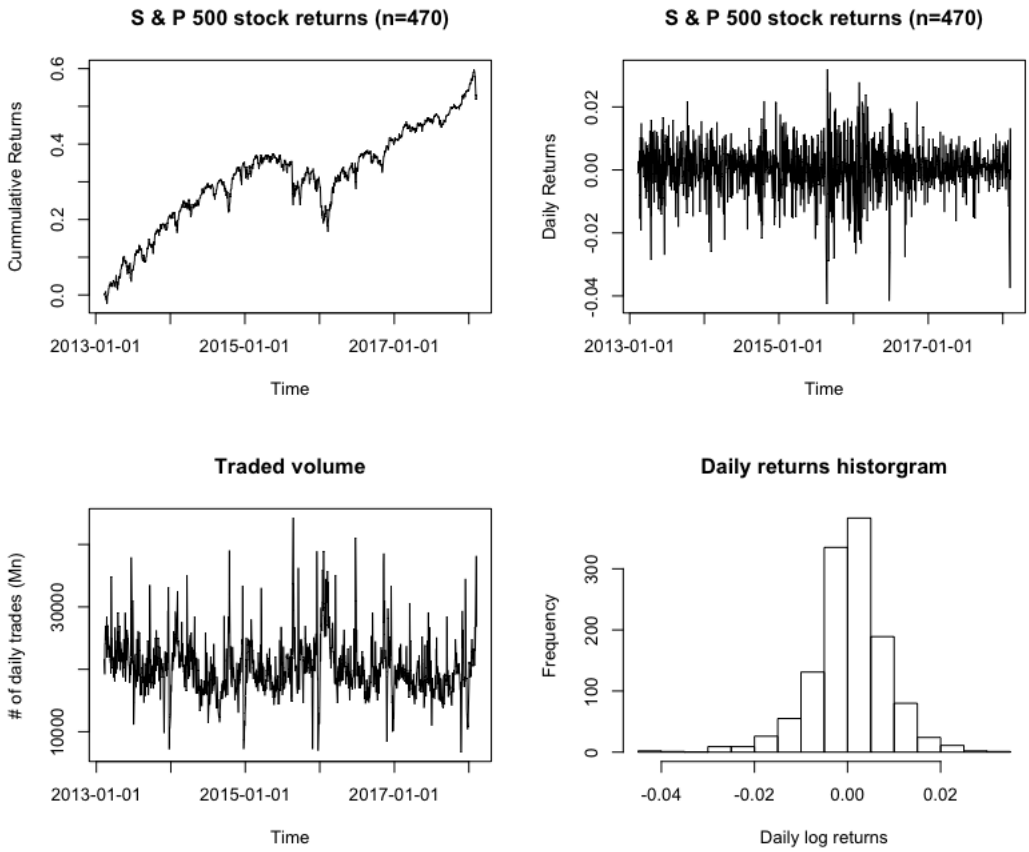


Figure 8: As part of the exploratory data analysis (Section 3.1) above is a graphical summary of the S&P market between February 2013 and February 2018. Only surviving stocks (stocks fully traded over the period) as selected. Frame 1: Cumulative market average daily log-return, Frame 2: Market average daily log-return; Frame 3: Total volume of stocks traded per day; Frame 4: Histogram of market average daily log-returns. Referring to frame 2, it's clear the market moves through at least two market states characterised by periods of high and low variability.

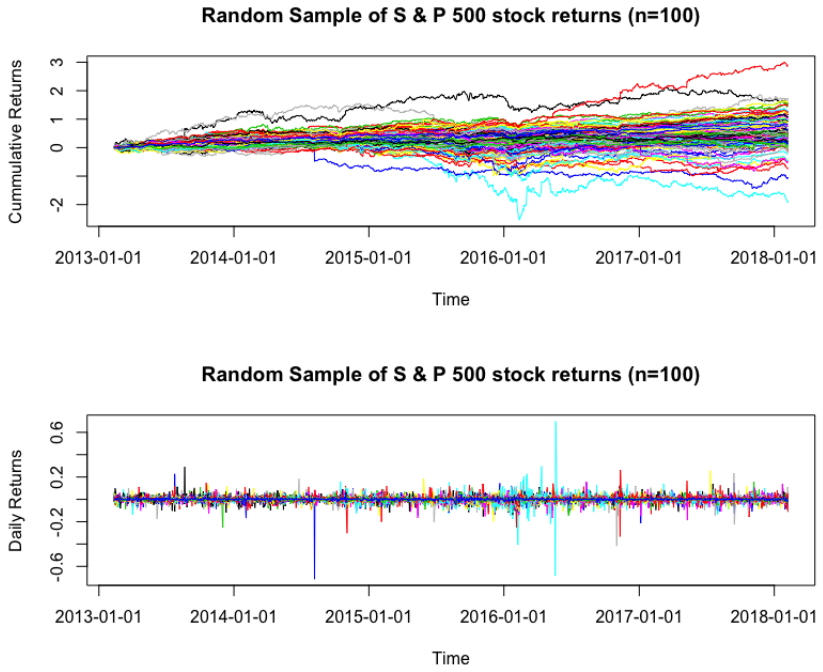


Figure 9: As part of the exploratory data analysis (Section 3.1) above is a graphical summary S&P market stocks between February 2013 and February 2018. Only surviving stocks (stocks fully traded over the period) as selected. Frame 1: Cumulative daily log-returns per stock, Frame 2: Daily log-returns per stock.

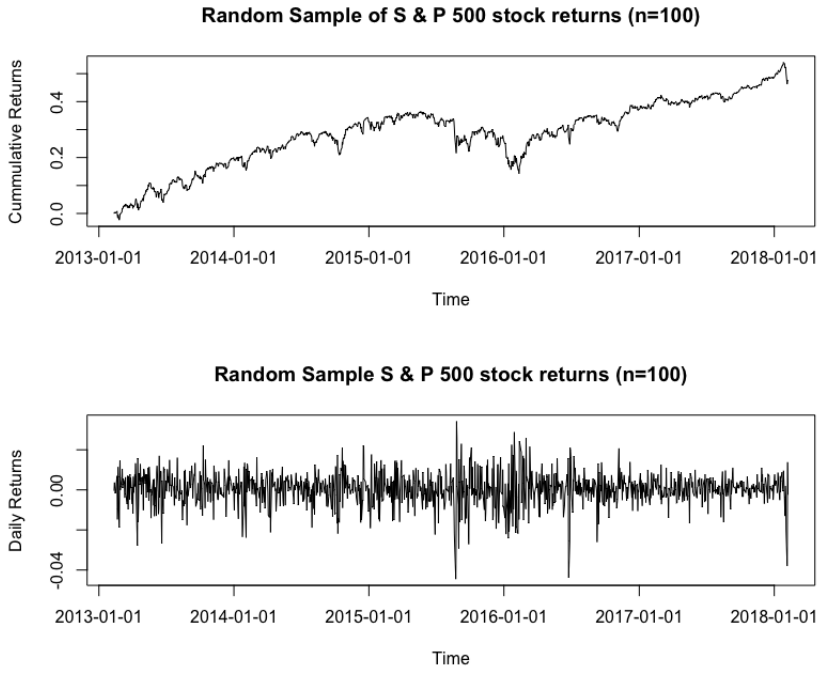


Figure 10: As part of the exploratory data analysis (Section 3.1) above is a graphical summary of 100 randomly sampled S&P 500 market stocks between February 2013 and February 2018. Only surviving stocks (stocks fully traded over the period) as selected. Frame 1: Cumulative market average daily log-return, Frame 2: Market average daily log-return. 100 stocks are samples simply to adhere to the modelling steps specified in the ICC paper (Procacci et al. 2019). By visually comparing the graphs in Figure 8 and the Figure above, we see that 100 stocks is a reasonably large enough sample to capture the market dynamics.

3.2 Inverse Covariance Clustering

3.2.1 Theory and Model Formulation

Inverse Covariance Clustering (ICC) is a financial time series clustering algorithm proposed by Procacci et al. (2019). The algorithm is based on the belief that financial markets move through states where the covariance structure changes in time. As such, multivariate observations of stock returns can be grouped into clusters of days where observations that are similar in distribution belong to the same cluster and dissimilar objects to different clusters. The authors highlight that the methodology differs from more commonly used methods such as Markov-Switching (Hamilton 2005) models and Volatility Clustering (Bollerslev et al. 1992) models due to its ability to cluster high-dimensional datasets - for example, a large number of stocks - without compromising on computational performance.

Consider a large data set $\mathbf{X} = \{\vec{\mathbf{x}}_i\}_{i=1}^{T}$ composed of T sets of multivariate n stock return observations denoted as $\vec{\boldsymbol{x}}_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,n}]$. T can represent any sampling frequency, in the paper the authors focus on daily sampled data. The algorithm begins by selecting the number of states K to be clustered. Observations are then randomly assigned to K sets. For each partitioned set, a sample mean, $\vec{\boldsymbol{\mu}}_k$, and sample sparse precision matrix $\mathbf{J_k}$ are computed, then observations are iteratively re-assigned to clusters with the smallest penalised distance:

$$\mathcal{M}_{t,k} = d_{t,k}^2 + \gamma \mathbb{1}\left\{\mathcal{K}_{t-1} \neq k\right\}, \tag{4}$$

where $d_{t,k}^2 = \left(\vec{\boldsymbol{x}}_t - \vec{\boldsymbol{\mu}}_k\right)^{\mathrm{T}} \boldsymbol{J}_k \left(\vec{\boldsymbol{x}}_t - \vec{\boldsymbol{\mu}}_k\right)$ is the square Mahalanobis distance of observation $\vec{\boldsymbol{x}}_t$. γ, is a state switching penalisation parameter which regularises the objective function to identify temporally coherent state sequences. The sparse precision matrix, $\mathbf{J_k}$, is computed using a information filtering technique titled: Triangulated Maximally Filtered Graph - Local/Global inverse matrix (TMFG-LoGo) (Massara et al. 2016).

Finally the optimal state sequence is identified using an iterative two step approach similar to the Expectation-Maximisation algorithm:

1. Given a randomised state path (or cluster partitions), compute sample statistics $\vec{\boldsymbol{\mu}}_k$, $\mathbf{J_k}$ per cluster k. Using the Viterbi algorithm, minimise the total distance by re-assigning observations to the smallest $\mathcal{M}_{t,k}$.
2. Given the Viterbi optimal path, update sample statistics and repeat until convergence.

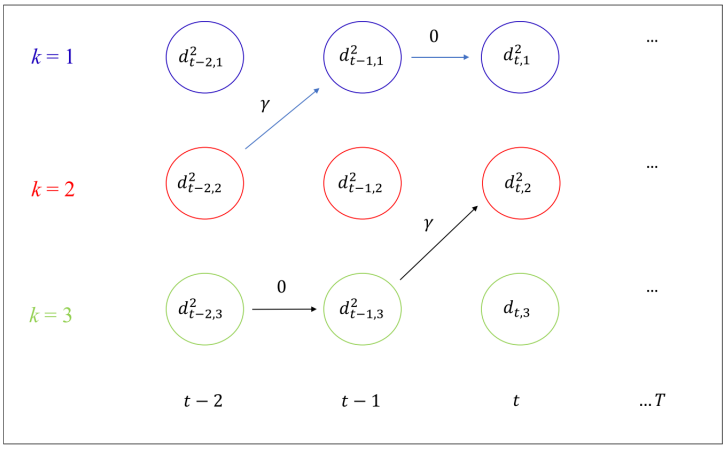


Figure 11: Illustration of how the Viterbi Algorithm (Section 3.2.1.1) sequentially assigns observations to states. This examples shows the possible pathways of allocating market days to one of three states ($K = 3$). The Viterbi algorithms to allocate days in a way that minimises the the total distance (Equation 4) and state switching is penalised by the hyper-parameter γ (Procacci et al. 2019).

This process is repeated until the total distance converges.

In the following sections we break up and review key components of the ICC algorithm and conclude with a preliminary visual presentation of the replication results.

3.2.1.1 Viterbi Algorithm

The Viterbi algorithm (Viterbi 1967) is a forward-pass dynamic programming algorithm. Its most popular use is in decoding hidden state sequences of Hidden-Markov Models. It transforms an otherwise polynomial time $O\left(K^T\right)$ procedure to a linear time complexity $O\left(KT\right)$. In order to find the best sequence of states, we need to consider all potential cluster assignments per observation; the problem becomes combinatorial hence the need for a dynamic programming solver. Figure 11 illustrates the problem of assigning observations considering $K = 3$ possible paths. Recall $d^2_{t,k}$ is the Mahalanobis distance of observation $\vec{\boldsymbol{x}}_t$ in state k. If an observation is assigned to the same cluster as the previous one, no penalty is applied; otherwise, a cost of γ is incurred for switching between states.

Algorithm 5 shows how we implemented the Viterbi algorithm in `R`. This version is almost identical to what is reported in the ICC paper except for lines 18 - 19. Originally these lines were found inside the inner-loop; however, we believe this could be a printing error as the original specification produces invalid results.

Algorithm 1: Viterbi algorithm

Input ;
$d^2_{t,k}$ = square Mahalanobis distance of observation t if assigned to state k ;
γ = time consistency parameter ;
Initialize ;
PreviousCost = array of K zeros ;
CurrentCost = array of K zeros ;
PreviousPath = array of K elements ;
CurrentPath = array of K elements ;
for *each observation* $t = 1, ..., T$ **do**
    for *each state* $k = 1, ..., k$ **do**
        $MinVal$ = index of minimum value of PreviousCost ;
        if *PreviousCost*$[MinVal] + \gamma >$ *PreviousCost* $[k]$ **then**
            CurrentCost$[k]$ = PreviousCost + $d^2_{t,k}$;
            CurrentPath$[k]$ = PreviousPath$[k]$.append$[k]$;
        else
            CurrentCost$[k]$ = PreviousCost + γ + $d^2_{t,k}$;
            CurrentPath$[k]$ = PreviousPath$[MinVal]$.append$[k]$;
    PreviousCost = CurrentCost ;
    PreviousPath = CurrentPath ;
FinalMinVal = index of minimum value of CurrentCost ;
FinalPath = CurrentPath $[FinalMinVal]$;

3.2.1.2 TMFG-Logo Algorithm

Sample covariance matrices tend to suffer from noisy estimates, leading to unstable clusters. Given that the objective is to discover the market structure and infer clusters that make economic sense, it is in our best interest to handle noise efficiently. TMFG-LoGo (Barfuss et al. 2016) aims to address this problem by providing a methodology to estimating a sparse precision matrix. In this methodology, the covariance matrix is positioned as a graph where vertices represent stocks, and edges are covariance estimates between pairs of stocks. Redundant information is filtered by removing edges that contribute the least amount of information, and then the graph-matrix is inverted to find the precision matrix. Two procedures enable TMFG-LoGo: TMFG (Massara et al. 2016) is responsible for converting a full covariance matrix into a sparse covariance matrix; LoGo (Barfuss et al. 2016) is the inversing procedure that solves the challenge of inverting large matrices by inverting sub-parts of the sparse graph-matrix produced by TMFG.

3.2.1.3 TMFG

Filtering information in complex graphs has been studied in literature by primarily using one of two approaches: the Minimum Spanning Tree (MST) and the Planar Maximally Filtered Graph (PMFG) (Marti et al. 2017). The common idea is to filter a dense matrix of weights by retaining the largest and most significant subgraph possible while imposing global constraints on the topology of the resulting graph. PMFG aims to produce a subgraph of the most significant weights (e.g., largest covariance estimates) while constraining the subgraph to be globally planar. Planar filtered graphs have been used effectively to diversify financial risk and build sufficiently diversified portfolios; however, implementing the algorithm is computationally expensive. TMFG is an extension of PMFG; it aims to produce planar filtered graphs while improving computational efficiency.

TMFG construction begins with a triangular planar-maximal clique (subgraph) of order 4. The next step is to add a vertex into a triangular face of the latest graph that maximises the sum of newly connected edges - this insertion is called T_2 move (See Figure 12). Concretely, a score function, $S(v_h, \{v_a, v_b, v_c\})$, is used to assess the gain achievable by adding vertex v_h inside triangle $\{v_a, v_b, v_c\}$. For example, given a dense matrix W, the gain achievable by inserting v_h in face $\{v_a, v_b, v_c\} : S(v_h, \{v_a, v_b, v_c\}) = W(v_h, v_a) + W(v_h, v_b) + W(v_h, v_c)$. Additionally, it is important to highlight why the T_2 operator is particularly attractive. Firstly, unlike PMFG, each insertion enforces the planarity constraint. Secondly, graphs produced by T_2 are chordal: every cycle of length greater than 4 has a *chord*, an edge not belonging to the cycle but connects two vertices of the cycle. Chordal are graphs polynomial time algorithms used to solve NP-hard problems such as, finding a maximum clique, graph coloring, and maximum independent set. Lastly a key output of TMFG is a set of cliques and seperators defining the sparsified covariance matrix, which serves as a key input into the LoGo procedure. Algorithm 2 illustrates in more detail how TMFG was implemented.

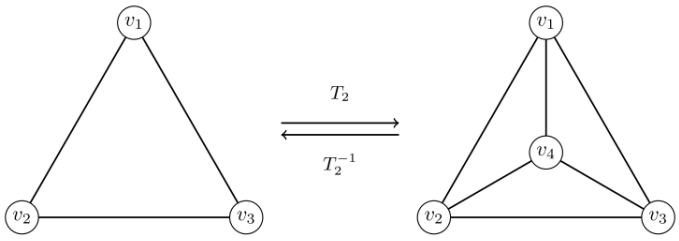


Figure 12: Illustration of how the TMFG algorithm (see Section 3.2.1.3) builds cliques by inserting vertices into a a triangular graph. More specifically, T_2 is an insertion that adds v_4 into a strictly 3-clique triangular graph $\{v_1, v_2, v_3\}$ such that a score function is maximised (Massara et al. 2016).

Algorithm 2: TMFG algorithm

Input ;
A dense $p \times p$ square matrix $\mathcal{W}$ with positive weights (i.e., covariance estimates) ;
Output ;
A sparse matrix, $\mathcal{P}$, a filtered version of $\mathcal{W}$ fulfilling the planarity constraint ;
Initialize ;
$\mathcal{C}_1 \leftarrow$ Tetrahedron, $\{v_1, v_2, v_3, v_4\}$, with highest overall total score ;
// Assign the four triangular faces in $\mathcal{C}_1$ to the array $\mathcal{T}$;
$\mathcal{T} \leftarrow \{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}$;
// Put the $p-4$ vertices not belonging to $\mathcal{C}_1$ in the array $\mathcal{V}$;
$\mathcal{V} \leftarrow \{v_5, \cdots, v_p\}$;
// Create an empty list of Separators ;
$\mathcal{S} \leftarrow \varnothing$;
// Assign the first tetrahedron to the list of cliques
$\mathcal{C} \leftarrow \mathcal{C}_1$;
$\mathcal{P} \leftarrow \mathcal{W}(\mathcal{C}_1, \mathcal{C}_1)$;
Calculate MaxGain for $\mathcal{T}$ and $\mathcal{V}$ as in Eq.(1);
Calculate BestVertex for $\mathcal{T}$ and $\mathcal{V}$ as in Eq. (2) ;
Main ;
// Insert $p-4$ vertices via T_2 ;
while $\mathcal{V}$ *is not empty* **do**
    Find the $t_i \in \mathcal{T}$ and $v_i \in \mathcal{V}$ that achieve the maximum in MaxGain ;
    Insert v_i into t_i // this creates three new triangles t_a, t_b, t_c ;
    $\mathcal{V} \leftarrow \mathcal{V} \backslash v_i$;
    $\mathcal{T} \leftarrow (\mathcal{T} \backslash \{t_i\}) \cup \{t_\alpha, t_b, t_c\}$;
    $\mathcal{S}_i \leftarrow \{t_i\}$;
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_i$;
    $\mathcal{C}_i \leftarrow \{t_i, t_a, t_b, t_c\}$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_i$;
    $\mathcal{P} \leftarrow \mathcal{P} + \mathcal{W}(\mathcal{C}_i, \mathcal{C}_i) - \mathcal{W}(\mathcal{S}_i, \mathcal{S}_i)$;
    Update MaxGain and BestVertex to reflect the changes in $\mathcal{T}$ and $\mathcal{V}$
return $\mathcal{P}$

3.2.1.4 LoGo

TMFG provides us with a chordal (decomposable) graph, comprised of a clique-tree constituted by 4-cliques, $\mathcal{C}_m$, connected with 3-clique separators, $\mathcal{S}_n$. Given that the resultant graph is a sparse covariance matrix, a precision matrix can be computed on local inversions of sub-covariance matrices associated with the cliques and separators:

$$J_{i,j} = \sum_{c \text{ s.t. } \{i,j\} \in C} \left(\mathbf{\Sigma}_C^{-1}\right)_{i,j} - \sum_{S \text{ s.t. } \{i,j\} \in \mathcal{S}} (k(\mathcal{S}) - 1) \left(\mathbf{\Sigma}_{\mathcal{S}}^{-1}\right)_{i,j}, \tag{5}$$

where the term $k(\mathcal{S}_n)$ counts the number of disconnected components produced by removing the separator, $\mathcal{S}_n$. This formula simplifies the global matrix inversion problem into the sum of local matrices of sizes no larger than 4 when TMFG is used. Figure 13 illustrates how the inverse covariance matrix is calculated in a system of ten variables with seven cliques and six separators.

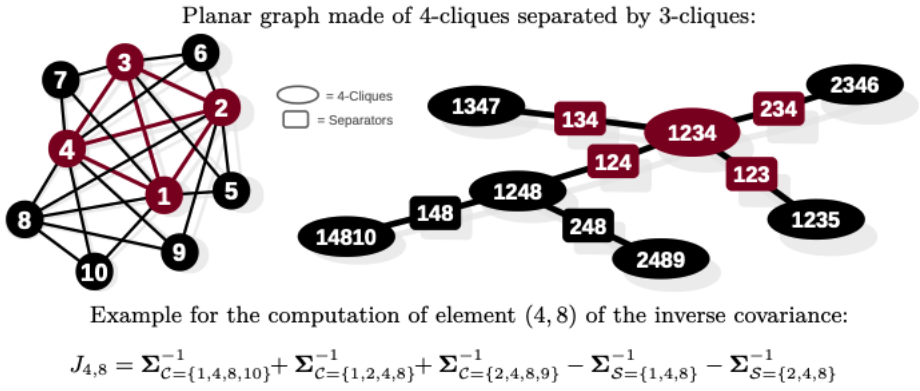


Figure 13: Above is an illustration of Local-Global inversion of the covariance matrix procedure explained in LoGo (see Section 3.2.1.4). Example for a system of $p = 10$ variables associated with a decomposable TMFG graph with $M_c = 7$ cliques and $M_s = 6$ separators (Barfuss et al. 2016).

3.2.1.5 Safely Computing the TMFG-LoGo Log-Determinant

A key requirement of ICC based forecasting is being able to compute the log-likelihood function:

$$\mathcal{L}_{s,k} = 1/2 \left(\log \|\mathbf{J}_k\| - d^2_{s,k} - p \log(2\pi)\right). \tag{6}$$

We discovered that direct computation of the determinant of the TMFG-LoGo precision would fail for large stock universes. We observed that for universe sizes larger than 80 stocks, the determinant would equal "Inf" (see Figure 14); this would in-turn impede our ability to compute the log-determinant. The leading cause for this error is that the value of the determinant is too large to be represented in R; this is commonly referred to as computational overflow. However, we are only interested in the log-determinant rather than the determinant itself. To solve this problem, we kept computations in the logarithmic scale by utilising the Cholesky Decomposition and some properties of triangular matrices (Lin 2020).

Given a positive definite matrix $\mathbf{J}_k$, the Cholesky decomposition factorises to a lower triangular matrix $\mathbf{L}$ multiplied by its transpose:

$$\mathbf{J}_k = \mathbf{L}\mathbf{L}^T. \tag{7}$$

For a triangular matrix, the determinant is equal to the product of its diagonal elements; hence, the log-determinant is equal to the sum of the logarithm elements. We can compute the log-determinant of large stock universes as:

$$\log \|\mathbf{J}_k\| = \sum \left[2 \log \left(diag(\mathbf{L})\right)\right]. \tag{8}$$

In Figure 14 we show in red the log-determinant for varying stock universe sizes of a simulated market of daily returns where $r_{t,p} \sim \mathrm{N}\left(\mu = 0, \sigma^2 = 0.015^2\right)$. Note that for stock sizes larger than 90 a log-determinant value cannot be plotted but for the Cholesky approach we are able to recover the same log-determinant values and compute for larger stock sizes.

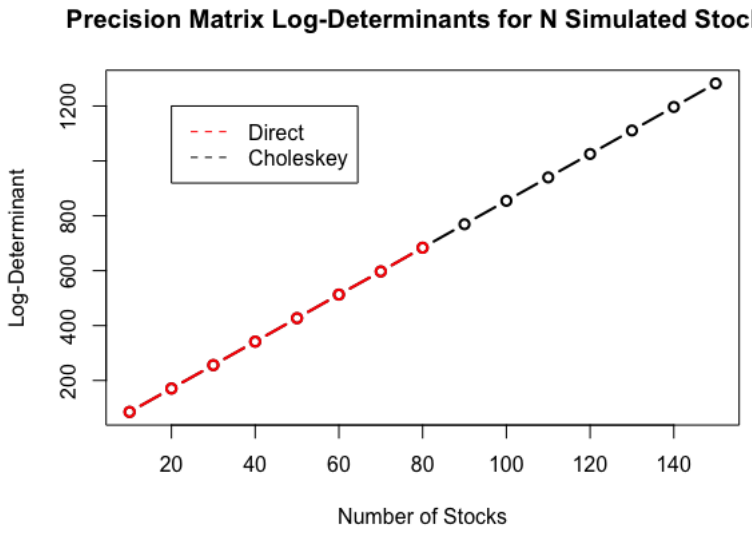


Figure 14: The Figure above further substantiate the challenges faced when trying to compute the precision matrix within the ICC objective function as discussed in Section 3.2.1.5. Computing the TMFG-Logo log-determinant directly (in Equation 4) tends to fail when the number of stocks are larger than 90. The red line shows that a rational solution is found up to 90 stocks, beyond that an infinite value is returned. Using the Cholesky factorisation (Equation 7) and log-transforming the diagonal values (Equation 8) we can compute the log-determinants for larger stock sizes and, ultimately, the log-likelihood function (in Equation 4).

3.2.2 Preliminary Replication Results

As a preliminary report, we illustrate how all the Viterbi, TMFG and LoGo algorithms come together to form the ICC algorithm R implementation. Our ICC implementation can be found in `MarketStates/R/05_ICC.R` directory and the replication script in `MarketStates/Scripts/ICC_Replication_script.R` and the appendix.

We used a random sample of 100 S&P 500 stocks. By setting $K = 2$ and $\gamma = 16$ we were able to identify states in a similar manner to the original work. In Figure 15 we show two graphs, the identified states of the average market daily log-return (left) and cumulative market daily log-return (right). The purple states appear to be times of increased market volatility, whereas the blue states have the opposite characteristic - lower volatility and predominantly positive returns. A different perspective can be observed by plotting each stock and state's average and standard deviation. In Figure 16 we see the red state (which corresponds to the purple state in Figure 15) has occurrences of significant negative returns and large jumps in volatility relative to the blue state (which corresponds to the blue state in Figure 15). Furthermore, we can compute the Sharpe ratios for each stock and overlay the two states; in Figure 16 we see larger risk-adjusted returns from the blue state relative to the red state. These results are consistent with those found in the paper.

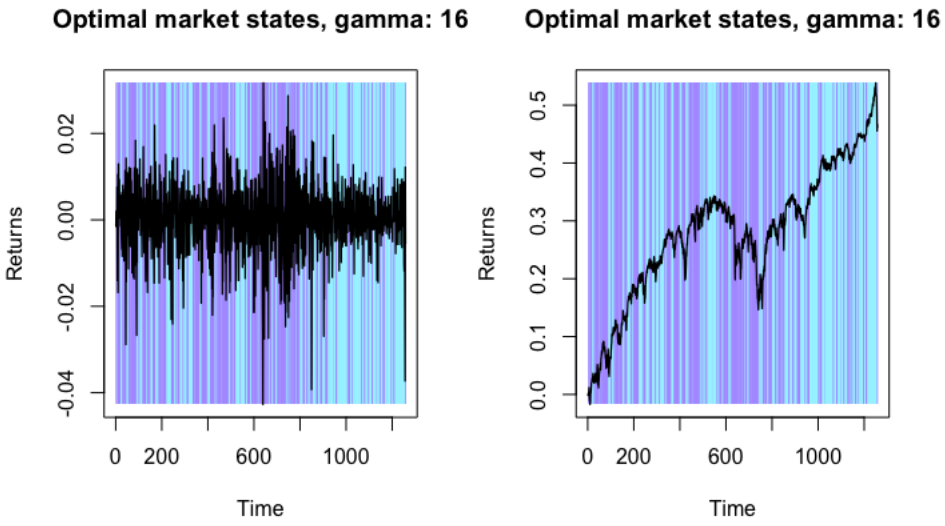


Figure 15: ICC implemented on S&P 500 dataset of a random sample of 100 stocks to identify a 2-state market (Section 3.2.2). Daily market returns are displayed on the left and cumulative market returns on the right. Optimal market states are superimposed onto both graphs in the colour purple and blue. As discussed in the exploratory data analysis (Section 3.1) it seems that a 2-state market is a reasonable assumption, ICC is able to identify states associated with high market volatility and downward market trends (purple) as well as low market volatility and upward market trends (blue).

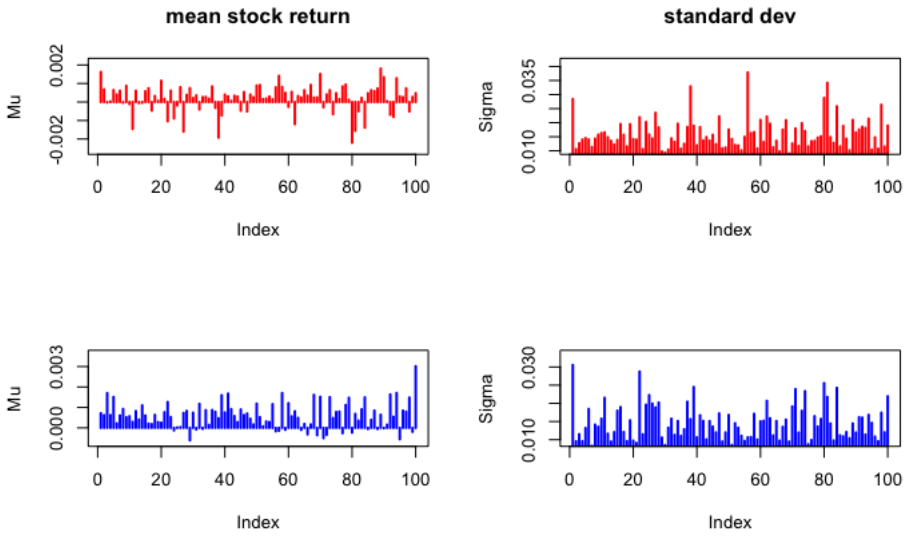


Figure 16: ICC implemented on S&P 500 dataset of a random sample of 100 stocks to identify a 2-state market (Section 3.2.2). The daily average return and standard deviation for each stock is computed. The red and blue states correspond to the purple and blue states in Figure 15. Similarly, the first state (red) corresponds with periods of higher volatility across the market relative to the second state (blue). With respect to market average returns, all the stocks of in the second state (blue) yield positive and higher magnitude returns relative to the first state (red).

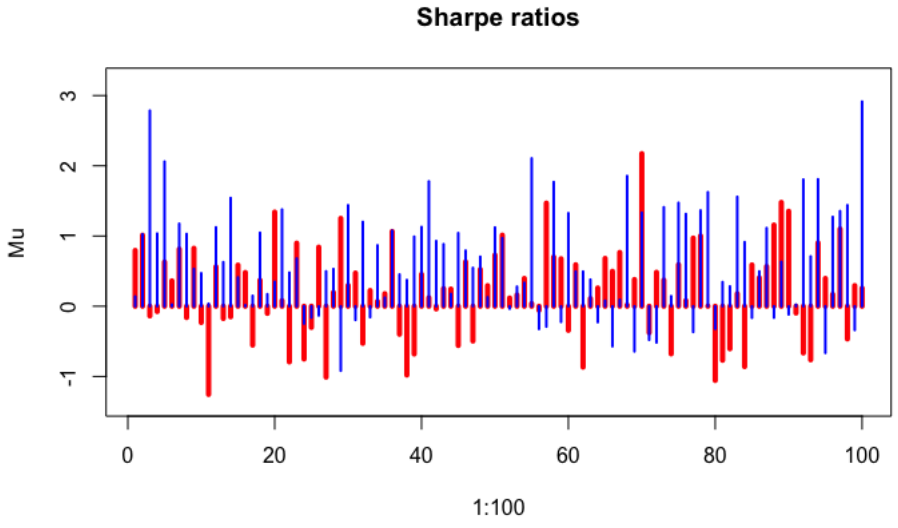


Figure 17: Using the same data and optimal path discovered in Section 3.2.2, sharpe ratio statistics for each state are computed and stock are computed. The red and blue states correspond to the purple and blue states in Figure 15. Once again it is clear that ICC is able to allocate market days into states that are sufficiently differentiated given the return and volatility dynamics.

Algorithm 3: ICC algorithm

Input ;
P = number of stocks ;
T = length of time-series ;
X = P X T matrix of stock returns ;
γ = state switching parameter ;
Initialize ;
states = randomly shuffled state series length T ;
K = Number of states;
Θ = list of sample statistics ($\vec{\boldsymbol{\mu}}_k$, $\mathbf{J_k}$ per cluster k) based on state allocation ;
Distance = K X T matrix containing the distance of multivariate observations conditioned on Θ ;
for *each iteration it* = 1, ...,*max.iters* **do**
    //E-Step ;
    OptimalViterbi = `viterbi`(Distance, K, γ) // reallocates observations to states and returns an optimal state path;
    // Check if state path is valid, i.e., no single state path returned ;
    if *Sum(diff(OptimalViterbi)) == 0* **then**
        states = reject Viterbi states and randomly shuffle new state path;
    else
        states = OptimalViterbi ;
    //M-Step ;
    Θ = update sample statistics ($\vec{\boldsymbol{\mu}}_k$, $\mathbf{J_k}$ per cluster k) based on new state allocation ;
    Distance = update distance matrix conditioned on new Θ ;
FinalStatePath = states ;

3.3 Agglomerative Super-Paramagnetic Clustering

3.3.1 Theory and Model Formulation

ASPC is a fast and flexible algorithm capable of identifying market states and asset sectors of high dimensional financial datasets (Yelibi et al. 2019). At its core, ASPC is based on the work of Giada et al. (2001), who developed a likelihood-based algorithm to identify clusters. In principle, the model removes the noise of correlated datasets without making distributional assumptions of the underlying data or deciding the number of states a priori, which is a common pitfall in most popular clustering algorithms such as ICC, Gaussian Mixture Models and K-Means. To understand ASCP better, we begin by reviewing the Giada et al. (2001) Likelihood model.

3.3.1.1 Giada-Marsili Likelihood Algorithm

Optimising objective functions to identify states in high dimensions is a hard problem. The Giada et al. (2001) Likelihood algorithm solves this problem by drawing inspiration from the Potts Model (Blatt et al. 1996), which is traditionally used to study the spins of the interacting systems under thermal fluctuations.

Consider a dataset $\Xi = \left\{\vec{\xi_i}\right\}_{i=1}^{N}$ composed of N objects with D measurable observations so that each object is represented in the vector $\vec{\xi_i} = \{\xi_i(d)\}_{d=1}^{D}$. For simplicity the objects of this dataset are assumed to be normalised with zero mean ($\sum_d \xi_i(d)/D = 0$) and variance of one ($\sum_d \xi_i^2(d)/D = 1$). The correlation matrix of this dataset is defined by:

$$C_{i,j}(D) \equiv \frac{1}{D}\sum_{d=1}^{D} \xi_i(d)\xi_j(d). \tag{9}$$

In their study of financial market correlations, Noh (2000) proposes an anzats based on the assumption that objects are similar if they have a common component. Hence, an object ($\xi_i(d)$) from the dataset can be expressed by:

$$\xi_i(d) = \frac{\sqrt{g_{s_i}}\eta_{s_i}(d) + \epsilon_i(d)}{\sqrt{1+g_{s_i}}}, \tag{10}$$

where $\xi_i(d)$ is generated by two components: a cluster effect, $\eta_s(d)$, which is common for objects in the same cluster and a random effect $\epsilon_i(d)$ which is universal for all objects. Both $\eta_s(d)$ and $\epsilon_i(d)$ are i.i.d. Gaussian variables with zero mean and variance one. s_i is an integer variable that controls which cluster the object belongs - these variables are analogous to spins of a Potts Model. Finally, each object's association to its cluster can be governed an by intra-cluster strength variable ($g_s > 0$)

In the context of asset sector discovery, the objects are assets, N represents the number of assets in the dataset, and D represents the number of observed days. However, in our case of market state discovery, the objects are days; hence N represents the number of days in the dataset, and D represents the observed stocks. s_i would govern the allocation of market days to market states; that is, market days that belong to the same market state ($s_i = s_j = s$) are correlated ($C_{i,j} \approx g_s / (1 + g_s)$) and days that are in different states ($s_i \neq s_j$) are independent.

Note that it is possible for each day to be assigned to a unique state (rightfully so if we do not know how many market states there are in the world given the best information at hand); hence, s_i can take all integer values up to N.

Tying the anzatz together, $\mathcal{S} = \{s_i\}_{i=1}^{N}$ is the set that describes the how market days are allocated to states and $\mathcal{G} \equiv \{g_s\}_{s=1}^{N}$ describes how strongly correlated they are to their states.

By introducing the Dirac-delta function, the joint likelihood of observing the data Ξ given a market state configuration $\mathcal{S}$ and parameters $\mathcal{G}$ is:

$$P(\Xi \mid \mathcal{S}, \mathcal{G}) = \prod_{d=1}^{D} \left\langle \prod_{i=1}^{N} \delta \left(\xi_i(d) - \frac{\sqrt{g_{s_i}} \eta_{s_i}(d) + \epsilon_i(d)}{\sqrt{1 + g_{s_i}}} \right) \right\rangle. \tag{11}$$

This Equation can be manipulated further to derive:

$$P(\Xi \mid \mathcal{S}, \mathcal{G}) \propto e^{-DH\{\mathcal{S},\mathcal{G}\}}, \tag{12}$$

where

$$\begin{aligned} H\{\mathcal{S}, \mathcal{G}\} = & \frac{1}{2} \sum_{s} \left[(1 + g_s) \left(n_s - \frac{g_s c_s}{1 + g_s n_s} \right) \right. \\ & \left. - n_s \ln (1 + g_s) + \ln (1 + g_s n_s) \right]. \end{aligned} \tag{13}$$

Given a state sequence $\mathcal{S}$ the likelihood is maximised by:

$$\hat{g}_s = \frac{c_s - n_s}{n_s^2 - c_s}. \tag{14}$$

$$n_s = \sum_{i=1}^{N} \delta_{s_i,s}, \quad \text{and } c_s = \sum_{i,j=1}^{N} C_{i,j} \delta_{s_i,s} \delta_{s_j,s}, \tag{15}$$

where $n_s > 1$ and $\hat{g}_s = 0$ for $n_s \leq 1$. n_s refers to the number of correlated days belonging to a state s_i and c_s defines the internal correlation of the state.

Furthermore the corresponding log-likelihood is:

$$H_c\{\mathcal{S}\} = \frac{1}{2} \sum_{s:n_s>0} \left[\log \frac{c_s}{n_s} + (n_s - 1) \log \frac{n_s^2 - c_s}{n_s^2 - n_s} \right], \tag{16}$$

and can be thought of a Potts Hamiltonian (Blatt et al. 1996) where the ground state $\mathcal{S}_0$ of H_c provides the maximum likelihood fit with the anzats in Equation 10

Recall that what makes this approach attractive is that it makes no assumptions on the distribution of the data. Referring to the likelihood above (16) it requires no parameter, only the correlations c_s which comes from the data. Another subtle advantage is that the dimensionality of the data does not affect the likelihood directly, only through computation of C_{ij} which can be optimised in a data preparation step, for example.

Another way to think of the likelihood function is that it provides a measure of similarity between objects. Hence, from the perspective of hierarchical clustering, we can use the cost function (H_c) to find the optimal market state structure.

To optimise H_c, Giada et al. (2001) propose the "Merging-Algorithm", the algorithm is implemented in the following steps:

1. Start from N clusters composed of one object, e.g., $s_i = i \forall i = 1, \ldots, N$.
2. Merge two cluster into a single one such that the likelihood H_c is maxmised.
3. Repeat step 1 N-1 times until the configuration with a single cluster is reached.

A dendrogram can represent the output of this algorithm.

3.3.1.2 ASPC

As mentioned above, ASPC (Yelibi et al. 2019) is very much based on the Giada et al. (2001) algorithm and proposes improvements to the Merging Algorithm. Similarly, ASPC starts with N clusters of one object and iteratively merges clusters in a "greedy fashion". Rather than considering every single merging combination at each iteration, Equation 16 is adapted to evaluate a single cluster

$$H_c\{\mathcal{S}\} = \frac{1}{2} \left[\log \frac{c_s}{n_s} + (n_s - 1) \log \frac{n_s^2 - c_s}{n_s^2 - n_s} \right]. \tag{17}$$

Consider the example of three clusters, C_1, C_2, $C_3 = C_1 + C_2$ where "+" means that C_1 and C_2 are merged. Merging cluster C_3 is accepted if it provides a better likelihood than their combination of the individual sub-clusters

1. **Acceptance case**:

$$\Delta H_c = H_c\left(C_3\right) - \left[H_c\left(C_1\right) + L_c\left(C_2\right)\right]. \tag{18}$$

This approach is inspired by the "Louvain algorithm" which provides a bottom-up agglomerate approach to evaluate potential clusters quickly. Combining the modified Giada-Marsili algorithm with the acceptance case above gives rise to the new Agglomerative Super-Paramagnetic Clustering algorithm. The detailed pseudocode is presented below

Algorithm 4: ASPC algorithm

```
Input ;
Correlation Matrix ;
Output ;
Tracker matrix;
Initialize ;
1st pass ; Produce an initial population of N individuals
for  i in N-1 do
    for j in i+1:N do
        merge labels i and j, store ΔH_c as candidates
    ;
; Main ;
2st pass ;
for  N-1 interations do
    Select candidates the two merged (root, and leaf) labels with max ΔH_c Stop if
     ΔH_c ≤ 0 Update S, and tracker ;
    for j in labels not root do
        merge labels root and i, store ΔH_c as candidates;
    ;
```

3.4 Data II: Simulated Market Data

The Noh (2000) anzats also provides a useful method to simulating datasets (Yelibi et al. 2019). Recall the anzats (Equation 10) assumes that the cluster effect and measurement error variables are i.i.d. Gaussian, hence, we can draw values from these distributions and generate time series of stocks with correlated market states. Simulating data from the Noh anzats is completed in the following steps:

1. **Hyper-parameters**: Specify the hyper-parameters that will govern the simulation. That is, the number of states C, the number of days N, the number of stocks D and the intra-binding strength g_s.
2. **State sequence** Simulate a random state-sequence $\mathcal{S}$.
3. **Random effects**: Create a CxN matrix with $\epsilon \sim \mathcal{N}(0, 1)$ as entries.
4. **Market state effects**: Create a CxN matrix of $\eta \sim \mathcal{N}(0, 1)$.
5. **Compute returns**: Create a NxD matrix and compute the daily returns $\xi(d)$ as specified by Equation 10.

A critical input is the state configurations ($\mathcal{S}$) which allocate days to states. While the original text (Yelibi et al. 2019) does not specify how the state sequences should be generated, we make use of a Hidden Markov Model to learn from real data and simulate Markovian state sequences.

Hidden Markov Models (Zucchini et al. 2016) in the context of statistical finance are time processes whereby an unobservable Markov-Chain governs the observation of financial log-returns. In line with the anzats' notation, the emergent returns $\xi_i(d)$ are governed by market state sequences s_i. By assuming the set $\mathcal{S}$ is a Markov process, we can use the S&P 500 dataset to learn a transition probability matrix and simulate state sequences as an input to the Noh simulation model. Figure 18 is a 2-state example of the discovered state sequences from a dataset of S&P 500 stocks between February 2013 and February 2018 in daily granularity. Table 1 shows the estimated state-transition probability matrix, which will serve as the basis for simulating state-sequences for this project.

Figure 32 shows the results of simulating a 2-state market consisting of 100 stocks over five years for varying values of intra-cluster strength. This is the generic standard Gaussian scenario. We make a slight adaptation to the Random Effects and Market State Effects for our tests to make the data more suited to the problem we aim to solve. Firstly, for the random effects, we change the variance parameter to 0.01; this is informed by the overall market variance observed in the S&P market data, which is approximately 0.008. Secondly, we adapt the parameters of the State Effect using the estimates identified in the

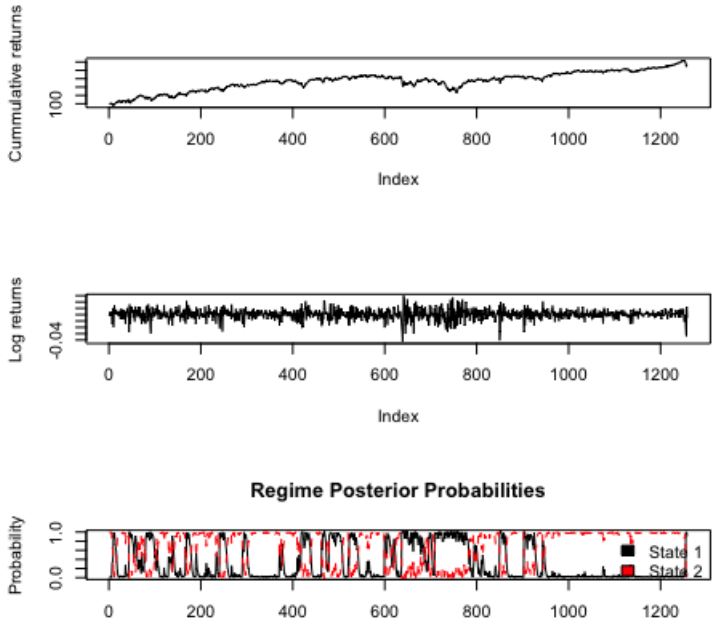


Figure 18: Graphical illustration of a fitted 2-state Hidden Markov Model on S&P 500 data as discussed in "Data II: Simulated Market Data" (Section 3.4). This fitted models enables us to extract a probability transition matrix for a 2-state market (Table 1) which are then used to simulate market state-sequences and used as an input into the Noh market simulation.

	To State 1	**To State 2**
From State 1	0.93	0.07
From State 2	0.04	0.96

Table 1: Probability transition matrix of a fitted 2-state Hidden Markov Model on S&P 500 data.

Hidden Markov Model. When searching for two states using the S&P 500 dataset, the mean and standard deviation for each state is $\mu_1 = -0.0004$,$\sigma_1 = 0.012$ and $\mu_2 = 0.0007$,$\sigma_2 = 0.005$ respectively. We do this change because we want to simulate states that are differentiated from each other that the ICC and ASPC models are able to identify. Figure 33 illustrates examples of simulated datasets under this adapted version

$$\epsilon \sim \mathcal{N}(0, 0.01) \quad \eta_s \sim \mathcal{N}(\mu_s, \sigma_s). \tag{19}$$

3.5 Performance Measurement: Adjusted Rand Index

The final component of our methodology will be how we test the performance of the models. As a short recap, we will simulate state-switching market data, whereas we know the underlying state process. Then we will feed this simulated market data to the ICC and ASPC to identify the market states. Lastly, we will need to compare the estimated market states to the ground truth market states to assess performance. The measure we use for this is the Adjusted-Rand Index (ARI) (Rand 1971). This is a popular measure of similarity between two data clusters. Formally it is defined s follows:

Given a set of two clusters, $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$. The overlap between X and Y can be summarised in a contingency table where each entry denotes the number of objects common between X_i and $Y_i := n_{ij} =\mid X_i \cap Y_j \mid$

$\overset{Y}{X}$	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

Hence, the ARI is defined as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}, \tag{20}$$

where n_{ij}, a_i, b_j are values from the contingency table. The ARI ranges from 0 to 1 where 0 means no similarity between clusters X and Y and 1 means perfect similarity between X and Y.

3.6 Asset Allocation: Mean-Variance Portfolio Optimisation

3.6.1 Theory and model formulation

Markowitz revolutionised the way investors approached Asset Allocation. Although the concept of an investment portfolio tailored to meet a specific risk-return target feels trivial today, in the early 1900s, investing was purely focused on buying stocks at the "right price", also referred to as Fundamental Analysis. Benjamin Graham is an example of a fundamental value investor. In his 1934 book "Security Analysis" (Graham 1932), he encouraged the investment community to value companies based on their operations and identify mispricings between the true value of a company and listed market price. By implication, investors were only focused on maximising return with little to no consideration of the risk contribution of the basket of stocks. Markowitz (1952) (in his seminal paper Portfolio Selection) rejects the rule that an investor should only maximise discounted return as it suggests there is no diversified portfolio that is preferable to non-diversified portfolios. In contrast, diversification is an observed and sensible behaviour. For example, this investment rule implies that if an investor placed all their wealth in the asset with the greatest discounted value. If two or more assets had the same value, then any two of these are as good as any other, and we know this is not the case. Many assets can offer the same return but have to vary different risk characteristics. Markowitz then proposes the Mean-Variance rule, which argues that an investor ought to maximise returns (or meet their desired return) while minimising undesirable portfolio variation (or risk).

Single Period Mean-Variance Portfolio Optimisation

The main goal of Markowitz's Mean-Variance Portfolio Optimisation (MVPO) is to help investors invest their wealth into a portfolio that helps achieve their risk-reward preferences and performs better than an equally weighted portfolio. Consider an investment universe of n risky assets and an investor's investment preference governed by a vector of weights $\vec{\boldsymbol{\omega}}$. The expected asset returns is $\mathbb{E}[\vec{\boldsymbol{R}}]$ and the covariance matrix is $\boldsymbol{\Sigma}$ with components $\Sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$. It follows that the expected portfolio return is:

$$\mathbb{E}\left[R_P\right] = \vec{\boldsymbol{\omega}}' \mathbb{E}[\vec{\mathbf{R}}] = \sum_{i=1}^{n} \omega_i \mathbb{E}\left[R_i\right], \tag{21}$$

and the expected portfolio risk is:

$$\sigma_P^2 = \vec{\boldsymbol{\omega}}' \boldsymbol{\Sigma} \vec{\boldsymbol{\omega}} = \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j \Sigma_{\bar{i}} = \sum_{i=1}^{n} \omega_i^2 \sigma_i^2 + 2 \sum_{\substack{j=1 \\ j=1+1}}^{n-1} \omega_i \omega_j \sigma_i \sigma_j \rho_{ij}. \tag{22}$$

As per Markowitz (1952) paper, an investor should aim to minimise the portfolio risk for a desired amount of risk. This objective can be modelled as follows

Minimise

$$\vec{\omega}'\mathbf{\Sigma}\vec{\omega}, \tag{23}$$

subject to

$$\vec{\omega}'\mathbb{E}[\vec{\mathbf{R}}] = \mathbb{E}\left[R_P\right], \tag{24}$$

and

$$\vec{\omega}'\mathbf{1} = 1. \tag{25}$$

The two constraints ensures the investors target return is met and all of the investors wealth is fully invested.

The model can be solved using a Lagrangian multiplier ($\vec{\boldsymbol{\lambda}}$):

$$L(\vec{\omega}, \vec{\boldsymbol{\lambda}}) = \frac{1}{2}\vec{\omega}'\mathbf{\Sigma}\vec{\omega} - \vec{\boldsymbol{\lambda}}' \begin{pmatrix} \vec{\omega}'\mathbb{E}[\vec{\mathbf{R}}] - \mathbb{E}\left[R_P\right] \\ \vec{\omega}'\vec{\mathbf{1}} - 1 \end{pmatrix}, \tag{26}$$

Note the $\frac{1}{2}$ has been added to simplify the algebra to follow. The optimal investment policy can be found by setting the first-order partial derivatives to zero:

$$\frac{\partial L}{\partial \vec{\omega}} = \mathbf{\Sigma}\vec{\omega} - \vec{\boldsymbol{\lambda}}' \begin{pmatrix} \mathbb{E}[\vec{\mathbf{R}}] \\ \vec{\mathbf{1}} \end{pmatrix} = \vec{\mathbf{0}}, \tag{27}$$

$$\frac{\partial L}{\partial \vec{\boldsymbol{\lambda}}} = \begin{pmatrix} \vec{\omega}'\mathbb{E}[\vec{\mathbf{R}}] - \mathbb{E}\left[R_P\right] \\ \vec{\omega}'\vec{\mathbf{1}} - 1 \end{pmatrix} = \vec{\mathbf{0}}. \tag{28}$$

From Equation 27, the optimal investment policy in terms of the Lagrangian multipliers is defined by:

$$\vec{\omega}^* = \mathbf{\Sigma}^{-1}(\lambda_1\mathbb{E}[\vec{\mathbf{R}}] + \lambda_2\vec{\mathbf{1}}). \tag{29}$$

Substituting $\vec{\omega}^*$ into the second partial derivative 28 produces the following equation:

$$\begin{pmatrix} \left[\boldsymbol{\Sigma}^{-1}(\lambda_1\mathbb{E}[\vec{\mathbf{R}}]+\lambda_2\vec{\mathbf{1}})\right]'\mathbb{E}[\vec{\mathbf{R}}] \\ \left[\boldsymbol{\Sigma}^{-1}(\lambda_1\mathbb{E}[\vec{\mathbf{R}}]+\lambda_2\vec{\mathbf{1}})\right]'\vec{\mathbf{1}} \end{pmatrix} = \begin{pmatrix} \mathbb{E}\left[R_P\right] \\ 1 \end{pmatrix}, \tag{30}$$

$$\begin{pmatrix} \lambda_1\mathbb{E}[\vec{\mathbf{R}}]'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}]+\lambda_2\vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}] \\ \lambda_1\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}]+\lambda_2\vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\vec{\mathbf{1}} \end{pmatrix} = \begin{pmatrix} \mathbb{E}\left[R_P\right] \\ 1 \end{pmatrix}, \tag{31}$$

$$\begin{pmatrix} \mathbb{E}[\vec{\mathbf{R}}]'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}] & \vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}] \\ \boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}] & \vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\vec{\mathbf{1}} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \mathbb{E}\left[R_P\right] \\ 1 \end{pmatrix}, \tag{32}$$

From this Equation we can define the following real numbers:

$$\begin{aligned} A = \vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}], \quad B = \mathbb{E}[\vec{\mathbf{R}}]'\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}], \\ C = \vec{\mathbf{1}}'\boldsymbol{\Sigma}^{-1}\vec{\mathbf{1}}, \qquad D = BC - A^2 \end{aligned} \tag{33}$$

By applying an inversion technique to the matrix pre-multiplied by $\vec{\boldsymbol{\lambda}}$, the optimal Lagrangian multipliers in terms of the investor's target return are:

$$\lambda_1^* = \frac{\mathbb{E}\left[R_P\right]C - A}{D}, \tag{34}$$

$$\lambda_2^* = \frac{B - \mathbb{E}\left[R_P\right]A}{D}. \tag{35}$$

Finally the optimal weights can be updated to produce:

$$\vec{\boldsymbol{\omega}}^* = \frac{1}{D}(B\boldsymbol{\Sigma}^{-1}\vec{\mathbf{1}} - A\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}]) + \frac{1}{D}\mathbb{E}\left[R_P\right](C\boldsymbol{\Sigma}^{-1}\mathbb{E}[\vec{\mathbf{R}}] - A\boldsymbol{\Sigma}^{-1}\vec{\mathbf{1}}). \tag{36}$$

This asset allocation method can be applied over multiple periods of time. Consider the example of an investor who wishes to invest their wealth every month. In the beginning of an investment period the investor can set their target portfolio return ($\mathbb{E}[R_P]$), estimate expected returns ($\mathbb{E}[\vec{\mathbf{R}}]$) and covariance matrix ($\boldsymbol{\Sigma}$) and solve for the optimal weights (Equation 36) to invest for the month. Next month, the investor can re-balance their portfolio to adhere to the optimal weights identified in the first investment period (Constant-Mix strategy) or hold for a defined period (Buy-Hold strategy).

State-dependent Mean-Variance Portfolio Optimisation

However, the asset allocation procedure does not account for the fact that the input parameters are not stationary, and the market structure changes over time. To illustrate this concept consider Figure 19. Three covariance matrices are computed, one for each market state of a two state market and another for the entire period. Visually there is a significant difference between the two states. The covariance matrix for state 1 is more yellow, suggesting less market volatility. In contrast, in state 2, the matrix has more shades of green, suggesting more market volatility in those periods. Given the difference in market structures between the two states, it would be likely that the optimal investment decisions for each state would differ. The investment decision would be constant, unlike the covariance matrix for the full period (which has similar colouring to state 1).

We will implement ICC and ASCP to identify market states. Once identified, the daily observations will be allocated to their corresponding clusters and used to estimate $\mathcal{S}$ many pairs of input parameters; therefore, the optimisation changes to incorporate states:

Minimise

$$\vec{\boldsymbol{\omega}'}_{s_i} \boldsymbol{\Sigma}_{s_i} \vec{\boldsymbol{\omega}}_{s_i}, \tag{37}$$

subject to

$$\vec{\boldsymbol{\omega}}'_{s_i} \mathbb{E}[\vec{\mathbf{R}}_{s_i}] = \mathbb{E}\left[R_P\right], \tag{38}$$

and

$$\vec{\boldsymbol{\omega}}'_{s_i} \vec{\mathbf{1}} = 1, \tag{39}$$

where s_i represents a market state.

We will limit ourselves to in-sample testing where we use ICC and ASPC to estimate the two market states for the entire dataset. We then split states into a train-test dataset, estimate the optimal weights for both market states, run a Constant-Mix strategy, and compare the profitability of both against a strategy where market state switching is ignored. Figure 19 illustrates how the strategy will be implemented.

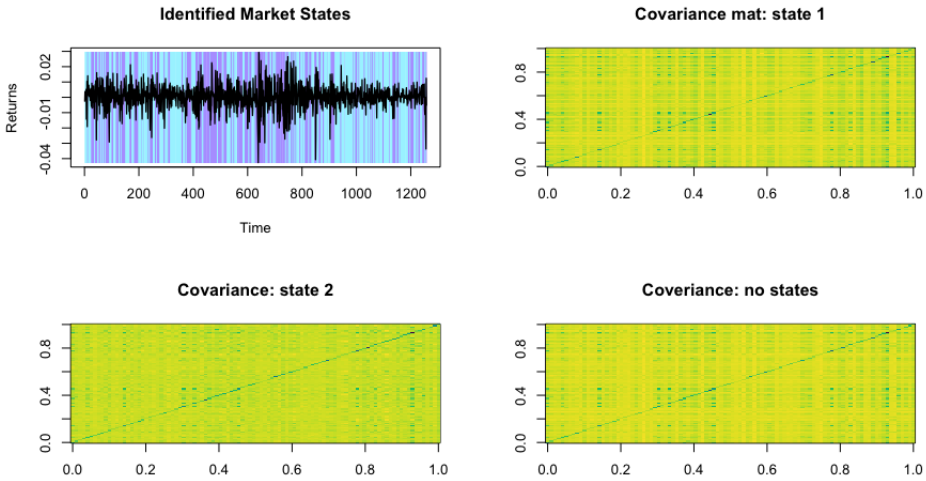


Figure 19: Conceptual illustration to explore state-dependent asset allocation. 2 market states on S&P 500 data are identified using ICC (the same market states discovered in the ICC Preliminary Replication Results (Section 3.2.2). Three covariance matrices are computed: one for each market state and another for the entire period. Visually there is a significant difference between the two states, The covariance matrix for state one is more yellow, suggesting less market volatility. In contrast, in state two, the matrix has more shades of green, suggesting more market volatility in those periods. Given the difference in market structures between the two states, it would be likely that the optimal investment decisions for each state would differ. Unlike the covariance matrix for the entire period (which has similar colouring to state 1), the investment decision would be constant.

4 Findings

The findings chapter is structured along two elements:

1. Model robustness: We test how each model performs under several simulated market state scenarios.
2. Real world application: We test which model produces a better outcome when applied to a state-dependant mean-variance asset allocation procedure.

4.1 Robustness to Simulated Scenarios

We created two test cases to assess each model's robustness: a base case and a stress case. The base case pertains to simulated datasets limited to two-state markets; the same dataset discussed in the Data II: Simulated Market Data section is applied to each model. We vary the ICS and number of dimensions to understand where the models begin to fail. The stress case aims to push the models to their boundaries, increasing market states from two to five.

4.1.1 Preliminary ICC Hyper-Parameter Grid Search

Recall that ICC utilises a hyper-parameter (γ) to manage the frequency of state switching and, in turn, the risk of overfitting the data. Increasing γ penalises the model from switching states; Hence, we need to run a preliminary grid search to identify which gamma value is appropriate for the simulated datasets.

Referring to Figure 20, ICC is applied to combinations of ICS (0.25, 0.5, 0.75 and 1) and γ (1, 10, 50 and 100). For each combination, we identify states and compute the ARI. The average index after 1000 simulations is presented on the y-axis. We observe that:

1. There is a slow decreasing trend between performance (measured in ARI) and γ. For γ values between 0 and 10, the performance slope is predominantly 0 for all combinations of ICS over this range of γ values. For γ values between 50 and 100, performance deteriorates significantly, reaching as low as 0 ARI for an ICS of 0.25.
2. There is a significant drop performance for ICS values of 0.25 and below, i.e., ARI values range between 0-0.3.

Given the slow decay in performance, we prefer to leave γ at 0 for the remainder of the chapter's experiments. This will help reduce the number of experiments as ICC model takes a long time to complete all iterations.

The second conclusion leads to a potential validation tool in identifying market states of real markets. With real datasets, the actual market states are unknown, and we need to look to other ways to validate whether the identified states are consistent with the actual state process. In practice, once states have been identified, one could estimate g_s (equation 14), if the value of g_s is in the region of 0.25, then there is enough reason to be concerned about the identified states' validity. This idea can be explored in future work.

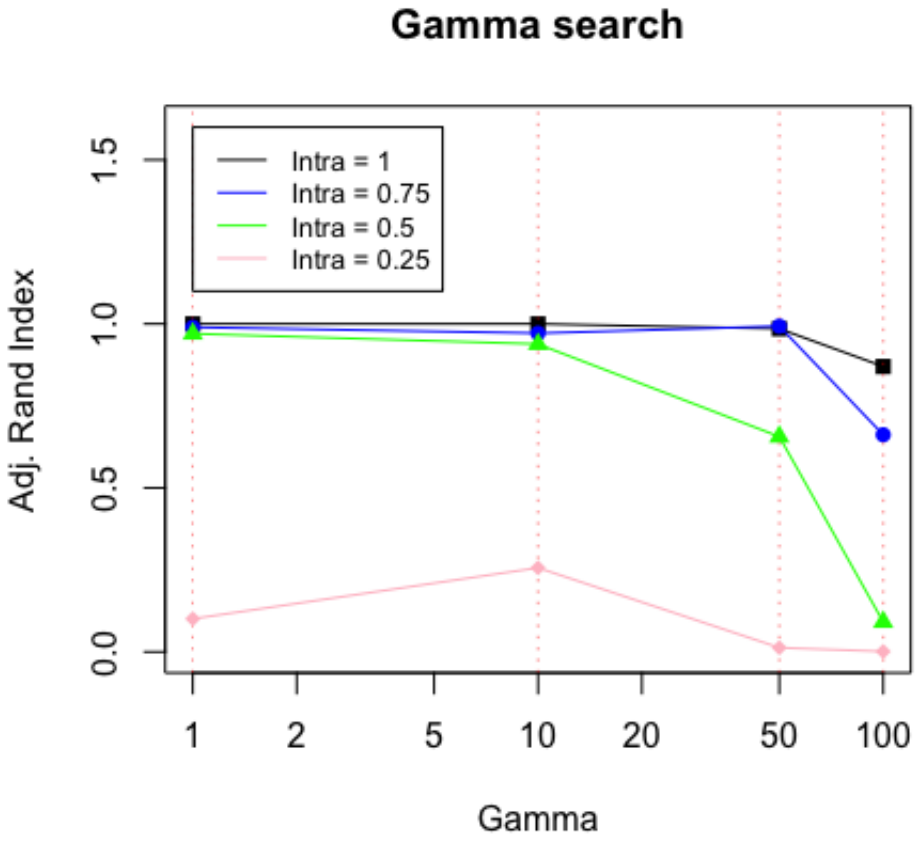


Figure 20: The Figure above illustrates the preliminary ICC hyper-parameter grid search (Section 4.1.1) to understand the ICC hyper-parameter's (γ)effects on performance (measured in Adjusted-Rand Index). Over a range of 0 to 100, the parameter, γ, has a negligible effect on performance. However, changes to the ICS drive a more significant effect on performance. Values in the region of 0.25 result in poor performance across all ranges of γ. For values 0.5 to 1, the performance trend is similar across the γ range. We have chosen to chose to keep gamma at 0 for all experiments.

4.1.2 Base Case (2-State Market Discovery)

The results for all combinations are presented in table 2 and 3 for ICC and ASPC respectively. Figures 21 and 22 show the performance results (measured in ARI) on the y-axis

for varying ICS scenarios on the x-axis.

In general, both models are able to perform reasonably well. There is a positive trend between ICS and ARI. In simplistic terms, the higher the ICS, the better we can expect both models to perform better for all stock dimensions.

Focusing on the ICC results in Figure 21, the model is able to recover nearly 100% of the market states sequences when ICS is 0.5 and larger. At 0.5 ICS for the 100-dimension case, the lowest average ARI result is 0.96 with a standard deviation of 0.13. As the number of dimensions increases, so does performance; In the 750-dimension case, the average ARI is 0.99. At ICS of 0.25, we observe a significant decline in performance and the impact of dimension changes becomes more clear. ARI ranges between 0.07 and 0.32 for 100-dimension to 750-dimension cases respectively. An interesting outcome is the non-effect dimensions have on performance at higher ICS, ICC is able to provide consistent performance in low dimensions (100-stocks) as well as high dimensions (750-dimension). This fact is in-line with the authors view that the model performs well in high-dimensional scenarios albeit under a simplistic 2-state market. Whether this remains the case shall be explored in the stress-case.

Focusing on ASPC results in Figure 22, the model is also able to perform reasonably well. Similar to ICC, there is also positive trend between ICS and ARI however, the number of dimensions has a larger effect on performance, the lower the dimensions the lower performance and the opposite is also observed. Consider the 100-dimension example at an ICS of 0.25, the ARI is 0.24 and the 750-dimension ARI is 0.85. Therefore, ASPC performs remarkably well when more dimensions are introduced even for extremely noisy simulations

In summary, both models are good at recovering 2-state markets. ICC will struggle more than ASPC with noisy datasets (or low ICS values) and adding more dimensions in noisy markets will not help improve the performance. This further validates the need to use g_s to help validate the quality of states discovered in real-world scenarios as discussed in the gamma grid-search section. ASPC is preferred as it performs well even with noisy datasets and performance can be improved by adding more dimensions into the dataset.

4.1.3 Stress Case (Multiple State Market Discovery)

The results for all combinations are presented in tables 2 and 3 for ICC and ASPC respectively. Figures 23 to 28 show stress cases from 3-state to 5-state markets.

Increasing the number of states tells a remarkably different story. ICC's performance tends to deteriorate quickly as the number of states increase. Considering the least noisy datasets where ICS is 1, ARI ranges between 1 and 0 across all state and dimension scenarios. In the more noisy datasets where ICS is 0.25 the highest average ARI is 0.02 across

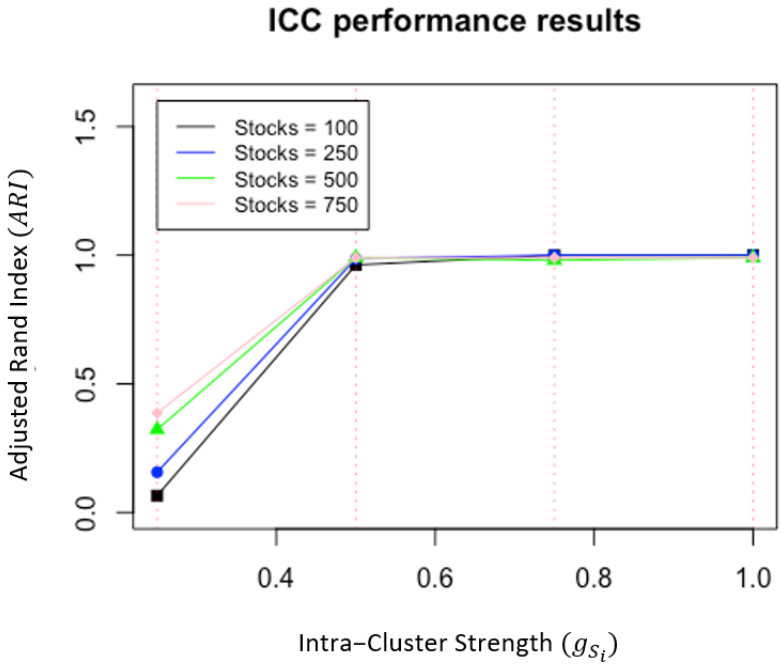


Figure 21: ICC Base Case (2-State Market) Results (Section 4.1.2): ICC's ability to recover hidden states of a simulated 2-state market (parameterised by S&P 500 market data) summarised in Adjusted Rand Index scores. Four intra-cluster strength (g_{s_i}) cases are considered: 0.25, 0.5, 0.75 and 1. Four dimension cases are considered: 100, 250, 500 and 750 stock sizes. For 2-state market, ARI along dimension changes is similar across all ICS values. A small ARI improvement for 250 and 500 stock size scenarios is observed. ICS has the strongest effect on performance, with almost perfect recovery from 0.5 to 1.

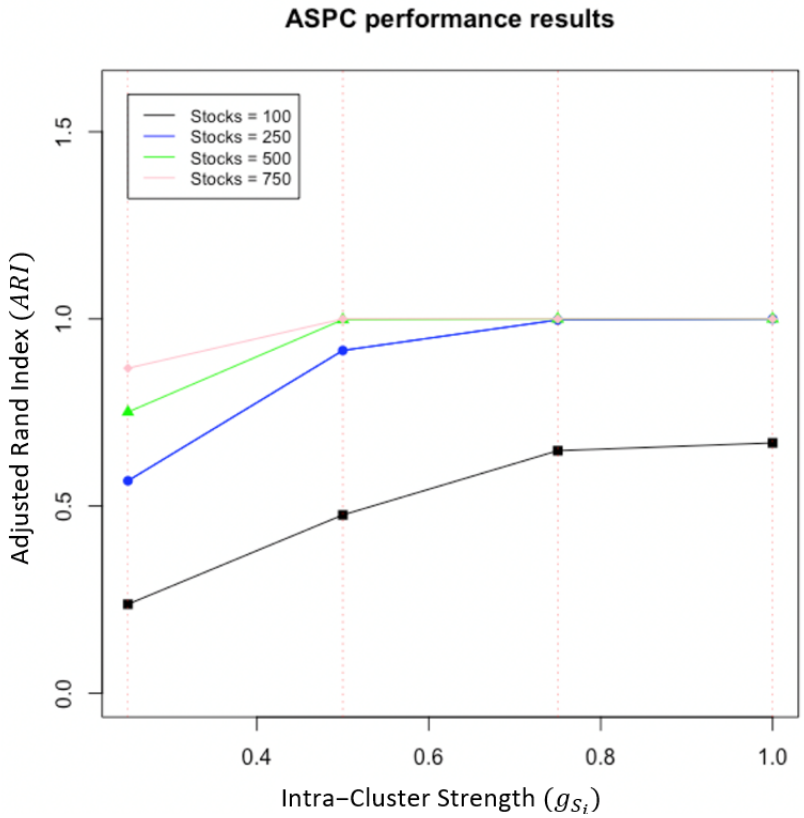


Figure 22: ASPC Base Case (2-State Market) Results (Section 4.1.2): ASPC's ability to recover hidden states of a simulated 2-state market (parameterised by S&P 500 data) summarised in Adjusted Rand Index scores. Four intra-cluster strength (g_{s_i}) cases are considered: 0.25, 0.5, 0.75 and 1. Four dimension cases are considered: 100, 250, 500 and 750 stock sizes. Similar to ICC (Figure 21), intra-cluster strength (g_{s_i}) has the strongest influence on performance. For values larger than 0.25, ARI improves significantly. In contrast, the dimension size plays a larger role as performance is maximised for datasets with larger stock sizes.

Intra-Cluster Strength(g_s)	0.25	0.5	0.75	1
100 Dimensions				
2-State Mean ARI (Std Dev.)	0.07 (0.14)	0.96 (0.13)	1.00 (0.002)	1.00 (0.001)
3-State Mean ARI (Std Dev.)	0.02 (0.01)	0.20 (0.16)	0.96 (0.03)	0.69 (0.51)
4-State Mean ARI (Std Dev.)	0.01 (0.003)	0.16 (0.03)	0.27 (0.02)	0.36 (0.02)
5-State Mean ARI (Std Dev.)	0.00 (0.002)	0.05 (0.01)	0.19 (0.14)	0.13 (0.03)
250 Dimensions				
2-State Mean ARI (Std Dev.)	0.16 (0.27)	0.99 (0.11)	1.00 (0.001)	1.00 (0.001)
3-State Mean ARI (Std Dev.)	0.03 (0.01)	0.10 (0.11)	1.00 (0.001)	0.77 (0.40)
4-State Mean ARI (Std Dev.)	0.02 (0.02)	0.13 (0.12)	0.19 (0.09)	0.22 (0.04)
5-State Mean ARI (Std Dev.)	0.02 (0.01)	0.02 (0.02)	0.05 (0.08)	0.06 (0.03)
500 Dimensions				
2-State Mean ARI (Std Dev.)	0.32 (0.42)	0.99 (0.10)	0.98 (0.14)	0.99 (0.01)
3-State Mean ARI (Std Dev.)	0.04 (0.03)	0.37 (0.54)	0.78 (0.39)	0.76 (0.41)
4-State Mean ARI (Std Dev.)	0.07 (0.12)	0.08 (0.11)	0.66 (0.05)	0.42 (0.26)
5-State Mean ARI (Std Dev.)	0.01 (0.01)	0.02 (0.04)	0.08 (0.001)	0.03 (0.03)
750 Dimensions				
2-State Mean ARI (Std Dev.)	0.39 (0.45)	0.99 (0.10)	0.99 (0.10)	0.99 (0.10)
3-State Mean ARI (Std Dev.)	0.02 (0.01)	0.13 (0.10)	0.61 (0.47)	1.00 (0.001)
4-State Mean ARI (Std Dev.)	0.02 (0.03)	0.18 (0.11)	0.21 (0.13)	0.30 (0.13)
5-State Mean ARI (Std Dev.)	.005 (.003)	.002 (.004)	.001 (.005)	.001 (.004)

Table 2: ICC Experiment Results (Section 4.1): For combinations of dimensions and intra-cluster strength (g_{s_i}) 100 simulated datasets are generated. ICC is used to recover the underlying states and the average Adjusted Rand Index (ARI) statistics are computed and presented in the table above. A key finding is that dimensionality does not affect the model's ability to recover the underlying state process. However, the time performance deteriorates exponentially for large datasets. For example the 100 Dimension experiments complete within 1 hour, the 750 dimension experiments take a 3-4 days to complete. Parallelisation does improve the computation slightly but there is cause to investigate how computation times can be improved.

Intra-Cluster Strength (g_{s_i}) (g_s)	0.25	0.5	0.75	1
100 Dimensions				
2-State Mean ARI (Std Dev.)	0.24 (0.04)	0.48 (0.10)	0.65 (0.20)	0.68 (0.20)
3-State Mean ARI (Std Dev.)	0.08 (0.05)	0.36 (0.12)	0.59 (0.24)	0.60 (0.21)
4-State Mean ARI (Std Dev.)	0.07 (0.02)	0.29 (0.05)	0.40 (0.08)	0.48 (0.08)
5-State Mean ARI (Std Dev.)	0.07 (0.02)	0.26 (0.05)	0.42 (0.08)	0.48 (0.08)
250 Dimensions				
2-State Mean ARI (Std Dev.)	0.58 (0.04)	0.89 (0.10)	1.00 (0.01)	1.00 (0.01)
3-State Mean ARI (Std Dev.)	0.19 (0.06)	0.63 (0.16)	0.89 (0.04)	0.95 (0.03)
4-State Mean ARI (Std Dev.)	0.13 (0.04)	0.41 (0.10)	0.66 (0.08)	0.81 (0.04)
5-State Mean ARI (Std Dev.)	0.19 (0.04)	0.36 (0.05)	0.47 (0.06)	0.59 (0.03)
500 Dimensions				
2-State Mean ARI (Std Dev.)	0.76 (0.04)	1.00 (0.01)	1.00 (0.01)	1.00 (0.01)
3-State Mean ARI (Std Dev.)	0.29 (0.10)	0.92 (0.04)	0.99 (0.01)	1.00 (0.00)
4-State Mean ARI (Std Dev.)	0.18 (0.07)	0.65 (0.06)	0.86 (0.02)	0.95 (0.02)
5-State Mean ARI (Std Dev.)	0.28 (0.04)	0.44 (0.03)	0.68 (0.04)	0.77 (0.03)
750 Dimensions				
2-State Mean ARI (Std Dev.)	0.85 (0.04)	1.00 (0.01)	1.00 (0.01)	1.00 (0.01)
3-State Mean ARI (Std Dev.)	0.34 (0.13)	0.98 (0.02)	1.00 (0.00)	1.00 (0.00)
4-State Mean ARI (Std Dev.)	0.26 (0.08)	0.77 (0.04)	0.94 (0.03)	0.98 (0.01)
5-State Mean ARI (Std Dev.)	0.31 (0.04)	0.49 (0.08)	0.76 (0.02)	0.83 (0.04)

Table 3: ASPC Experiment Results (Section 4.1): For combinations of dimensions and intra-cluster strength (g_{s_i}), 1000 simulated datasets are generated. ASPC is used to recover the underlying states and the mean Adjusted Rand Index (ARI) and standard deviation statistics are computed and presented in the table above. A key finding is that dimensionality helps improve model performance. The more dimensions in a dataset, the higher the performance for a given ICS and State value. For example, consider the column with ICS set to a value of 1, for each of the state simulations the ARI is increasing as the number of dimensions increase. As expected, a more significant performance driver is the ICS. For noisy datasets simulated by an ICS value of 0.25, the maximum possible ARI is 0.31. This is a useful finding which can be utilised to assess the quality of estimated states of real datasets where no ground truth state process is available.

all dimensions and states. This highlights how inconsistent and variable ICC performance is under the stress cases. ASPC's performance also deteriorates relative to its base case but not as severely as ICC. In the base case, maximum performance can be reached from ICS values of 0.5 and higher. In the stress case performance improvement is delayed and we start observing higher ARI values at ICS of 0.75 and above. Detailed discussions on how each model performs at the different market-state figures is presented below.

4.1.3.1 3-State Market Results

Figures 23 and 24 illustrates the performance results for a 3-state market simulation.

Focusing on ICC, recall that the base case performance trend in Figure 21 rapidly increases and maximises (ARI = 1) at an ICS of 0.5. In the 3-state market, it is clear that ICC struggles to deliver the same performance as the number of states are increased. For example, consider the 100-dimension scenario at ICS of 0.5; in the base case average ARI was 0.96. In the 3-state market, performance has deteriorated by up to 79% with an average ARI of 0.2. Similarly, in the 750-dimension scenario at the same ICS of 0.5, the base case produced an average ARI of 0.99, and the 3-state market an ARI of 0.13. Referring to Figure 23, we also observe a peculiar trend. In general, for all dimensions, performance increases as ICS increases; however the relationship between a number of dimensions and performance is lost. For example, at ICS of 0.75, the 750-dimension scenario results in the weakest performance, whereas at ICS of 1, the opposite occurs, 750-dimension scenario has the strongest performance. To add to this peculiarity, we note a kink at an ICS of 0.75 where performance is maximised for the 250-dimension and 500-dimension scenarios and then declines when ICS is 1. Indeed, these observations were unexpected and after re-running the experiments multiple times, we recovered similar results. Perhaps in future work, we could increase the number of ICS experiments and not limit to the four cases (0.25, 0.5, 0.75, 1) but again, the ICC is not an efficient model and takes a significant time to iterate through all the scenarios. For the purpose of this experiment, we accept the results as-is and note that ICC provides questionable results at higher market states and should be explored further in future research.

Focusing on ASPC, similar performance trends to the base case can be observed. Firstly, we observe a positive increasing trend in performance (ARI) and ICS for all dimension scenarios. Secondly, performance improves as the the number of dimensions are increased, the 750-dimension scenarios provides the best result reaching average ARI of 1 from an ICS of 0.25. Unlike the ICC, we do not observe any non-intuitive trends, as ICS increases so does the ARI, maximum performance (ARI=1) for all dimension scenarios is reached at ISC of 1.

Despite similarity in trends, we observe a slight deterioration in performance relative to

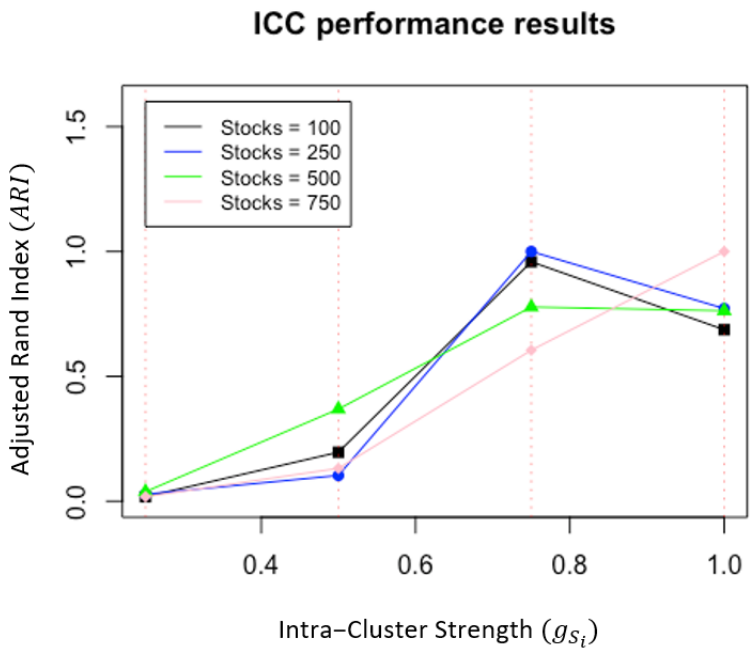


Figure 23: ICC Stress Case (3-State) Market Results (Section 4.1.3.1): ICC's ability to recover hidden states of a 3-State market summarised in Adjusted Rand Index scores. The ARI trend is rather unexpected for two reasons: first, the performance trend peaks at an intra-cluster strength (g_{s_i}) of 0.75; secondly, the effect of the number of stocks is unclear as the 100 to 250 stock datasets are the weakest performers at 0.25 intra-cluster strength (g_{s_i}) and the strongest performers at 0.75 intra-cluster strength (g_{s_i}).

the base case in the 100-dimension and 250-dimensions. The average minimum and maximum ARI range in the base case was between 0.24-0.68 and 0.58-1 for the 100 stock and 250 stock scenarios respectively. In the 3-state market this performance has reduced to 0.08-0.60 and 0.19-0.95 the 100 stock and 250 stock scenarios respectively. We expect to see this gap increase in the higher state markets. Overall ASPC results are more intuitive and lead to better results when compared to ICC.

4.1.3.2 4-State Market Results

Figures 25 and 26 illustrates the performance results for a 4-state market simulation.

Focusing on ICC, the performance is below 0.5 for nearly all the combinations of experiments with the exception of what appears to be an outlier at 0.75 ICS and 500-stock dataset. ICC struggles to recover the underlying state processes for all ICS values. Even for the dataset with the least amount of noise, performance is poor and cannot be used to with confidence in real world applications. Similar to the 3-state datasets, we cannot discern a clear linear relationship between performance and the number of stocks.

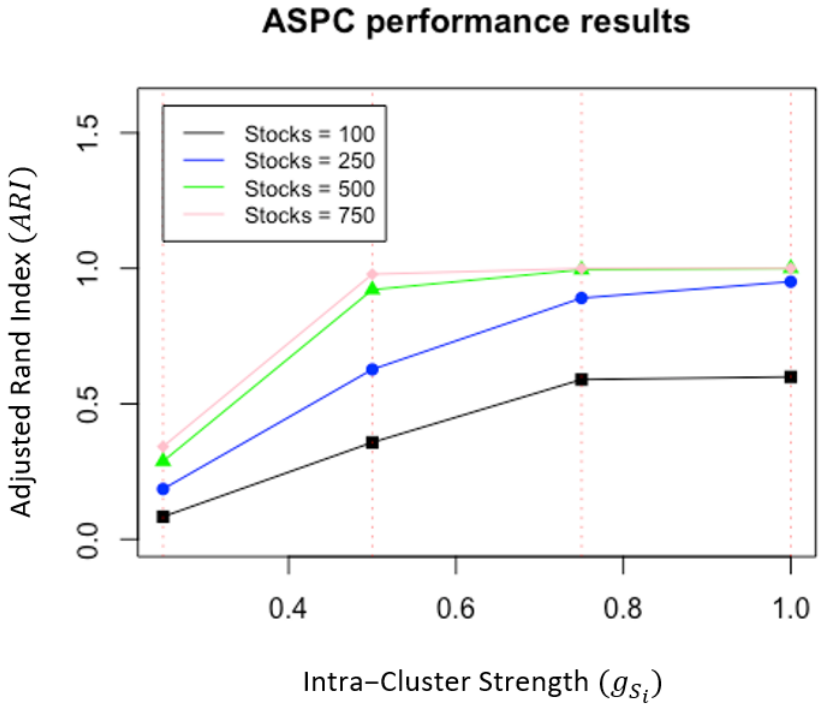


Figure 24: ASPC Stress Case (3-State Market) Results (Section 4.1.3.1): ASPC's ability to recover hidden states of a 3-State market summarised in Adjusted Rand Index scores. The ARI trend for ASPC is consistent with what was observed in the Base Case. Additionally the quality of performance is similar ranging between 0.3 and 1 for intra-cluster strength (g_{s_i}) values between 0.25 and 1. We can observe slight deterioration in performance for 250 stock size datasets relative to the Base Case (Figure 22) which recovers to perfect state discovery at an intra-cluster strength (g_{s_i}) value of 1.

Focusing on ASPC, the performance begins to deteriorate further even for higher dimension datasets. In the 3-state dataset, 750-stock datasets were able to performance well (>0.9 ARI) even in noisy scenarios (0.5 = ICS). In this case the performance is only maximised (ARI = 1) for the 750-stock datasets and ICS values of 1. As seen before, increasing the number of dimensions leads to better overall performance in the 4-state experiments.

4.1.3.3 5-State Market Results

Figures 27 and 28 illustrates the performance results for a 5-state market simulation.

Focusing on ICC, performance has completely deteriorated for all combinations of experiments. Considering the lowest ICS value, ARI is 0 for all stock dimension datasets. In other words ICC is unable to allocate a single day to the correct state. Consider the highest ICS value, ARI is still significantly small, ranging from [0.001 - 0.13] and again ICC is

unable to recover the state sequences. Regarding the relationship between stock dimensions and performance, we can observe a small trend where the smaller stock dimensions performs better than the higher stock dimensions. Again, it is not clear why this pattern exists and should be a topic of further exploration in future research.

Focusing on ASPC, performance trends are similar and consistent to what was observed in the other stress cases. Firstly, overall performance decreases for all combinations as the number of states increase. Secondly, the dataset with the highest number of dimensions performs the best for all ICS cases.

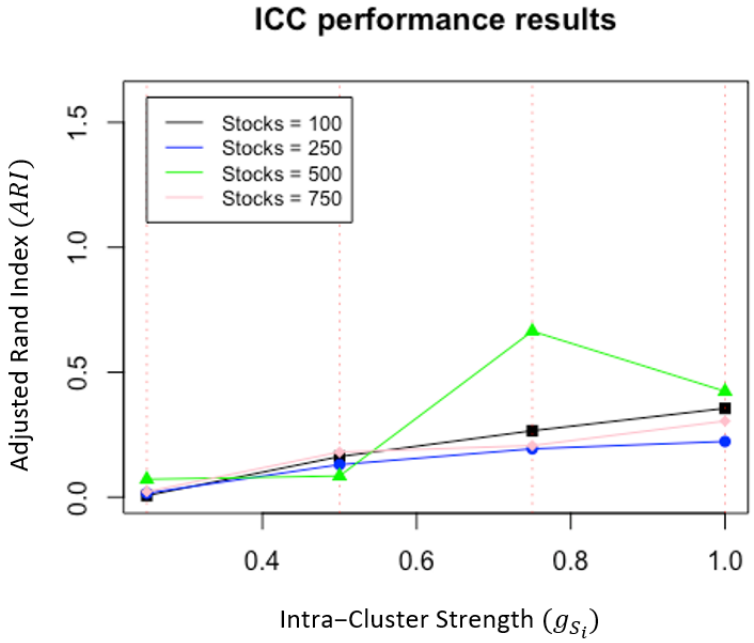


Figure 25: ICC Stress Case (4-State) Market Results (Section 4.1.3.2): ICC's ability to recover hidden states of a 4-State market summarised in Adjusted Rand Index scores. Again the ARI trend is rather peculiar with a kink observable at an intra-cluster strength (g_{s_i}) of 0.75 for the 500 stock case. A key insight is that ICC is unable to meet a minimum of 50% ARI performance in a 4 state market which raises concerns of the model's applicability to real dataset which are naturally more complex.

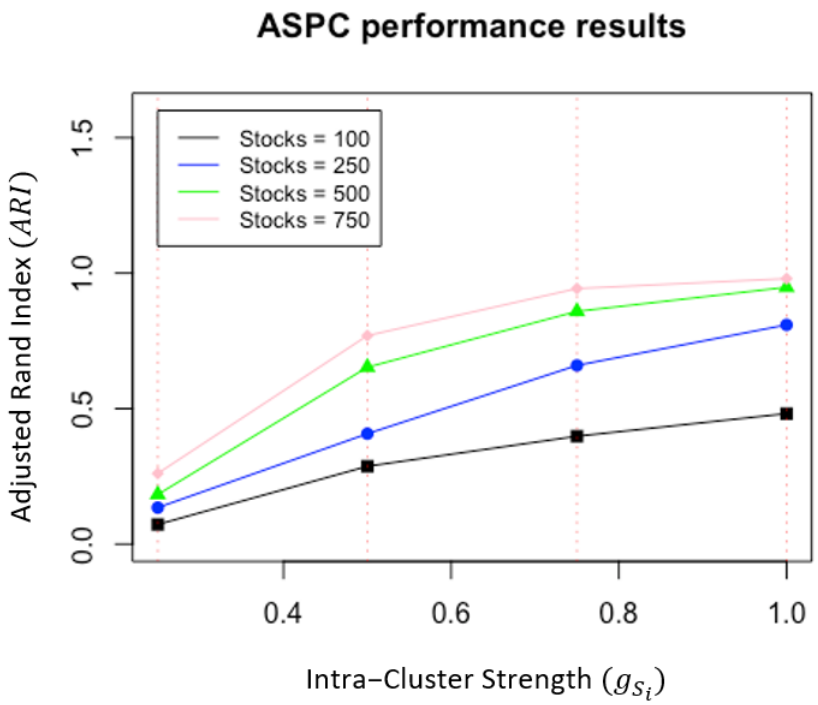


Figure 26: ASPC Stress Case (4-State Market) Results (Section 4.1.3.2): ASPC's ability to recover hidden states of a 4-State market summarised in Adjusted Rand Index scores. The ARI trend for ASPC remains consistent with what was observed in the Base Case and 3-State case (figures 22 and 24). We observe further deterioration in performance with the 100 and 250 stock datasets resulting in ARI values less than 50% for intra-cluster strength (g_{s_i}) values ranging between 0.25 - 0.75.

4.2 Real World Application: State-Dependent Asset Allocation

In practice, market state discovery is a pre-processing step when implementing an investment strategy. Hence we take the work a small step further to observe which of the two algorithms lead to better performance in a state-dependent asset allocation problem.

We have implemented ICC and ASPC to the full S&P 500 dataset. The states identified are presented in figures 29 and 30 for ASCP and ICC respectively. The identified states are complete opposites, ICC suggests frequent switching between states where ASPC sees 95% of the data as a single state (when forced do search for 2 states).

The identified market states are split into 70% training and 30% testing. The training set is used to find optimal investment weights for each state using the MVPO model and the testing set is what is used for the investment. In practice, one would need to use the identified market states to forecast the out of sample market states. For simplicity we assume that the out of sample states are known.

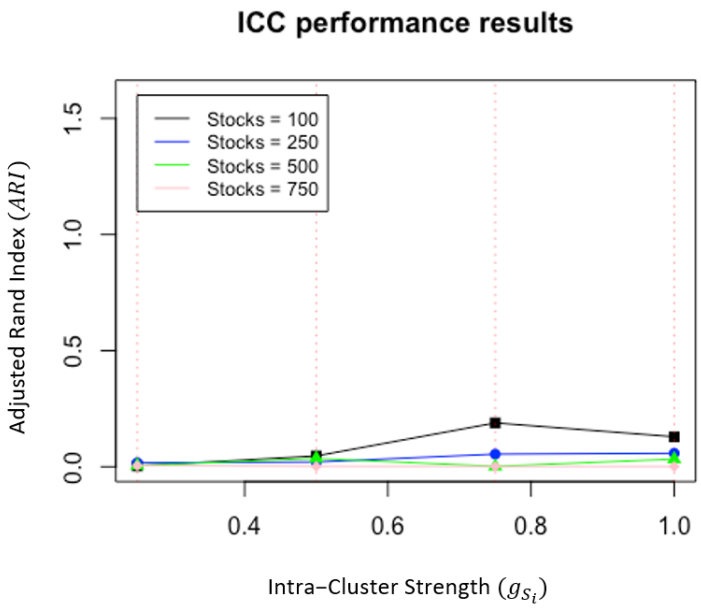


Figure 27: ICC Stress Case (5-State) Market Results (Section 4.1.3.3): ICC's ability to recover hidden states of a 5-State market summarised in Adjusted Rand Index scores. In the 5-state scenario, ICC performs poorly for all intra-cluster strength (g_{s_i}) values. Again we observe a slight kink at intra-cluster strength (g_{s_i}) of 0.75 but hardly at a performance level that could be useful.

Finally with the optimal market investment weights identified we invest using constant-mix strategy where the optimal weights are re-balanced every day, further more, if the states for a period change the corresponding investment weights are applied. Figure 31 show the cumulative wealth when using ICC, ASPC and a standard MVPO where a single market state is assumed. Given ASPC suggested minimal state-switching, the performance is similar to the standard MVPO. However, ASPC performs significantly better than the ICC MVPO algorithm. It may be possible to improve ICC's performance by increasing γ to reduce state-switching but that would require additional grid-searching to find a suitable penalisation number.

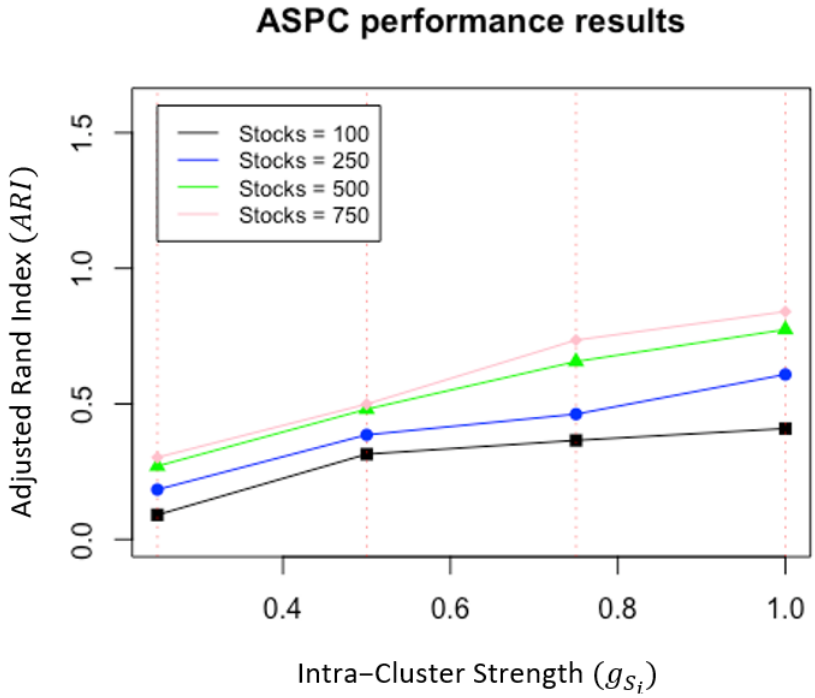


Figure 28: ASPC Stress Case (5-State) Market Results (Section 4.1.3.3): ASPC's ability to recover hidden states of a 5-State market summarised in Adjusted Rand Index scores. Again we observe a linear positive ARI trend. For all stock sizes we observe improving performance as the intra-cluster strength (g_{s_i}) increases. For stock sizes 100 and 250 the performance is below 50% suggesting that as the states increase the more stocks in the dataset the better the performance.

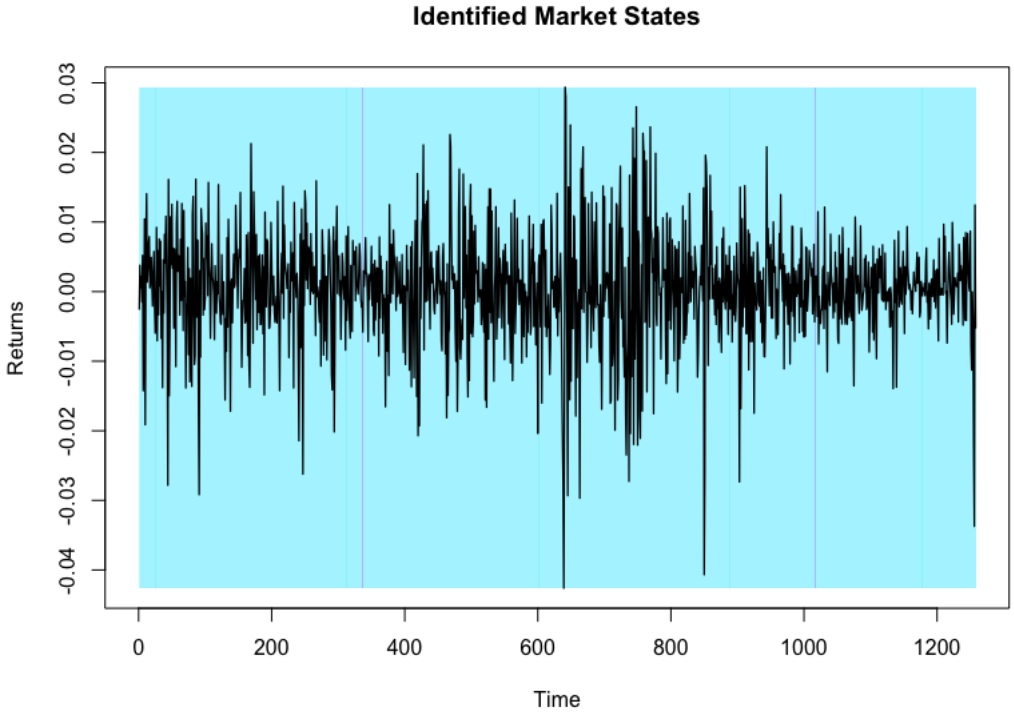


Figure 29: ASPC identified market states over of the S&P 100-stock sample dataset (Section 4.2). Here ASPC proposed 2 market states with almost no state switching excerpt for two low volatility time points.

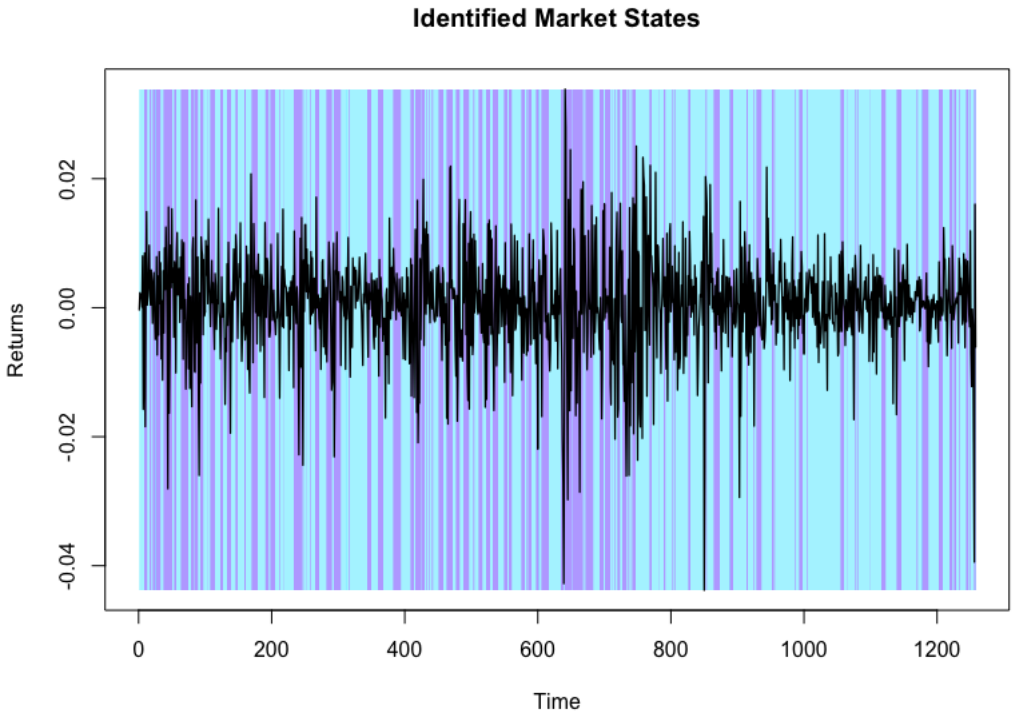


Figure 30: ICC identified market states over the entire period of the S&P 100-stock sample dataset (section 4.2). Here ICC was set to 2-states. The first state (purple) appears to be associated with high volatility time periods whilst the second is associated with low volatility time periods.

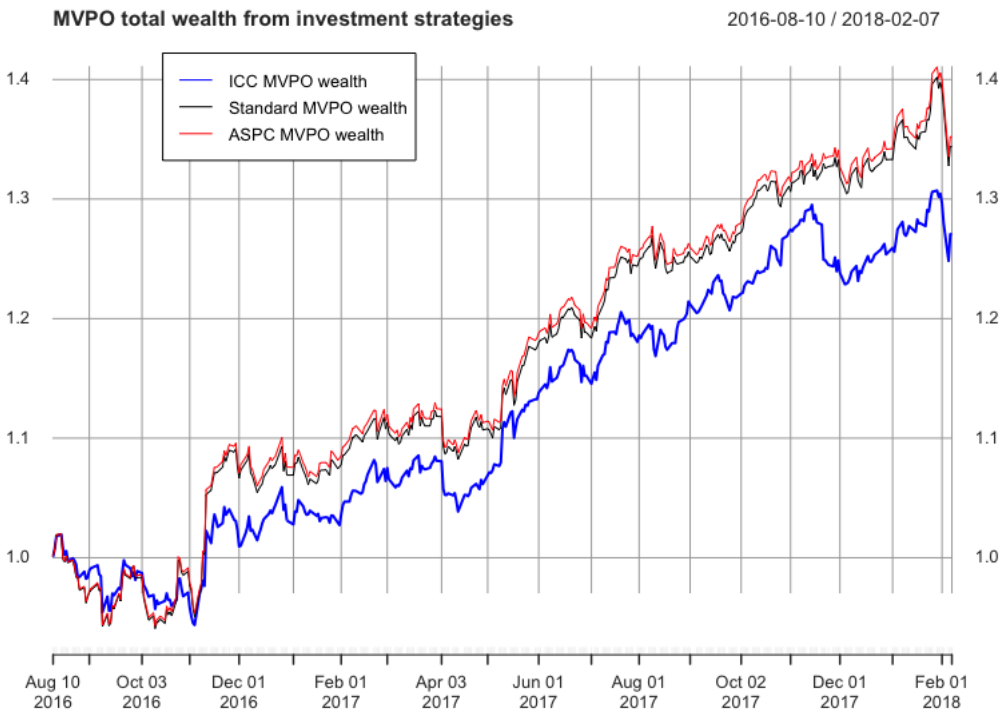


Figure 31: ASPC, ICC and Standard MVPO investment strategies compared. ASPC performs slightly better than the standard MVPO and significantly better than the ICC (Section 4.2).

5 Conclusion

The objective of this project was to explore the concept of market state discovery by assessing the robustness of two algorithms in estimating market states from simulated data, namely: Inverse Covariance Clustering (ICC) and Agglomerative Super-Paramagnetic Clustering (ASPC).

Our robustness assessment was implemented in the following steps:

1. Simulate market states using a Markov-state process parameterised by mean and variance estimates of real S&P 500 market data (see Section 3.4).
2. Simulate market log-returns using a Noh anzats model (Equation 10), which enables us to control the number of stocks (D) and intra-cluster strength[1] (g_{s_i}) (see Section 3.4).
3. Estimate market states from simulated log-returns by implementing ICC and ASPC (see sections 3.2 and 3.3).
4. Measure the algorithms' ability to recover the underlying market states from log-returns by using the Adjusted Rand-Index[2] (ARI) as a performance metric (introduced in Section 3.5).

The Noh anzats (see Equation 10) was instrumental in ensuring the simulated datasets had sufficient diversity to explore ICC and ASPC strengths and weaknesses. Simulated datasets varied in the number of states ($\mathcal{S}$), the number of stocks (D) and the intra-cluster strength (g_{s_i}). More specifically, experiments were structured according to two cases:

1. **The Base Case**: Simulations consisting of 2 market states, intra-cluster strength ranging (g_{s_i}) from 0.25 - 1 (in 0.25 increments) and the number of dimensions, precisely: 100, 250, 500 and 750 stocks (see findings in Section 4.1.2).
2. **The Stress Case**: Simulations consisting of 3, 4 and 5 market states, intra-cluster strength (g_{s_i}) ranging from (0.25 - 1, in 0.25 increments) and the number of dimensions, precisely: 100, 250, 500 and 750 stocks (see findings in Section 4.1.3).

[1] Intra-cluster strength is a parameter of the Noh anzants which controls the degree of association the log-returns of a stock has with the average log-return of the market state. It ranges from 0 to 1, where 0 is very weak association and 1 is strong association.

[2] ARI is a similarity index that measures the similarity between two clustering. The index ranges from 0 to 1, where 0 is no similarity and 1 is perfect similarity

The Base Case

ICC and ASPC performed remarkably well in the Base Case. The ARI numbers were larger than 0.7 in 70% and 68% of the simulations for ICC and ASPC respectively (see tables 2 and 3).

ICC was slightly more robust than ASPC when intra-cluster strength (g_{s_i}) was larger than and equal to 0.5. In Figure 21, the ICC ARI numbers remained within a stable range of 0.96 - 1 when the number of stocks changed. However, ASPC was more sensitive to changes in the number of stocks. For example, in Figure 22, 0.48 - 0.68 was the minimum-maximum ARI range for 100 stocks; this improved to 0.89 - 1 (approximately 40-80% better performance) when the number of stocks increased to 250.

ASPC was significantly superior to ICC when intra-cluster strength (g_{s_i}) was 0.25. In the 750 stocks experiment, ASPC and ICC's maximum ARI was 0.85 and 0.39 respectively. This pattern was also observed in a lower-dimensional setting (100 stocks), the ASPC and ICC ARI were 0.24 and 0.07, respectively. ASPC's sensitivity to the number of stocks turned out to be an advantage (see figures 21 and 22).

The Stress Case

For both algorithms, the ARI generally deteriorates as the number of states increases. The Stress Case showed that ICC's deterioration in performance is significantly more severe than ASPC.

Considering the 3-state market (see Section 4.1.3.1), ASCP showed similar trends to those observed in the Base Case (see figures 22 and 24). Increasing the number of stocks and intra-cluster strength (g_{s_i}) was consistently associated with higher performance (ARI). However, the ICC trends (in Figure 23) were counterintuitive. The positive linear relationship between the number of stocks and performance was no longer observable. For example, at 0.5 intra-cluster strength (g_{s_i}), the 750 stocks dataset yielded the lowest ARI relative to the 100, 250 and 500 stocks datasets. At 1 intra-cluster strength (g_{s_i}), the opposite occurs, the 750 stocks dataset yields the highest average ARI.

Considering the 5-state market (see Section 4.1.3.3), ICC's performance is entirely inadequate. Irrespective of the number of stocks and intra-cluster strength (g_{s_i}), the ICC minimum-maximum ARI numbers ranged from 0.001 - 0.19. ASCP was still able to produce stronger results with minimum-maximum ARI numbers ranging from 0.07 - 0.83 (compare figures 27 and 28).

ICC's computation times were an unexpected disadvantage discovered during the implementation. As the number of states and stocks increase, ICC would take 2-3 days to produce state estimates. In the worst-case, ASPC would take about 3-5 hours.

State-Dependent Asset Allocation

In addition to the robustness assessment, a simple mean-variance asset allocation procedure was implemented to see if there are advantages to state-dependent asset allocation using states discovered by ICC and ASCP.

ICC and ASCP were implemented on daily S&P 500 data to discover the optimal 2-state process (see figures 30 and 29). Estimated states were split into 70% training and 30% testing sets. The training set was used to find the optimal investment weights per state. The testing set was used to signal the prevailing out-of-sample market state and which investment weight is optimal for the period. For simplicity, we assumed to know the out-of-sample state process. In reality, a supervised learning algorithm is required to forecast the out-of-sample state process.

In Figure 31, ASPC resulted in a better investment outcome when compared to ICC. ICC also underperformed against the baseline non-state dependent mean-variance portfolio.

Acknowledging that this is a weak test, it highlights a key practical issue with implementing ICC. ICC requires tweaking of the state-switching hyper-parameter (γ) in order to find a good state process where ASCP does not. The fact that one needs to rely on this hyper-parameter to yield a good result increases room for error and further disadvantages ICC as a state discovery algorithm.

Concluding Remarks

In conclusion, based on the results of the study, ASCP is a more robust algorithm for state discovery than ICC. ICC's primary disadvantage is its inability to produce reliable estimates in 3, 4 and 5 state markets relative to ASCP. In addition, ICC heavily relies on the art of selecting good hyper-parameters[3]: the temporal switching parameter (γ) and the number of states (K). Therefore ICC's utility as a market state discovery algorithm is limited.

[3]referring to the γ and K found in the ICC objective function in Equation 4

References

Aste, T. (2016). *J-LoGo sparse inverse covariance*. URL: https://www.mathworks.com/matlabcentral/fileexchange/22026-safe-computation-of-logarithm-determinat-of-large-matrix.

Aste, T., Matteo, T. Di, and Hyde, S. (2005). "Complex networks on hyperbolic surfaces". In: *Physica A: Statistical Mechanics and its Applications*, pp. 20–26.

Authur, B. (1995). "Complexity in economic and financial markets". In: *Complexity*, pp. 20–25.

Baitinger, E. and Papenbrock, J. (2017). "Interconnectedness risk and active portfolio management: The information-theoretic perspective". In.

Barfuss, W., G. P. Massara, T. Di Matteo, and Aste, T. (2016). "Parsimonious modeling with information filtering networks". In: *Physical Review E*, pp. 1–17.

Beveridge, S. and Nelson (1981). "A New Approach to Decomposition of Economic Time Series into Permanent and Transitory Components with Particular Attention to Measurement of the 'Business Cycle". In: *Journal of Monetary Economics*, pp. 151–174.

Billio, M. et al. (2012). "Econometric measures of connectedness and systemic risk in the finance and insurance sectors". In: *Journal of Financial Economics*, pp. 535–559.

Blatt, M., Wiseman, S., and Domany, E. (1996). "Superparamagnetic clustering of data". In: *Physical Review Letters*.

Bollerslev, T., Chou., R. Y., and Kroner, K. F. (1992). "ARCH modelling in finance. A review of the theory and empirical evidence". In: *Journal of Econometrics*, pp. 5–59.

Campbell, John, Y., and Mankiw, N. Gregory (1987). "Permanent and Transitory Components in Macroeconomic Fluctuations". In: *American Economic Review Papers and Proceedings*, pp. 111–117.

Donnat, P., Marti, G., and Very, P. (2016). "Toward a generic representation of random variables for machine learning". In: *Pattern Recognition Letters*, pp. 24–31.

Durante, F. et al. (2015). "A portfolio diversification strategy via tail dependence measures". In: *Soft Methods for Data Science*.

Fiedor, P. (2014). "Networks in financial markets based on the mutual information rate". In: *Physical Review E*.

Giada, L. and Marsili, M. (2001). "Data clustering and noise undressing of correlation matrices". In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, pp. 1–8.

Graham, B. (1932). *Security analysis: principles and technique. 1st ed. New York: McGraw-Hill*.

Gunnar, C. and Facundo, M. (2010). "Characterization, Stability and Convergence of Hierarchical Clustering Methods". In: *Journal of Machine Learning Research*, pp. 1425–1470.

Hamilton, J. D. (1989). "A New Approach to the Economic Analysis of Non-stationary Time Series and the Business Cycle". In: *Econometrica*, pp. 357–384.

Hamilton, J. D. (2005). "Regime Switching Models". In.

Hendricks, D., Gebbie, T., and Wilcox, D. (2016). "Detecting intraday financial market states using temporal clustering". In: *Quantitative Finance*, pp. 1657–1678.

Kenett, D. Y. et al. (2010). "Dominating clasp of the financial sector revealed by partial correlation analysis of the stock market". In: *PLoS ONE 5*.

King, F. B. (1996). "Market and Industry Factors in Stock Price Behavior". In: *The Journal of Business*, pp. 139–190. URL: http://www.jstor.org/stable/2351742.

Kullmann, L., Kertesz, J., and Mantegna, R. (2000). "Identification of clusters of companies in stock indices via potts super-paramagnetic transitions". In: *Physica A: Statistical Mechanics and its Applications*, p. 287.

Lee, G. S. and Djauhari, M. A. (2017). "Multidimensional stock network analysis: An Escoufier's RV coefficient approach". In: *AIP Conference Proceedings*, pp. 550–555.

Lin, D. (2020). *Safe computation of logarithm-determinat of large matrix*. URL: https://www.mathworks.com/matlabcentral/fileexchange/22026-safe-computation-of-logarithm-determinat-of-large-matrix.

Mantegna, R. N. (1999). "Hierarchical structure in financial markets". In: *The European Physical Journal B-Condensed Matter and Complex Systems*, pp. 193–197.

Markowitz, H. (1952). "Portfolio Selection". In: *The Journal of Finance*, pp. 77–91.

Marti, G., Nielsen, F., and Donnat, P. (2016). "Optimal copula transport for clustering multivariate time series". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2379–2383.

Marti, G. et al. (2015). "A proposal of a methodological framework with experimental guidelines to investigate clustering stability on financial time series". In: *14th IEEE International Conference on Machine Learning and Applications*, pp. 32–37.

Marti, G. et al. (2017). "A Review of Two Decades of Correlations, Hierarchies, Networks and Clustering in Financial Markets". In.

Massara, G. P., Matteo, T. Di, and Aste, T. (2016). "Network filtering for big data: Triangulated maximally filtered graph". In: *Journal of Complex Networks*, pp. 161–178.

Noh, J. D. (May 2000). "Model for correlations in stock markets". In: *Phys. Rev. E* 61 (5), pp. 5981–5982.

Nugent, Cam (2018). *S & P 500 stock data*. Kaggle. URL: https://www.kaggle.com/camnugent/sandp500.

Procacci, P. F. and Aste, T. (2019). "Forecasting market states". In: *Quantitative Finance*, pp. 1491–1498.

Rand, W. M. (1971). "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical Association*, pp. 846–850.

Tumminello, M. et al. (2005). "A tool for filtering information in complex systems". In: *Proceedings of the National Academy of Sciences of the United States of America*.

Tumminello, M. et al. (2007). "Spanning trees and bootstrap reliability estimation in correlation-based networks". In: *International Journal of Bifurcation and Chaos*, pp. 2319–2329.

Viterbi, A. (1967). “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transaction on Information Theory*, pp. 260–269.

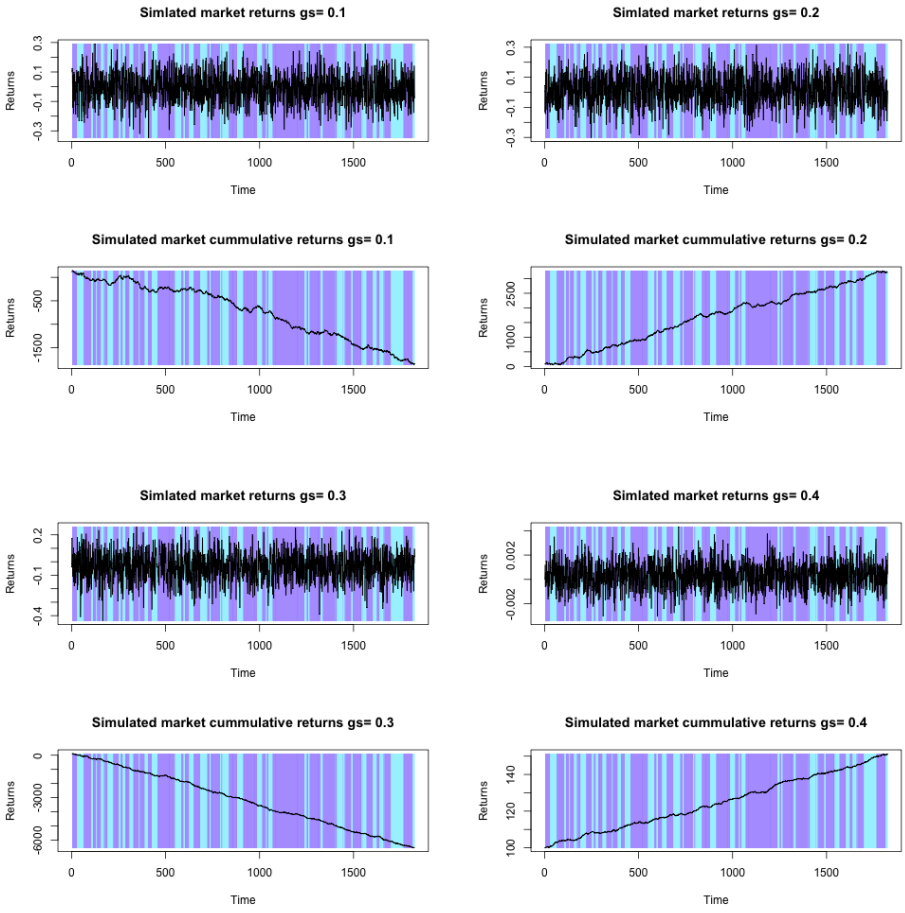
Wilcox, D. and Gebbie, T. (2014). “Hierarchical causality in financial economics”. In: *SSRN Electronic Journal*, pp. 1–16.

Yelibi, L. and Gebbie, T. (2019). “Agglomerative fast super-paramagnetic clustering”. In.

Zucchini, W., MacDonald, I. L., and Langrock, R. (2016). *Hidden Markov Models for Time Series, An introduction using R, Second Edition*. Chapman and Hall/CRC.

A Appendices

A.1 State-dependent simulated datasets



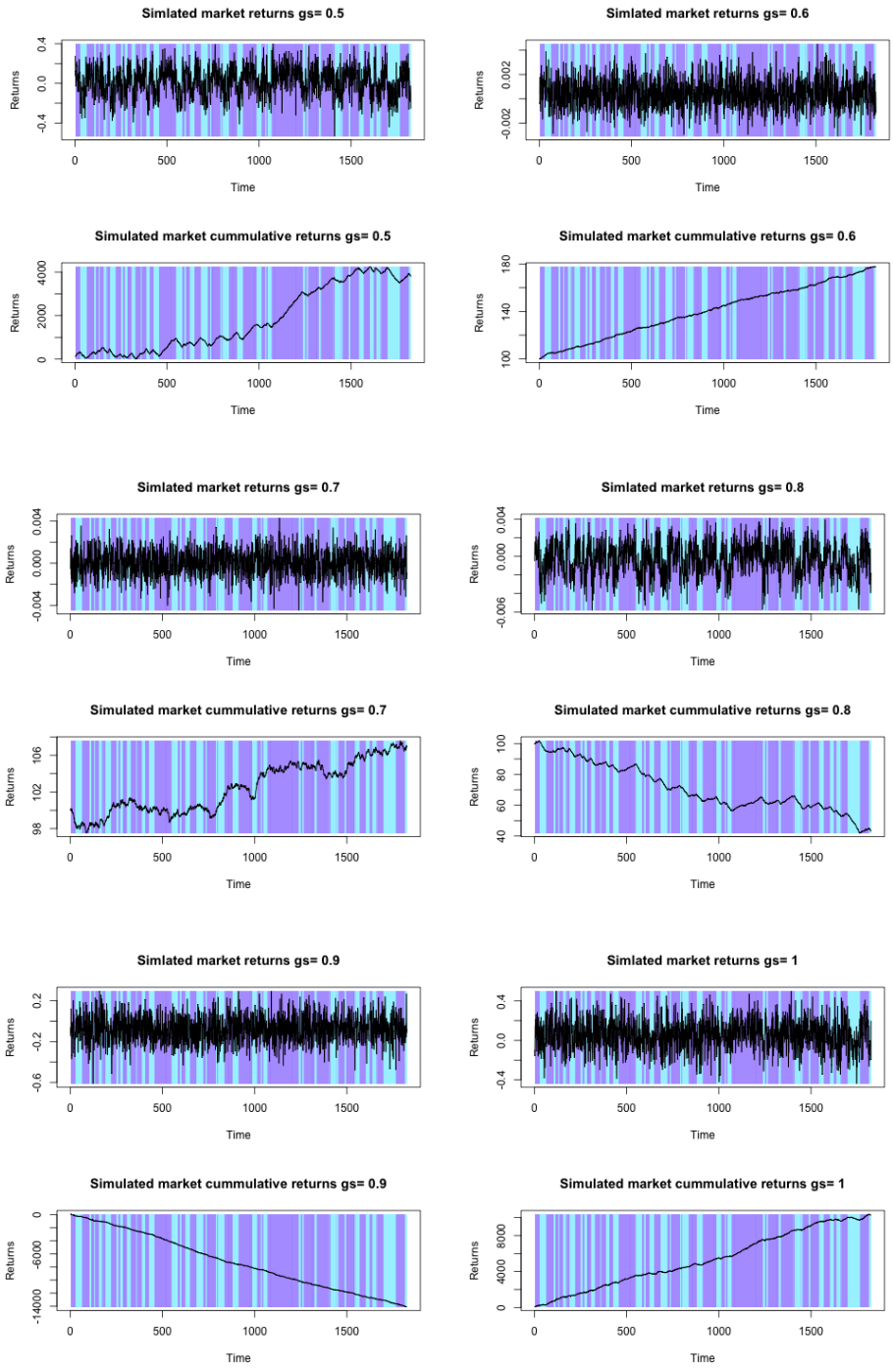
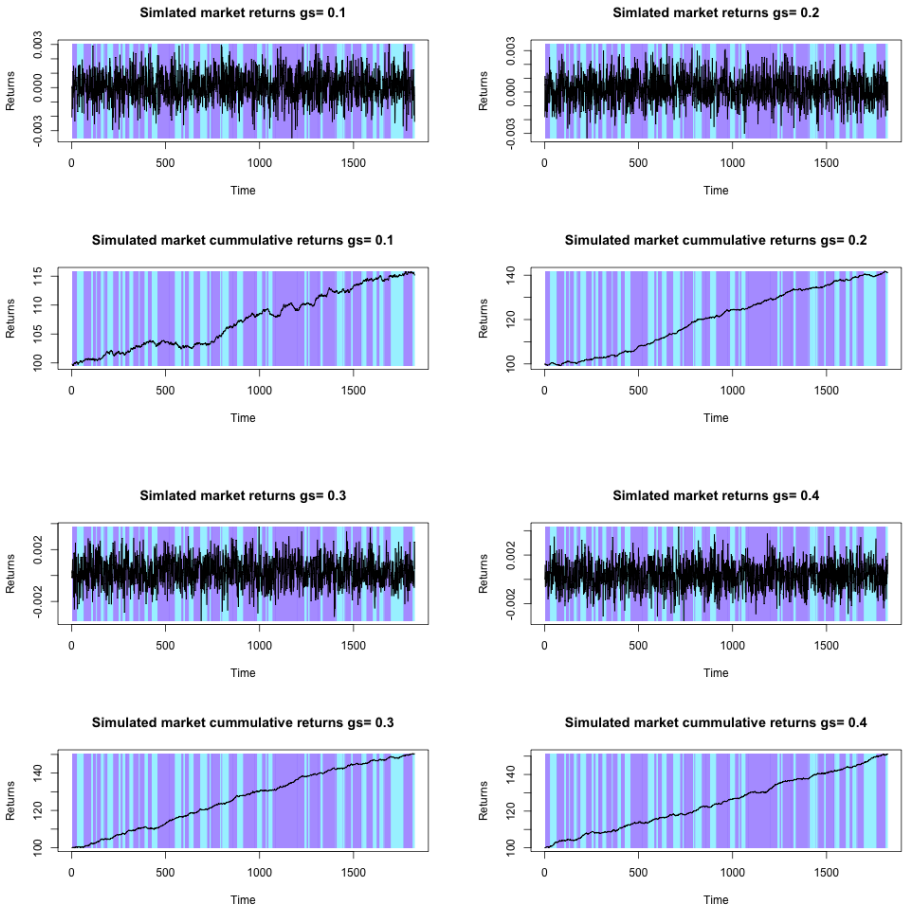


Figure 32: Data II: Simulated Market Data: Noh Anzats 2-State market simulation under the standard Gaussian model where $\epsilon \sim \mathcal{N}(0,1)$ and $\eta \sim \mathcal{N}(0,1)$. The line graph is the daily market average of a 100 stocks over a period of 5 years. Intra-clustering strength is varied from 0.1 which implies weak intra-cluster correlation to 1 which implies very strong cluster correlation. Under this case given the cluster effects η have the same parameterisation we do not expect to see much of a pattern between state hence the need to vary how we parameterise the distributions for the purpose of this experiment


Simlated market returns gs= 0.1
Returns
Time
Simlated market returns gs= 0.2
Returns
Time
Simulated market cummulative returns gs= 0.1
Returns
Time
Simulated market cummulative returns gs= 0.2
Returns
Time
Simlated market returns gs= 0.3
Returns
Time
Simlated market returns gs= 0.4
Returns
Time
Simulated market cummulative returns gs= 0.3
Returns
Time
Simulated market cummulative returns gs= 0.4
Returns
Time

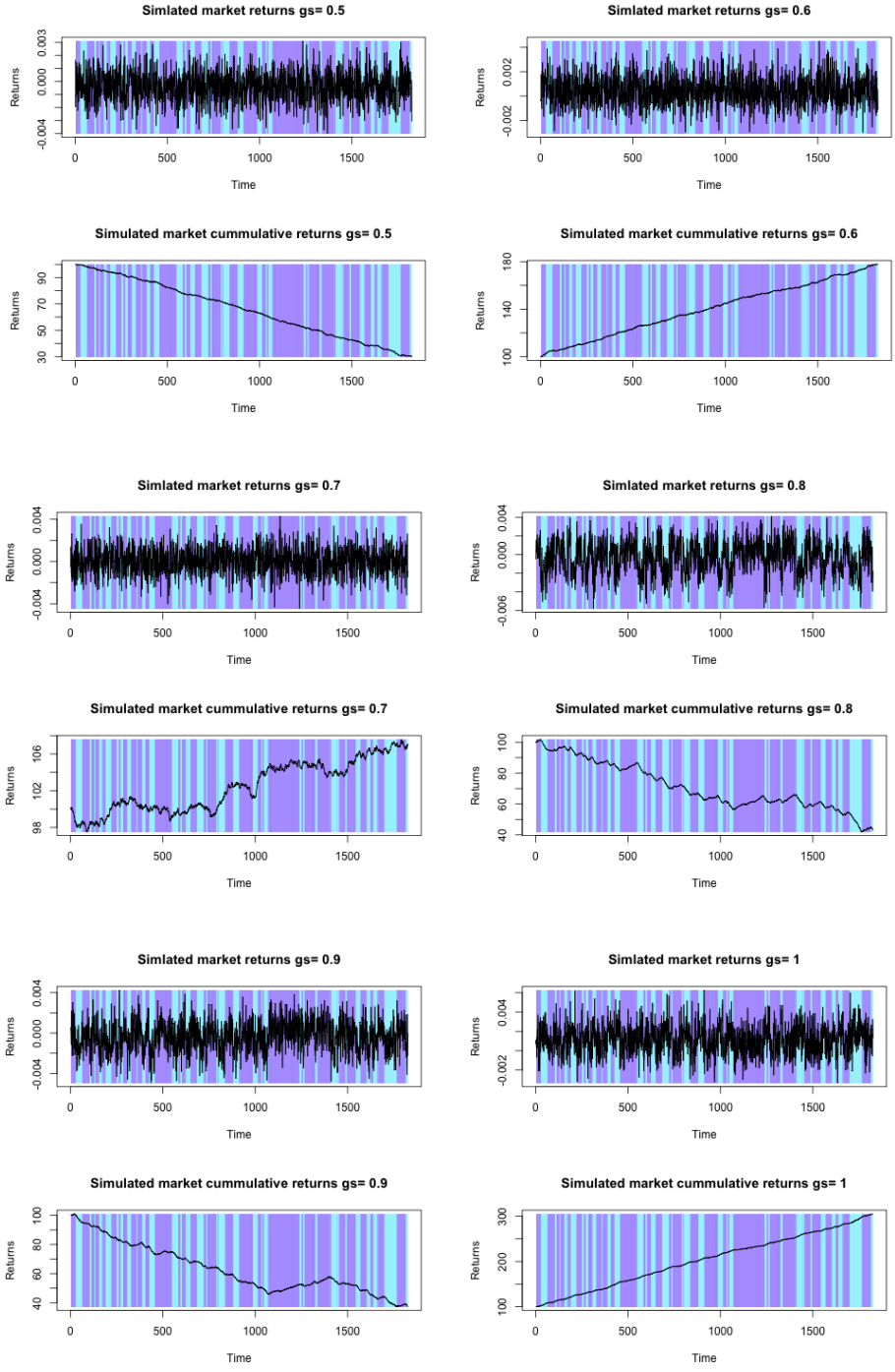


Figure 33: Data II: Simulated Market Data: Noh Anzats 2-State market simulation under the adapted parameterisation where $\epsilon \sim \mathcal{N}(0, 0.01)$ and $\eta_1 \sim \mathcal{N}(-0.0004, 0.012) \quad \eta_2 \sim \mathcal{N}(0.0007, 0.005)$ for each state respectively. The parameters for the cluster effects a estimted by using a a hidden markov model on SnP 500 dataset. The line graphs is the daily market average of a 100 stocks over a period of 5 years. Intra-clustering strength is varied from 0.1 which implies weak intra-cluster correlation to 1 which implies very strong cluster correlation.

A.2 R code: TMFG-LoGo replication

Using Aste (2016) Matlab implementation of TMFG-LoGo, we were able to create R versions `03_TMFG.R` and `04_LoGo.R` which are accessible under the `MarketStates/R` directory.
The main function, `LoGo.solve`, accepts three arguments: `S`, which can either be full covariance matrix or a sparse covariance matrix; and an index table of `separators` and `cliques` which are required if `S` is a sparse matrix. Given a full covariance matrix the function will call `TMFG.filter` which will apply algorithm 2 and return a sparse matrix, separators and cliques. Thereafter equation 5 is computed and the result is a sparse precision matrix.

To test our version, we simply created a set of matrices of varying sizes, applied TMFG-Logo using the Matlab version and our R version and assessed whether we could obtain the same outputs. Figure 34 shows the outputs of from Aste's version and our version. In all cases we were able to confirm the same results. The full testing script is accessible `MarketStates/tests/04_LoGo_test.R` and `MarketStates/tests` directories.

```
LoGo.solve = function(S # full or sparse precision matrix,
    separators = NULL , cliques = NULL # required agruments if S is a
    ↪  TMFG sparse matrix)
```

Listing 1: Input arguments for `04_LoGo.R` function

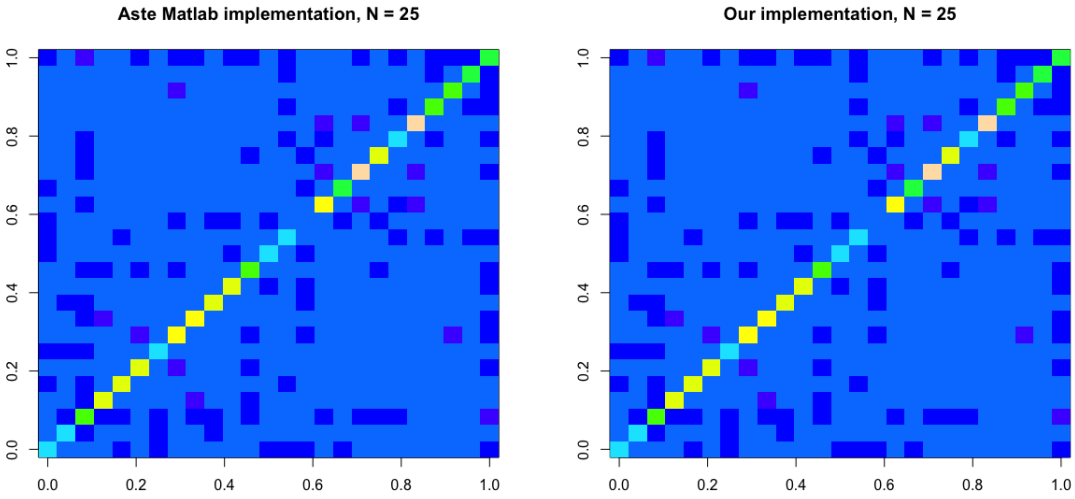


Figure 34: TMFG-LoGo test case on a randomly sampled 25x25 matrix. The left side is the output from Aste (2016) Matlab implementation and the right side is our implementation. In all cases we were able to recover the exact same image confirming that the two versions are able to achieve the same output

A.3 R code: Viterbi replication

Our Viterbi function and testing script is labelled `01_Viterbi.R` and `01_ViterbiTest.R` respectively, both scripts can be found in the `MarketStates/R` and `MarketStates/tests` directories. The `viterbi` function takes the following arguments (presented in code listing 4): `D` a T x K matrix containing penalised Mahalanobis distance for each observation (i.e. as denoted in equation 4); `K` number of states to be identified; `gamma` is the temporal cohesion parameter; `debug` is a Boolean that prints tracing statements when set to `True`.

```
viterbi <- function(D,      # matrix of distances for each ith
↪  observation
                    K,      # Number of states to optimise
                    gamma,  # State switching penalty
                    debug = F # activate tracing statements for
                    ↪  debugging purposes
                    )
```

Listing 2: Input arguments for `01_Viterbi.R` function

The output (presented in code listing 5) of the function is a list of two objects: `Final_path` is the optimal state sequence solved by the algorithm; `Final_Cost` is the minimum Mahalanobis distance corresponding to the optimal path

```
return(
        list('Final_Path' = FinalPath,
            'Final_Cost' = min(CurrentCost)
            )
        )
```

Listing 3: Output code from `01_Viterbi.R` function

Our Viterbi function and testing script is labeled `01_Viterbi.R` and `01_ViterbiTest.R` respectively, both scripts can be found in the `MarketStates/R` and `MarketStates/tests` directories. The `viterbi` function takes the following arguments (presented in code listing 4): `D` a T x K matrix containing penalised Mahalanobis distance for each observation (i.e. as denoted in equation 4); `K` number of states to be identified; `gamma` is the temporal cohesion parameter; `debug` is a Boolean that prints tracing statements when set to `True`.

```
viterbi <- function(D,      # matrix of distances for each ith
↪  observation
                    K,      # Number of states to optimise
                    gamma,  # State switching penalty
                    debug = F # activate tracing statements for
                    ↪  debugging purposes
                    )
```

Listing 4: Input arguments for `01_Viterbi.R` function

The output (presented in code listing 5) of the function is a list of two objects: `Final_path` is the optimal state sequence solved by the algorithm; `Final_Cost` is the minimum Mahalanobis distance corresponding to the optimal path

```
return(
        list('Final_Path' = FinalPath,
            'Final_Cost' = min(CurrentCost)
            )
        )
```

Listing 5: Output code from `01_Viterbi.R` function

We created a 4-state distance matrix which was used as an input to `01_Viterbi.R`. First we added a the ground truth path using a distance value of 10 (the path is highlighted in blue in figure 35). Then we filled in the remaining entries to the matrix with values in the range of 11 - 13, forcing the only optimal path to be the ground truth path we defined (assuming gamma is zero). Code listing 6 shows the function call and outputs, the the block we can see the final path identified by the algorithm and the ground truth path below. Finally we use the adjusted rand index (ARI) to compare accuracy, in the same block we can see an ARI of 1 which means perfect reconstruction of the state path.

Time	1	2	3	4
1	10	12	12	12
2	13	10	12	12
3	12	10	12	13
4	12	10	13	13
5	12	13	10	13
6	13	13	10	13
7	11	11	13	10
8	12	11	12	10
9	12	10	13	13
10	11	10	12	13
11	10	11	13	11
12	10	12	12	11
13	12	10	12	13
14	10	13	12	12
15	12	12	12	10
16	12	13	12	10
17	13	13	13	10
18	13	11	10	13
19	12	13	10	13
20	13	12	10	12

Figure 35: Test case 1: Distance matrix used as an input to `01_Viterbi.R`. Rows represent distance observations in time and columns represent possible states (K =4). Ground truth state path is highlighted in red

```
> viterbi(D = DistanceMatrix, K = 4, gamma = 0, debug = F )
$Final_Path
 [1] 1 2 2 2 3 3 4 4 2 2 1 1 2 1 4 4 4 3 3 3
attr(,"na.action")
[1] 1
attr(,"class")
[1] "omit"

$Final_Cost
[1] 200

> InputData$`True path`
 [1] 1 2 2 2 3 3 4 4 2 2 1 1 2 1 4 4 4 3 3 3
> adj.rand.index(OptimalViterbi$Final_Path, InputData$`True path`)
[1] 1
```

Listing 6: Viterbi results from test 1

A.4 R code: Mahalanobis distance metric

The Mahalanobis distance metric, $d^2_{t,k} = (\vec{\boldsymbol{x}}_t - \vec{\boldsymbol{\mu}}_k)^{\mathrm{T}} \boldsymbol{J}_k (\vec{\boldsymbol{x}}_t - \vec{\boldsymbol{\mu}}_k)$, measures the distance between an observation and a distribution. It can be seen as a multivariate equivalent to Euclidean distance, however, it accounts for the potential correlation between variables. This is achieved by transforming variables into uncorrelated variables and re-scaling the variance to 1. In figure 36 we illustrate how the distance metric varies for a collection of financial time series observations randomly assigned to two clusters. Recall that the overall objective of ICC is to re-assign observations to the minimum distance. The figure provides a nice illustration of how this is achieved.

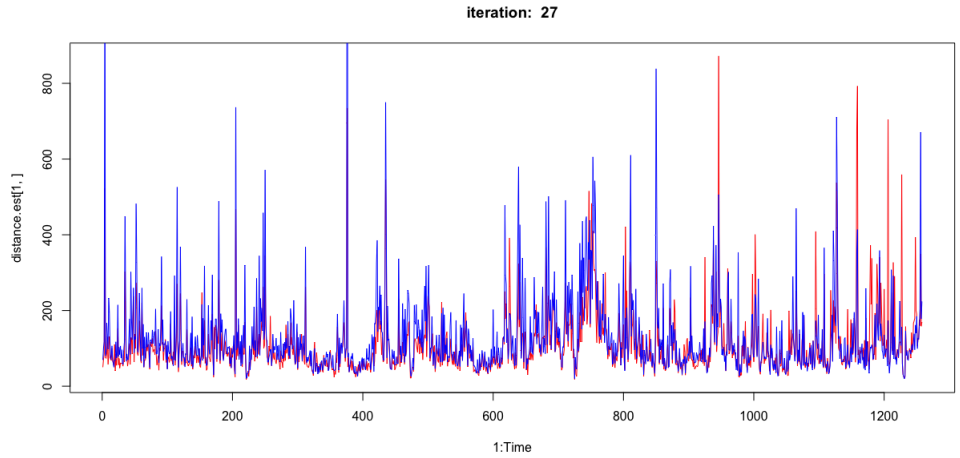


Figure 36: Mahalanobis distance metric evaluated for the full time series of S & P market stocks between February 2013 and February 2018. This graphic illustrates how ICC operates, in the first part of the time series, allocating to the "blue" state will add more distances to the total distance, hence it may be more optimal to allocate to the "red" state. Around the 900th time step the opposite occurs, allocating to the "blue" state is preferred as the "red" state results in more distance.

A.5 R code: State-identification with ICC

Our ICC implementation can be found in `MarketStates/R/05_ICC.R` directory and the replication script in `MarketStates/Scripts/ICC_Replication_script.R`
The main function,`ICC.cluster`, accepts 6 arguments:

- `returnsMatrix` P X T matrix of stock returns
- `gamma`: State switching hyperparameter
- `sparseMethod`: Optional parameter to select full or sparse precision matrix
- `distanceFunction`: Optional parameter to select the distance metric, i.e., Mahalanobis or Likelihood
- `K`: Number of states to identify
- `max.iters`: Maximum number of iterations

Algorithm 5: ICC algorithm

Input ;
P = number of stocks ;
T = length of time-series ;
X = P X T matrix of stock returns ;
γ = state switching parameter ;
Initialize ;
states = randomly shuffled state series length T ;
K = Number of states;
Θ = list of sample statistics ($\vec{\boldsymbol{\mu}}_k$, $\mathbf{J_k}$ per cluster k) based on state allocation ;
Distance = K X T matrix containing the distance of multivariate observations conditioned on Θ ;
for *each iteration it* = 1, ...,*max.iters* **do**
    //E-Step ;
    OptimalViterbi = `viterbi`(Distance, K, γ) // reallocates observations to states and returns an optimal state path;
    // Check if state path is valid, i.e., no single state path returned ;
    if *Sum(diff(OptimalViterbi)) == 0* **then**
        states = reject Viterbi states and randomly shuffle new state path;
    else
        states = OptimalViterbi ;
    //M-Step ;
    Θ = update sample statistics ($\vec{\boldsymbol{\mu}}_k$, $\mathbf{J_k}$ per cluster k) based on new state allocation ;
    Distance = update distance matrix conditioned on new Θ ;
FinalStatePath = states ;