Hierarchical Principal Component Analysis for Correlation Matrix Cleaning

Samuel Ahou, Stephan Hengl, Lennart W. Mailänder

January the 26th, 2025

Abstract

In financial markets, estimating correlation matrices from high-frequency data is challenging due to noise and asynchronous trading times. This study evaluates Hierarchical Principal Component Analysis (HPCA) as a method for improving correlation matrix estimation, comparing it against Eigenvalue Clipping and the Hayashi-Yoshida estimator. HPCA, which leverages clustering and eigenvalue filtering, effectively captures both intra- and inter-sector relationships, offering a structured view of market dynamics. While Eigenvalue Clipping reduces noise and highlights major market trends, its insights are less detailed. The Hayashi-Yoshida estimator alone provides a baseline but struggles to account for market complexity. This report juxtaposes these approaches for correlation matrix estimation in noisy, high-dimensional financial datasets.

FIN-525: Financial big data

Contents

1 Introduction

In today's financial markets, primarily driven by the availability of intraday data, the amount of accessible data is enormous. However, as data is sampled at a higher frequency, the noise increases rapidly, and the information signal gets drowned in the noise. Hence, it essential to rely on precise tools and methods to manage and analyze the provided input effectively. Covariance matrix estimation, a crucial cornerstone of risk management and portfolio optimization, is vital in understanding the underlying market mechanisms and the statistical relationship of traded assets. Yet, its estimation from intraday data poses significant challenges, as raw, noisy data that may contain outliers, can highly skew statistical measures.

Many methods have been proposed to address this pitfall. This paper shall focus on Hierarchical Principal Component Analysis (HPCA) which aims to counteract the difficulties mentioned above by first defining clusters of assets for which the in-between noise is assumed to be lower. Additionally, the covariance matrices within the defined clusters are further cleaned by applying Principal Component Analysis (PCA) and performing eigenvalue selection. Plerou et al., 2002 and Laloux et al., 1999 discuss how eigenvalues of covariance matrices can be used to analyze the proportion of noise for financial settings.
This paper juxtaposes the described method with other proposed empirical covariance cleaning techniques. Eigenvalue clipping and the Hayashi-Yoshida method (Hayashi & Yoshida, 2005) for estimating covariances matrices are considered. The method proposed by Hayashi and Yoshida, 2005 builds the foundation for the two other methods. It allows the handling of asynchronous time series effectively without introducing new bias due to data aggregation or to ignorance of time differences. This is crucial for both the HPCA and Eigenvalue Clipping. Therefore, the estimated correlation matrix via Hayashi-Yoshida is treated as the benchmark.

The paper is structured as follows: First, the mathematical framework of HPCA is introduced. Section 3 then introduces the dataset and the preprocessing steps. Subsequently, the different algorithms for cleaning are compared against each other. Section 5 provides encountered limitations and challenges, as well as possible improvements. Lastly, findings are contextualized.

2 Theory

This section follows the notational framework of Avellaneda, 2019.

Consider a $T \times n$ matrix of log-returns for n assets over a time span T. In the context of PCA, we define the k-th Principal Component or eigenvector of $R \in \mathbb{R}^{n \times n}$ as

$$V^{(k)} = \underset{V \in \mathbb{R}^n}{\operatorname{argmax}} V^\top R V \quad \text{s.t.} \quad \|V\| = 1,\ V^\top V^{(l)} = 0 \quad \text{for} \quad l = 1, \ldots, k-1, \tag{1}$$

where R is the correlation matrix of the assets. For the eigenvalues, it holds that $\lambda^{(1)} \geq \lambda^{(2)} \geq \cdots \geq \lambda^{(n)}$. We can write the Karhunen-Loeve representation of the standardized returns as

$$X_j = \sum_{k=1}^{n} \sqrt{\lambda^{(k)}} V_j^{(k)} F^{(k)}, \tag{2}$$

where the $F^{(k)}$ for $k = 1, \ldots, n$ are called the standardized eigenportfolios of the returns and can be computed as

$$F^{(k)} = \frac{1}{\sqrt{\lambda^{(k)}}} \sum_{i=1}^{n} V_i^{(k)} X_i. \tag{3}$$

In this framework, we call $F^{(1)}$ the first eigenportfolio, which describes most of the variance. Hence, we can write the one-factor model of the standardized log-returns as

$$X_j = \beta_j F^{(1)} + \epsilon_j \tag{4}$$

with $\beta_j = \sqrt{\lambda^{(1)}} V_j^{(1)}$ and an error term ϵ_j which is uncorrelated from $F^{(1)}$.
The first eigenportfolios will be crucial for defining inter-cluster correlation in the HPCA setting.

The framework of HPCA presupposes the market structure to be such that assets can be clustered. This paper exemplifies the method by considering a clustering based on sectors. Theoretically, any other partition of assets is be possible. Thus, the $T \times n$ data matrix of log returns for T timestamps and n assets can be partitioned into k sub-matrices, where each partition contains n_i stocks for $i = 1, \ldots, k$. In our setup, the partitioning will be done according to the industrial sectors of the stocks in the S&P500.

First, we perform a PCA separately to each sector (cluster). We now express (4) as

$$X_j = \beta_j F^{(1,I(j))} + \epsilon_j, \tag{5}$$

where $I(j) = k$, if asset j belongs to sector k and ϵ_j is the corresponding residual. The main assumption of HPCA is that for two assets X_j and X_i, $\mathrm{Corr}(\epsilon_j, \epsilon_i) = 0$ if $I(j) \neq I(i)$. Hence, their correlation depends only on the correlation of the first eigenportfolios of each cluster. For the correlation of the eigenportfolios we write

$$\bar{\rho}^{k,l} = \mathrm{Corr}(F^{(1,k)}, F^{(1,l)}). \tag{6}$$

The new correlation matrix of all n assets combined is constructed as

$$\begin{aligned}
\tilde{R}_{ij} &= \mathrm{Corr}(X_i, X_j) \\
&= \mathrm{Corr}(\beta_i F^{(1,I(i))} + \epsilon_i, \beta_j F^{(1,I(j))} + \epsilon_j) \\
&= \beta_i \beta_j \bar{\rho}^{I(i),I(j)} + \mathrm{Corr}(\epsilon_i, \epsilon_j) \\
&= \begin{cases} R_{ij} & \text{if } I(i) = I(j) \\ \beta_i \beta_j \bar{\rho}^{I(i),I(j)} & \text{if } I(i) \neq I(j) \end{cases}.
\end{aligned} \tag{7}$$

Since we now have derived the theoretical background for HPCA, we shall also present a practical method to compute the eigenvalues and eigenvectors of the correlation matrix shown in Equation (7).

1. On each sector l, perform a PCA to compute the eigenvalues $\lambda^{(1,l)} \geq \cdots \geq \lambda^{(n_l,l)} \in \mathbb{R}$ and the corresponding eigenvectors $V^{(1,l)}, \ldots, V^{(n_l,l)} \in \mathbb{R}^{n_l}$. Next, embed the eigenvectors into $\mathbb{R}^n$ by

setting

$$W_j^{(i,l)} = \begin{cases} V_j^{(i,l)} & \text{if } I(j) = l \\ 0 & \text{if } I(j) \neq l \end{cases} . \tag{8}$$

Consequently, $\{W^{(i,l)}\}_{i=1,\dots,n_l;l=1,\dots,k}$ forms an orthogonal basis of $\mathbb{R}^n$.

2. Before we can proceed with further analysis, we have to note that the subspace $\Omega \subseteq \mathbb{R}^n$ defined as

$$\Omega = \left\{ \sum_{l=1}^{k} \alpha_l W^{(1,l)} \mid \alpha \in \mathbb{R}^k \right\}$$

is invariant under the linear operator $\tilde{R}$, since the $W^{(1,l)}$ presents the embedding of the eigenvectors.

3. Finally, we construct the eigenvectors and eigenvalues of the adapted correlation matrix $\tilde{R}$. First, construct a matrix M as

$$M^{k,l} := \sqrt{\lambda^{(1,k)}\lambda^{(1,l)}}\bar{\rho}^{k,l}. \tag{9}$$

Next, compute the eigenvalues $\mu^{(1)} \geq \cdots \geq \mu^{(k)}$ of M and also the corresponding normalized eigenvectors $\alpha^{(1)}, \dots, \alpha^{(k)}$. Transforming W leads to the eigenvectors of $\tilde{R}$

$$\tilde{W}^{(i,l)} := \sum_{p=1}^{k} \alpha_p^{(l)} W^{(i,p)} \tag{10}$$

with corresponding eigenvalues $\mu^{(1)}, \dots, \mu^{(k)}$. This enables the construction of the estimated correlation matrix $\tilde{R} = \tilde{W}^T D \tilde{W}$, where $\tilde{W}$ and $D = \text{diag}(\mu^{(1)}, \dots, \mu^{(k)})$ are the eigenvector and eigenvalue matrices respectively.

Thus, following the steps outlined above, one can ultimately construct the HPCA correlation matrix's eigenvalues and eigenvectors.

3 Data Processing

3.1 Dataset

Our analysis estimates the correlation matrix for a subgroup of the S&P 500's assets.[1] We process tick-by-tick data from the year 2010, more precisely, data for the month of May. We have a total number of 393 assets before preprocessing, for which we have a single file for each day of the month. We chose this dataset because the HPCA algorithm needs many assets to form clusters of sufficient sizes, in order to compute intra-cluster correlations.

The datasets at hand contain information about all executed trades for a specific asset for each open market day of May 2010. They consist of the following columns: *xltime*, which encodes the timestamp of the trades, *trade-price* and *trade-volume*, which contain the trading volume and the corresponding price for each trade. The two columns *trade-stringflag* and *trade-rawflag* give

[1] The data can be downloaded from the link provided on Moodle and can be found here: https://drive.switch.ch/index.php/s/0X3Je6DauQRzD2r

information about the type of trade. Table 1 gives the first lines of the raw data for the AAP stock on the 03/05.

xltime	trade-price	trade-volume	trade-stringflag	trade-rawflag
40301.500359	45.10	1132	marketclosed—volumeupdate	[CTS_QUAL]
40301.538732	45.10	31796	blocktrade—marketclosed—volumeupdate	[CTS_QUAL]
40301.557121	45.10	1650	marketclosed—volumeupdate	[CTS_QUAL]
40301.557147	45.10	1650	marketclosed—volumeupdate	[CTS_QUAL]
40301.562507	45.13	200	uncategorized	[CTS_QUAL] XTR

Table 1: Excerpt of trade data for ticker AAP on the 03/05/2010.

In the following sections, we refer to 'Day' and 'Month' data for our analysis. We choose to evaluate our methods on tick-by-tick data for a single day, chosen to be the 03/05/2010 (first trade day of the month), as well as on the aggregated data for the whole month of May. To ease notation, we refer to the data of the 3rd of May as 'Day' data and the aggregated data as 'Month' data.

3.2 Preprocessing

We preprocess the data to have the desired structure and eliminate data that is possibly corrupted. Since the data folder is provided as a *tar*-archive, we must first unpack all the folders. We then also simplify the folder structure for each asset to avoid unnecessary enclosing of the raw daily trade files.

Owing to the fact that vast amounts of data are processed at once, the Python library *polars* ("Polars - DataFrames for the new era", 2024) is used to process the files efficiently. Basic sanity checks for each asset are conducted and corrupted files excluded. Additionally, all assets pertaining to faulty files are excluded. This leaves 304 assets for analysis. Furthermore, a new directory that includes all the information for the specific day is created. Columns are no transformed further. We transform the *xltime* column to a *Timestamps* object according to Eastern Daylight Time (UTC-04:00). We then exclude special trades according to the *trade-stringflag* column and drop the *trade-volume*, *trade-stringflag* and *trade-rawflag* columns. Last but not least, we compute the log-returns and replace *trade-price* by their corresponding log-returns. We then save the resulting datasets in a clean directory for further analysis. Table 2 gives the first lines of the processed dataset for the AAP stock on the 3rd of May.

Time	Log-Return
2010-05-03 09:30:01-04:00	0.000665
2010-05-03 09:30:04-04:00	-0.002209
2010-05-03 09:30:06-04:00	0.002430
2010-05-03 09:30:10-04:00	-0.002430
2010-05-03 09:30:18-04:00	-0.000221

Table 2: Excerpt of processed trade data for AAP on the 03/05/2010.

Figure 1 shows log-returns of the AAP stock. Periods with no trading activity, i.e. weekends and closing hours of the Stock Exchange, are clearly visible on the plot of 'Month' data.

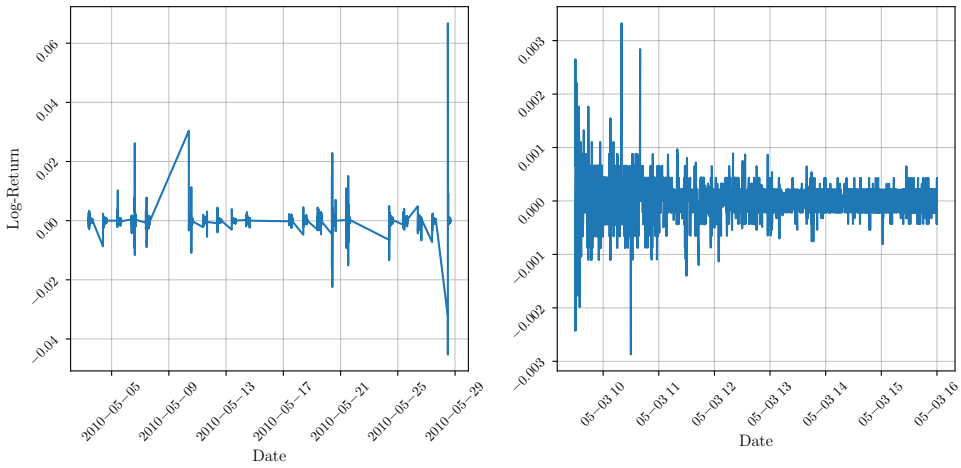

Figure 1: Log-Returns of AAP for 'Month' (left) and 'Day' data (right).

3.3 Standardization

One – but not the only – way of mitigating the effect of volatility clustering, is to de-GARCH each asset's log-returns. This process somewhat diminishes the influence of periods of high or low volatility on the correlation of financial instruments. Indeed most log-returns display some level of heteroskedasticity. By taking the standardized residuals of the fitted GARCH model, we theoretically achieve homoskedasticity (constant variance over time). Note that this might not be true in this specific case, since it is well known that financial time series (especially tick-by-tick data) do not follow a normal or t-distribution. Since it is not feasible to fit a GARCH model on a polars *LazyFrame*, we need to load each asset's dataset one after another to avoid memory overflow. The data loading and fitting process was parallelized in order to handle at least 10 assets simultaneously, achieving faster runtimes. Approximate runtimes are reported by Table 3.

Data Type	**Normal**	**Parallelized**
Day	15 min	3 min
Month	55 min	11 min

Table 3: Runtimes for standard/parallelized de-GARCHing process for 'Day' and 'Month' data.

Figure 2 represents the resulting de-GARCHed log-returns of the AAP stock. If we compare the log-returns for the desired day of Figure 1 with the standardized residuals of Figure 2, it is clear that the residuals' volatility is indeed much less clustered, except for a few outliers.

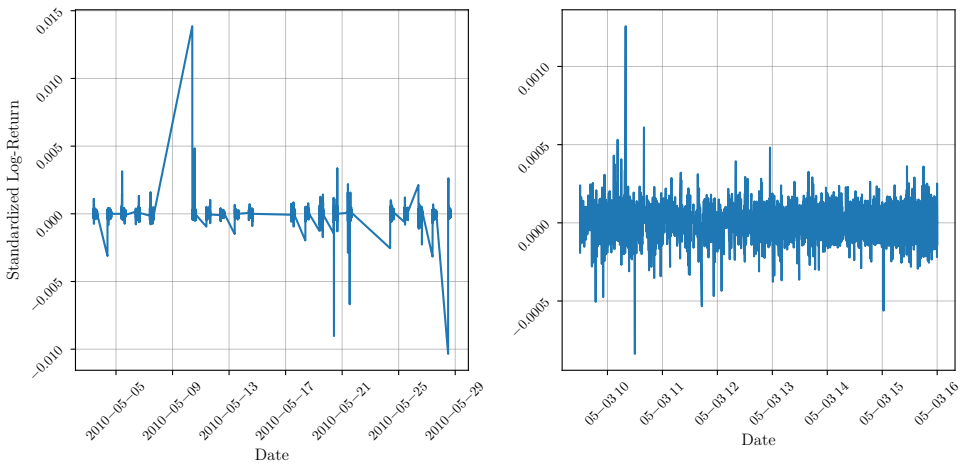

Figure 2: Standardized log-returns (Residuals) of AAP for 'Month' (left) and 'Day' Data (right).

4 Implementation & Results

4.1 Implementations

As said in the previous sections, we aim to compare the three following correlation matrix estimation methods:

1. **Hayashi-Yoshida Estimator**
   When computing the correlation, it is sensible to account for the time lag between two asynchronous time series. We do this by using a method proposed by Hayashi and Yoshida, 2005. This method also serves as the benchmark for the other estimation methods.

   Implementation-wise, we followed the procedure outlined in the lecture (Challet, 2024, p. 20). However, we opted to compute the Hayashi-Yoshida estimator for each pair of assets separately, in order to prevent forward filling of a large amount of empty cells. This choice was made due to computational and memory limitations. We then simply combined them to build the correlation matrix. This approach allows us to load data efficiently, preventing possible memory overflow.

2. **Eigenvalue Clipping**
   Eigenvalue Clipping is a simple but effective way to account for noise in the observed data. It uses the Marchenko-Pastur distribution (Challet, 2024, p. 45) to filter/clip eigenvalues that pertain to the noisy part of the data. We implemented the Eigenvalue Clipping algorithm as presented the lectures (Challet, 2024, p. 58).

3. **HPCA**
   For the HPCA, we consider the Yahoo Finance sector classification in order to cluster the S&P500 assets. This enables us to use the *yfinance* API ("yfinance - PyPi", 2024) to extract the sector information for a specified ticker from the Yahoo Database. Nonetheless, we still have to add some missing classifications manually. We implemented the HPCA algorithm following the theory presented in Section 2 and taken from Avellaneda, 2019 and Avellaneda and Serur, 2020.

These implementations allowed us to compute correlation matrices on both 'Day' and 'Month' data. Runtimes are reported in Table 4. From a computational standpoint, the HPCA outperforms the other two methods by a factor 4 approximately.

Data Type	**Hayashi-Yoshida**	**Eigenvalue Clipping**	**HPCA**
Day	3 min	3 min	1 min
Month	40 min	40 min	10 min

Table 4: Approximate runtimes for each method for 'Day' and 'Month' data.

4.2 Results

In order to determine the performance of the correlation estimation methods described in Section 4.1, we compare the distribution of the correlation matrix eigenvalues to the Marchenko-Pastur distribution. Stemming from Random Matrix Theory, the Marchenko-Pastur distribution describes the asymptotic probability density of the eigenvalues of $Y = n^{-1}XX^\top$ where $X_{ij} \overset{\text{iid}}{\sim} (0, \sigma^2)$.
Eigenvalues distributed according to the Marchenko-Pastur distribution indicate a small signal-to-noise ratio and thus a poor estimate of correlation. Eigenvalue Clipping is solely based on this fact and clips eigenvalues lying in the range of Marchenko-Pastur's distribution, leaving larger eigenvalues untouched which are considered as actual signal.
We now compare results for correlation matrices and eigenvalue distributions obtained for all three methods and so for 'Day' and 'Month' data.

Day Data

Figure 3 gives the correlation matrices (in absolute values) obtained via the three methods described hereabove. Hayashi-Yoshida shows a very sparse correlation matrix, close to the identity matrix. This would imply that the assets are approximately uncorrelated. Indeed, Figure 4 shows that the distribution of eigenvalues is very close to the Marchenko-Pastur distribution, indicating that according to this estimation method, the assets are uncorrelated.

Correcting the previous method via the Eigenvalue Clipping method somewhat mitigates the illusion of uncorrelatedness. Indeed, after replacing all eigenvalues in the range of Marchenko-Pastur distribution (and smaller) by a constant value, we observe some structure in the correlation matrix. This structure should be related to the largest eigenvalues present in the spectrum, representing the most important modes present in the market (the market itself, large caps, sectors etc).

As per the HPCA, the correlation matrix is highly structured; clusters are clearly visible on the diagonal and the inter-cluster correlation can be spotted through the presence of the off-diagonal correlation blocks. This result was to be expected, as clusterisation is core to this method. Furthermore, we note that the HPCA eigenvalues do not have Marchenko-Pastur density. Rather, eigenvalues are either quite big ($> 10^1$) or quite small ($< 10^{-1}$). This indicates that the HPCA efficiently latches on to the most significant modes (market and sectors) and discards the remaining eigenvalues by setting them to small values.

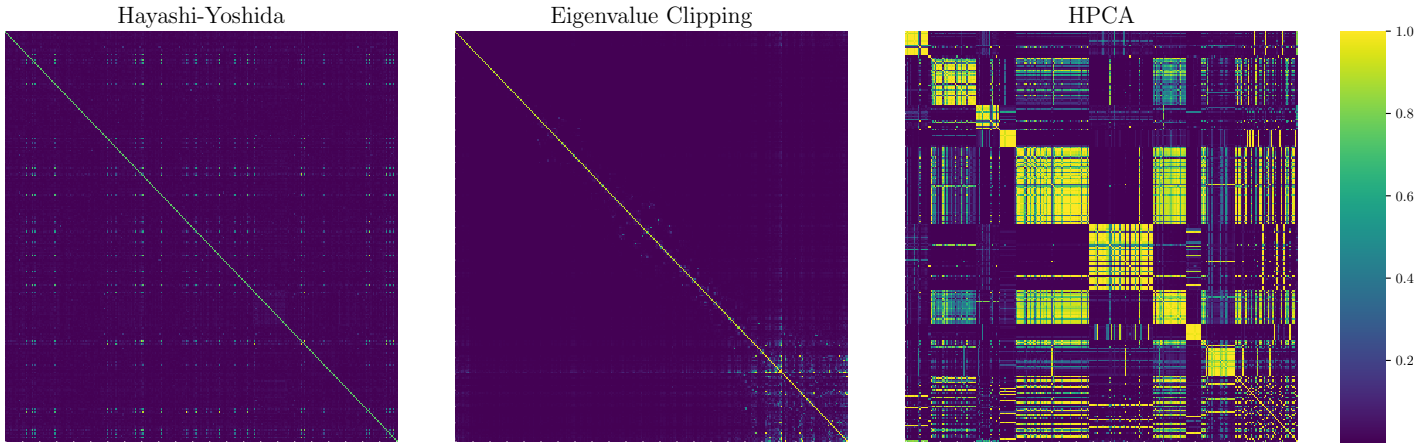

Figure 3: Correlation Matrices - Hayashi-Yoshida (left), Eigenvalue Clipping (center), HPCA (right) for the 3^{rd} May 2010.

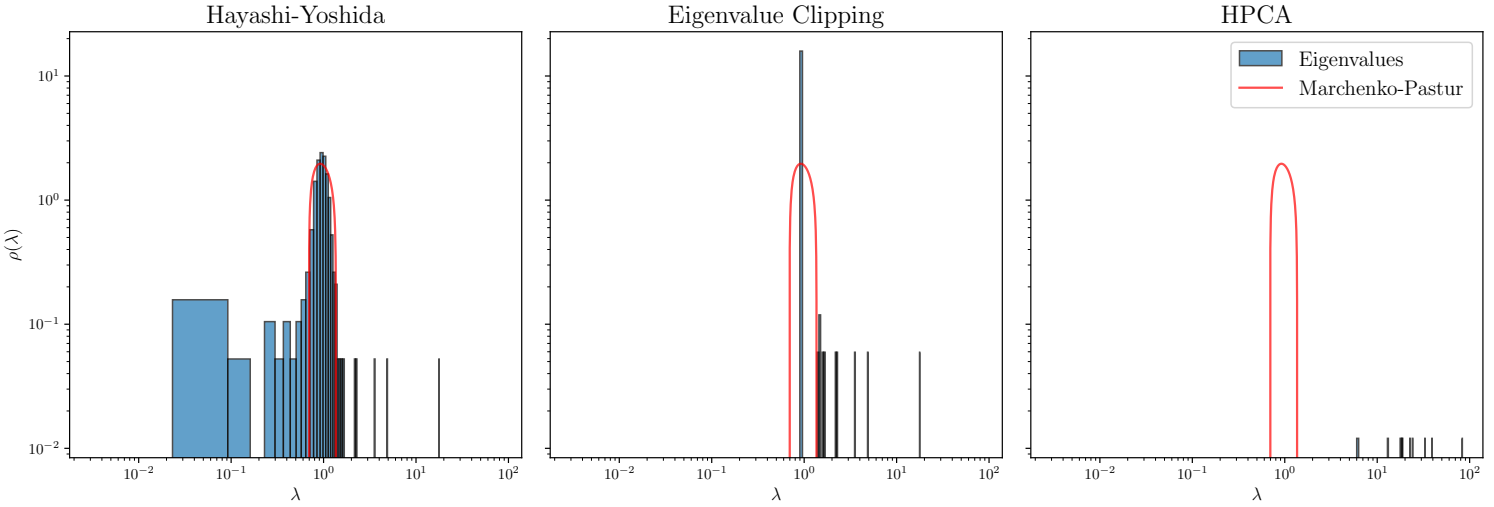

Figure 4: Eigenvalues' vs Marchenko-Pastur's Distribution - Hayashi-Yoshida (left), Eigenvalue Clipping (center), HPCA (right) for the 3^{rd} May 2010.

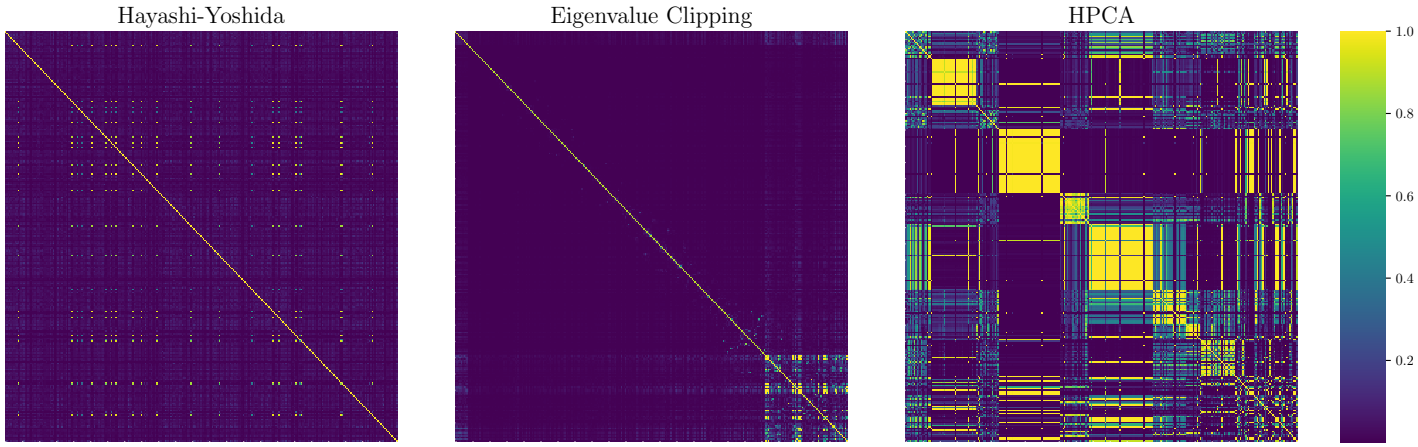

Figure 5: Correlation Matrices - Hayashi-Yoshida (left), Eigenvalue Clipping (center), HPCA (right) for May 2010.

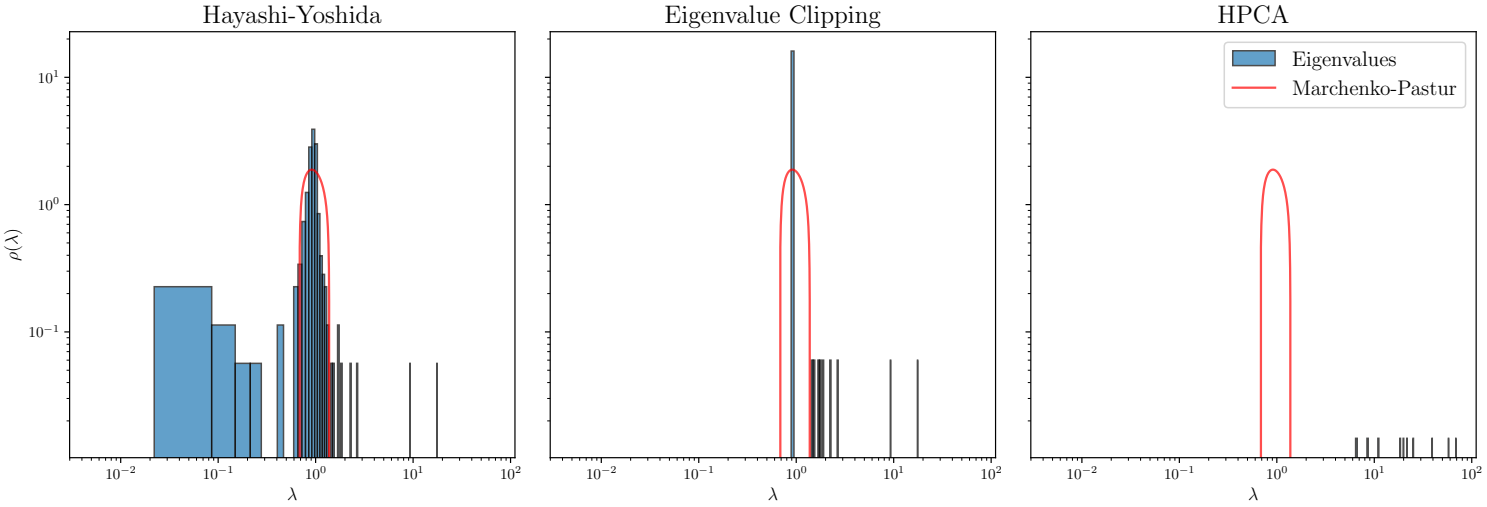

Figure 6: Eigenvalues' vs Marchenko-Pastur's Distribution - Hayashi-Yoshida (left), Eigenvalue Clipping (center), HPCA (right) for May 2010.

Month Data

Figures 5 and 6 show the same results for the 'Month' data. The conclusions are similar to the ones drawn for a single day. Indeed, the Eigenvalue Clipping shows again, and even more clearly, the structure seen in Figure 3. We note however that the HPCA seems to suffer to some extent from the extended time frame. Although the method demonstrates again superior performance, the structure seems less defined. The off-diagonal blocks are much more spread apart or less tidily packed.

5 Limitations and Possible Improvements

This section briefly overviews the challenges faced while implementing the above-described methods and provides a short outline of possible improvements.

Data Size

A crucial part of the runtime performance and feasibility of computations depends on the data size. Already, at an early stage of the project, we faced difficulties in processing all the data at once.[2] This is due to the fact that intraday data entail a vast amount of observations at a fine asynchronous time grid. Therefore, by merging all assets to one matrix, we create a matrix of several million rows and hundreds of columns. Even when processing this matrix with polar's streaming functionality, we must reduce the chunk size to as low as 100. Thus, the computation times skyrocket to unbearable values. Therefore, we opt to handle the data's size effectively by only accessing files needed for the current computations.

GARCH Standardization

While the de-GARCHing of financial time series is crucial to mitigate the influence of high and low volatility phases, it highly depends on the distributional assumptions made. We assume that the series could be modelled by a GARCH model with an underlying t-distribution to try to account for the heavy tails of the data. However, it is well known that, especially log-returns of tick-by-tick data, do not follow such a distribution. Furthermore, we only use a grid search between 1 and 3 for the parameters q and p due to computational constraints. To measure the performance of each combination, we use the Bayesian Information Criterium (BIC) since it favors simple models. In future experiments, this could be investigated further and optimized. Moreover, one might consider different approaches to account for volatility clustering. One could e.g., model volatility separately via stochastic volatility models.

Hayashi-Yoshida Implementation

As described in the subsection discussing the data size, the first approach employed the Hayashi-Yoshida correlation estimator using a matrix containing each asset's log-returns. However, apart from the pitfalls described above, this also leads to long periods of constant values due to the forward filling. This might result in biased correlations. We counteract this problem by computing the correlation for each pair of assets separately and combining them afterwards. In this case,

[2] Most of the computations are conducted on an Apple MacBook Pro with an M1 Max Chip and 32GB of RAM.

we opt not to parallelize this procedure because it heavily relies on *numpy* operations, which are already highly optimized and parallelized.

HPCA Implementation

To implement the HPCA, we decided to reconstruct the matrix's eigenvalues and eigenvectors. At first, this might seem unnecessary since a concise definition of how each entry can be computed exists. However, this would include estimating the regression coefficient β. Due to the asynchronous time series, this can not be done without interpolating each time series on a common grid, which we try to avoid to not introduce additional bias. However, we must still use interpolation to compute the inter-cluster correlation when computing the first eigenportfolios.

6 Conclusion

The above study explored the possibilities of Hierarchical PCA as a method to estimate correlation matrices in the context of intraday data. Further, it compared its efficiency, runtime, and feasibility with one other approach, eigenvalue clipping, and a baseline estimation using the Hayashi Yohsida method to account for the asynchronicity of the time series. While the HPCA is based on the assumption that the signal-to-noise ratio can be improved by dividing the assets into clusters and then computing their intra-cluster correlation, eigenvalue clipping uses results from random matrix theory to identify and downweight eigenvalues mainly containing information about the noise component.

Firstly, section 2 introduced the mathematical foundation and justification of why and how HPCA can be used to estimate correlation matrices. Two methods were shown to compute the HPCA correlation matrix. On one hand, by regressing each eigenportfolio for a cluster on the corresponding assets and then using the inter-cluster correlation and the resulting regression coefficients to calculate each entry separately. On the other hand, by computing the eigenvalues and eigenvectors of the matrix and then reconstructing it. While Avellaneda, 2019; Avellaneda and Serur, 2020 implemented the HPCA by directly computing the correlation matrix, this paper opted to use the eigendecomposition of the matrix to mitigate the effects of the asynchronous time series.

Initial drawbacks during the processing of the data led to various adaptations and improvements in the code's implementation. To minimize the size of the loaded data, most operations were conducted by loading and processing files in small chunks. This, among other things, allowed to parallelize the de-GARCHing of the time series effectively.

Concerning the empirical study itself, the following conclusion could be drawn. The HPCA stood out for its ability to reveal both sector-specific and cross-sector relationships, offering a clearer picture of market structure. Its performance, however, showed some sensitivity when applied to longer timeframes, suggesting room for improvement in handling extended datasets. The Eigenvalue Clipping method proved effective in reducing noise and uncovering some market patterns but lacked the nuanced insights provided by HPCA. On the other hand, the baseline Hayashi-Yoshida often failed to capture the complexity of asset relationships, due to its sparse results.

Though they were outside the scope of this report due to time and complexity constraints, further

adaptations and improvements should be considered. While this report focuses on dividing the assets according to financial sectors, it might be beneficial to consider other clustering approaches. Moreover, a more rigorous fine-tuning of the interpolation step might make estimations more robust.

These findings underscore the importance of selecting the right tools for the task when working with noisy, asynchronous financial data. While HPCA shows great promise, further refinements are needed to enhance its robustness across different data scales and conditions.

References

Avellaneda, M. (2019). Hierarchical pca and applications to portfolio management. https://arxiv.org/abs/1910.02310

Avellaneda, M., & Serur, J. A. (2020). Hierarchical pca and modeling asset correlations. https://arxiv.org/abs/2010.04140

Challet, D. (2024). Financial big data: Week 9 - correlations and noise cleaning [Lecture slides, FIN-525: Financial Big Data, École Polytechnique Fédérale de Lausanne (EPFL). November 13, 2024. Available from course materials.].

Hayashi, T., & Yoshida, N. (2005). On covariance estimation of non-synchronously observed diffusion processes. Bernoulli, 11(2), 359–379.

Laloux, L., Cizeau, P., Bouchaud, J.-P., & Potters, M. (1999). Noise dressing of financial correlation matrices. Physical review letters, 83(7), 1467.

Plerou, V., Gopikrishnan, P., Rosenow, B., Amaral, L. A. N., Guhr, T., & Stanley, H. E. (2002). Random matrix approach to cross correlations in financial data. Physical Review E, 65(6), 066126.

Polars - dataframes for the new era [Accessed: 2024]. (2024).

Sheppard, K., Khrapov, S., Lipták, G., van Hattem, R., mikedeltalima, Hammudoglu, J., Capellini, R., alejandro-cermeno, bot, S., Hugle, esvhd, Fortin, A., JPN, Judell, M., Russell, R., Li, W., 645775992, Adams, A., jbrockmendel, ... Çelik, B. (2024, November). Bashtage/arch: Release 7.2 (Version v7.2.0). Zenodo. https://doi.org/10.5281/zenodo.14035889

Yfinance - pypi [Accessed: 2024]. (2024).