Stat

The ISI's Journal for the Rapid
Dissemination of Statistics Research

(wileyonlinelibrary.com) https://doi.org/10.1002/sta4.176

Multiple day biclustering of high-frequency financial time series

Haitao Liu[a], Jian Zou[b]* and Nalini Ravishanker[c]

Received 8 December 2017; Accepted 28 December 2017

With recent technological advances, high-frequency transaction-by-transaction data are widely available to investors and researchers. To explore the microstructure of variability of stock prices on transaction-level intra-day data and to dynamically study patterns of comovement over multiple trading days, we propose a multiple day time series biclustering algorithm (CC-MDTSB) that extends the time series biclustering algorithm (CC-TSB). For identifying biclusters within each trading day, our algorithm provides a faster alternative to the random replacement method in the CC-TSB algorithm. Moreover, our algorithm does not require prespecification of the number of biclusters for each trading day. Instead, we set a threshold on the number of stocks within the biclusters to yield an adaptive stopping criterion for multiple day analysis. An analysis of the biclusters determined over multiple trading days enables us to study the dynamic behaviour of stocks over time. We effectively estimate the comovement probability of each *m*-tuple of stocks conditional on the other stocks within the dynamic biclusters and propose a method to forecast comovement days using a nonparametric double exponential smoothing procedure. Copyright © 2018 John Wiley & Sons, Ltd.

Keywords: biclustering; comovement; high-dimensional data; high frequency; time series

1 Introduction

With recent technological advances, high-frequency transaction-by-transaction data are readily available to investors and researchers. Analysis of these data is very useful for understanding traders' activity and market efficiency in financial applications and risk management (O'Hara, 2015; Menkveld, 2013; Manahov et al., 2014). There is a need to develop attractive methods for analysing non-linear temporal features and stochastic interdependencies in stock characteristics over time. Volatility analysis is one way to understand non-linear temporal dependence. For instance, Zou et al. (2015) proposed a two-stage preaveraging method to estimate the volatility matrix of high-frequency data. Zou & Huang (2016) proposed an optimized block realized kernel method to estimate the volatility matrix for large portfolio allocation problems.

Another attractive approach for studying interdependencies between stocks is to *cluster* them on the basis of characteristics such as returns. Under an efficient clustering scheme, we would expect "similar" stocks that comove stochastically to be grouped into the same cluster. The clusters can provide insight into stock properties via this segmentation, and investors can avoid volatility risk by allocating investments among the stocks in different clusters (Dose

[a]Data Science Program, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[b]Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[c]Department of Statistics, University of Connecticut, Storrs, CT 06269, USA
*Email: jzou@wpi.edu

& Cincotti, 2005; Tola et al., 2008; Harvill et al., 2016). However, a clustering algorithm may fail to extract local comovement patterns across a certain subset of time intervals. Biclustering algorithms were proposed to address this issue.

Biclustering was first proposed by Hartigan (1972) to investigate voting data of US states in order to identify groups of observations with similar patterns under a certain subset of features. Many variants of biclustering follow a similar framework when both row and column factors are of particular interest. For instance, Lee et al. (2010) proposed a biclustering method via sparse singular value decomposition to seek a low-rank, checkerboard structured matrix approximation to data matrices. Lee & Huang (2014) studied biclustering for binary data matrices using maximum penalized Bernoulli likelihood estimation. Tan & Witten (2014) used L_1 penalty on the means of biclusters to obtain sparse and interpretable biclusters. Laclau & Nadif (2017) investigated the problem of co-clustering binary data in a latent block model framework with diagonal constraints. Mankad & Michailidis (2014) introduced a biclustering method for exploration and pattern detection in three-dimensional data arrays. Biclustering in the Bayesian framework was studied by Sheng et al. (2003), Zhang (2010), Chekouo et al. (2015) and Wyse & Friel (2012).

Cheng & Church (2000) generalized the biclustering algorithm to time series data for biological gene expression applications by alternately eliminating rows (genes) and columns (time points) to minimize the mean squared residue score. However, their algorithm ignored the need to preserve contiguity of the time sequence. To address this issue, Zhang et al. (2005) extended the Cheng & Church (2000) algorithm by eliminating only the start and end points in a time sequence. Specifically, their time series biclustering algorithm (CC-TSB) alternately eliminated rows and time points according to low mean squared residue scores while considering only the starting and ending points in a time interval as being eligible for deletion. The CC-TSB algorithm requires prespecification of the number of biclusters, which is usually difficult in practice. Furthermore, the algorithm only works on a set of consecutive discrete time points; it is not possible to include data from multiple days or data with gaps within a time interval.

Moreover, the literature is rather sparse on applications of biclustering to financial data. *Static* biclustering algorithms for stocks based on low-frequency, daily stock prices are generally constructed on the basis of a single block of daily data that span a few years under low-dimensional settings (Babu et al., 2012; Darie & Cosmin, 2009). Huang (2011) applied a biclustering algorithm to explore comovement in a small set of currencies across non-consecutive time periods. Xue et al. (2015) used a biclustering method on daily stock prices for learning trading rules. To be effective in practice, it is most useful to bicluster stocks within a moving time window and to utilize interdependencies within and between the time windows for financial applications. There is a need to develop novel biclustering methods using moving time windows on transaction-level data within a day and to dynamically study patterns of comovement over multiple trading days. To this end, we propose a multiple trading day time series biclustering (CC-MDTSB) algorithm. Our algorithm sets a threshold for the number of stocks explained by the biclusters within a trading day rather than a threshold for the number of biclusters as in the CC-TSB algorithm. This yields an adaptive stopping criterion for multiple day analysis. In addition, we also analyse the comovement probability that a selected m-tuple of stocks falls within the same bicluster in a given time period.

The rest of the paper is organized as follows. Section 2 provides a review of static biclustering. Section 3 describes the high-frequency S&P 100 stock prices and 1-minute averaged returns for 2013. Section 4 gives details of the CC-MDTSB algorithm. Section 5 defines the comovement probability of an m-tuple stocks and proposes an approach for forecasting the number of trading days for which the m-tuple falls within the same biclusters in a given time period. Section 6 presents a detailed analysis of the S&P 100 data using our approach. Section 7 summarizes the paper and our contributions.

(wileyonlinelibrary.com) https://doi.org/10.1002/sta4.176

2 Review of static biclustering

Unlike clustering, a biclustering algorithm clusters the rows and columns of a data matrix simultaneously. Figure 1 illustrates the difference between clustering and biclustering for an example on the log returns of seven stocks at nine consecutive time points. A biclustering algorithm would consider a subset of columns (time points) and identify the light grey-shaded submatrix as a bicluster, which indicates that stocks 3, 4 and 5 comove from T3 to T6. By contrast, a clustering algorithm would use all the columns (from T1 to T9) to assess similarity between the stocks. While stocks 3, 4 and 5 may not exhibit similarity for the entire period from T1 to T9, we observe strong similarities from T3 to T6. Therefore, stocks 3, 4 and 5 can be viewed as belonging to a bicluster from times T3 to T6 rather than as being clustered over the entire time period.

The literature on biclustering discusses four main types (Pontes et al., 2015), that is, (i) biclusters with constant values; (ii) biclusters with constant values on rows or columns; (iii) biclusters with coherent values; and (iv) biclusters with coherent evolutions. The simplest case is type (i) where all the entries have a constant value. This is illustrated in Figure 2. Since biclustering was first proposed by Hartigan (1972) to investigate voting data of US states, various biclustering algorithms have been proposed, such as exclusive row and column biclusters, overlapping biclusters with hierarchical structure and non-overlapping biclusters. Pontes et al. (2015) gave a comprehensive review of the evolution of various biclustering algorithms.

Cheng & Church (2000) generalized the biclustering algorithm to alternately eliminate rows and time points based on the mean squared residue score (1), in order to investigate biological gene expression data. Given an $R \times C$ data matrix, let $I \subset R$ and $J \subset C$ denote subsets of rows and columns. The pair (I, J) defines a submatrix A_{IJ} with the following mean squared residue score $H(I, J)$, which is used as a measure of coherence of the bicluster:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2. \tag{1}$$

In (1), a_{ij} denotes the entry at the intersection of the ith row and jth column, a_{iJ} and a_{Ij} are the means of ith row and jth column, respectively, a_{IJ} is the mean of the submatrix A_{IJ}, and $|I|$ and $|J|$ denote the number of rows and number of columns, respectively, in the submatrix A_{IJ}. A small $H(I, J)$ score indicates that the rows fluctuate in unison, and in the financial data setup, it indicates comovement of a set of stocks. The ideal bicluster has score $H(I, J) = 0$ but corresponds to a trivial or constant bicluster, which is not desirable; therefore, in practice, we must set a threshold for $H(I, J)$. As mentioned in Section 1, the Cheng & Church (2002) algorithm allows us to eliminate any rows or columns,

Time Stock	T1	T2	T3	T4	T5	T6	T7	T8	T9
Stock1	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}
Stock2	r_{21}	r_{22}	r_{23}	r_{24}	r_{25}	r_{26}	r_{27}	r_{28}	r_{29}
Stock3	r_{31}	r_{32}	r_{33}	r_{34}	r_{35}	r_{36}	r_{37}	r_{38}	r_{39}
Stock4	r_{41}	r_{42}	r_{43}	r_{44}	r_{45}	r_{46}	r_{47}	r_{48}	r_{49}
Stock5	r_{51}	r_{52}	r_{53}	r_{54}	r_{55}	r_{56}	r_{57}	r_{58}	r_{59}
Stock6	r_{61}	r_{62}	r_{63}	r_{64}	r_{65}	r_{66}	r_{67}	r_{68}	r_{69}
Stock7	r_{71}	r_{72}	r_{73}	r_{74}	r_{75}	r_{76}	r_{77}	r_{78}	r_{79}

Figure 1. Cluster versus bicluster.

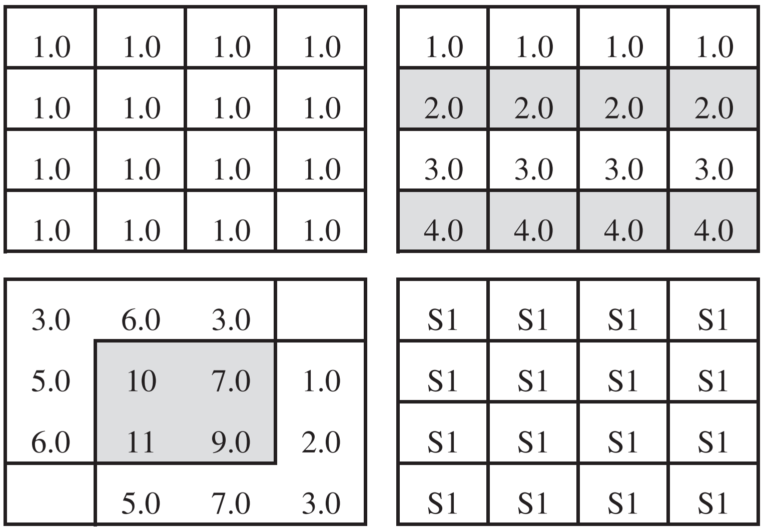


Figure 2. Four main types of bicluster.

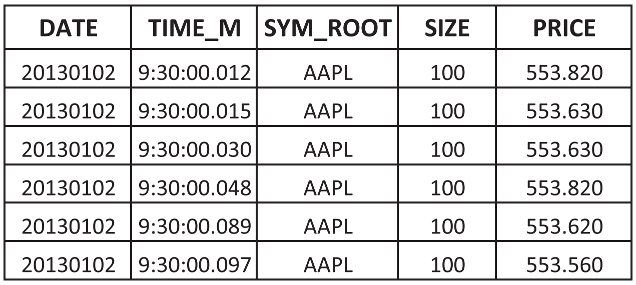
which results in non-continuous time points even when the columns constitute a continuous time interval. Zhang et al. (2005) extended the Cheng & Church (2002) algorithm by only allowing elimination of the start and end time points.

3 Data description

We use stock prices of the S&P100 index from 9:30 am to 4:00 pm on each trading day in 2013. The data are downloaded from Trade and Quote database of Wharton Research Data Services. Due to an adjustment of the S&P100 index, there were 104 stocks in 2013 (companies may be added to or removed from the S&P100 index on the basis of their performance). In order to have data for a complete year, we exclude stocks that were removed from the S&P100 index before December 31, 2013, or added to the index after January 1, 2013. We also remove stocks whose trading was suspended in 2013, ending up with 94 stocks for our data analysis. The raw data are very large (over 60 GB) and messy, so data preprocessing is necessary. We remove the transactions with the condition of special sales with any of the indicators including O, Z, B, T, L, G, W, J and K, because these special trading data are out of the scope of our analysis. We only keep the transactions with positive price and volume.

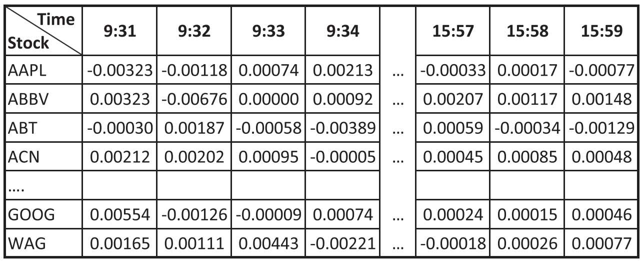
We first obtain 1-minute averages of the transaction-by-transaction stock prices. For example, the stock price at 9:30 am is the average of the prices from 9:30.00.001 am to 9:30.00.999 am. There were around one billion records of transaction-by-transaction stock prices from 9:30.00.001 am to 4:00.00.000 pm for the 94 stocks in 2013. Figure 3 shows an example of the data before the 1-minute averaging process. The challenge here is to efficiently construct 1-minute averages of stock prices for such a large data set. Our solution is to create a key for each transaction, which combines the system root, trading date and trading time. For example, Apple's stock price at 9:30.00.001 am on January 2, 2013, is assigned a key "APPL 20130102 9:30" so that all of Apple's stock prices have the same key from 9:30.00.001 am to 9:30.00.999 am on January 2, 2013. In this way, we group prices by the key to obtain the 1-minute averages quickly and efficiently. We then analyse the logarithms of the returns constructed from the averaged stock prices.

In our data frame, the row records are these log returns for each stock, and the columns are consecutive time points (minutes) from 9:31 am to 4:00 pm. There were 252 trading days in 2013. However, most stocks stopped trading



DATE	TIME_M	SYM_ROOT	SIZE	PRICE
20130102	9:30:00.012	AAPL	100	553.820
20130102	9:30:00.015	AAPL	100	553.630
20130102	9:30:00.030	AAPL	100	553.630
20130102	9:30:00.048	AAPL	100	553.820
20130102	9:30:00.089	AAPL	100	553.620
20130102	9:30:00.097	AAPL	100	553.560

Figure 3. The data set before 1-minute averaging.



Time / Stock	9:31	9:32	9:33	9:34		15:57	15:58	15:59
AAPL	-0.00323	-0.00118	0.00074	0.00213	…	-0.00033	0.00017	-0.00077
ABBV	0.00323	-0.00676	0.00000	0.00092	…	0.00207	0.00117	0.00148
ABT	-0.00030	0.00187	-0.00058	-0.00389	…	0.00059	-0.00034	-0.00129
ACN	0.00212	0.00202	0.00095	-0.00005	…	0.00045	0.00085	0.00048
….								
GOOG	0.00554	-0.00126	-0.00009	0.00074	…	0.00024	0.00015	0.00046
WAG	0.00165	0.00111	0.00443	-0.00221	…	-0.00018	0.00026	0.00077

Figure 4. Log returns of stock prices after preprocessing on January 2, 2013.

after 1:00 pm on July 3, November 29 and December 24 (because of Independence Day, Thanksgiving and Christmas Day). To keep our data consistent among all trading days, the transactions of these three trading days are excluded from our analysis, resulting in a data set of 94 stocks with 97,425 records of log returns over 249 trading days. Figure 4 illustrates a snapshot of the preprocessed data on January 2, 2013.

4 Multiple day time series biclustering

As mentioned previously, the CC-TSB algorithm implements static biclustering of data observed over consecutive times. For high-frequency financial data, this algorithm can be applied to obtain biclusters within a single trading day from 9:30 am to 4:00 pm. However, we have multiple day information on stocks, and our goal is to extract patterns over weeks, months or even years. Such intra-day stock prices over weeks, months or years cannot be viewed as a big data matrix as required by the CC-TSB algorithm, because the time intervals are no longer consecutive from one trading day to the next. The stock prices at the beginning of a new trading day must reflect the information from the previous night because this (discrete) time interval from 4:00 pm of any day until 9:30 am of the next trading day can significantly impact the comovement direction of stock prices between those 2 days. In order to deal with these issues, we propose a multiple trading day time series biclustering algorithm (CC-MDTSB).

Our algorithm is an extension of the CC-TSB algorithm and can deal with high-frequency intra-day data over multiple trading days. CC-MDTSB follows two major steps: first, identify the biclusters within each trading day; second, compute the probability of comovement for any m-tuple of stocks across a selected set of trading days. Let S denote the number of stocks in our universe ($S = 94$), T denote the number of selected trading days ($T = 249$) and N denote the number of selected 1-minute time intervals within each day ($N = 389$). For each of the T days, let $\mathbf{A}_1$ denote the $S \times N$ matrix of 1-minute averaged log returns. Our biclustering algorithm operates on the matrix $\mathbf{A}_1$ by step 2 successively deleting rows (stocks) and permitted columns (minutes) according to the absence of similarity between the relevant entries, followed by step 3 adding back rows and columns appropriately and step 4 repeating these steps a few times in order to determine an optimal number of biclusters (involving selected stocks over times) for that day. The output of the algorithm is the comovement probability for each chosen m-tuple of stocks across the selected trading days. Let θ, α and β be user-defined thresholds. For a single trading day, the algorithm consists of the following steps.

Step 1: Initial H score. For the $S \times N$ matrix of log returns $\mathbf{A}_1$, we compute the H score as H_1 using (1). Let $I = 1$ to S $(S = 94)$ and $J = 1$ to N $(N = 389)$.

Step 2: Row and column deletion. Start with the input matrix $\mathbf{A}_1$. The pair (I, J) will change dynamically when a row (stock) or a column (time point) is deleted.

2(a) For each $i \in I$, compute

$$R_i = \frac{1}{H(I,J)\ |J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2. \tag{2}$$

If $R_i \geq \alpha$, then row (stock) i is deleted from the row list, that is, the list of stocks (I). Let $\mathbf{A}_{2a}$ denote the resulting matrix after removing one or more stocks. For example, if five out of the 94 stocks are deleted, then $\mathbf{A}_{2a}$ will be an 89×389 matrix. Compute the H score of $\mathbf{A}_{2a}$ as H_{2a} using (1).

2(b) For each $j \in \{\min(J), \max(J)\}$, (i.e., we only need to consider two values: $j = \min(J)$ (the start time point) and $j = \max(J)$ (the end time point)), compute

$$C_j = \frac{1}{H(I,J)\ |I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2. \tag{3}$$

If $C_j \geq \alpha$, then the corresponding time point is deleted from the column list, that is, the list of times points (J). Suppose the resulting matrix is $\mathbf{A}_{2b}$. For example, if one time point is deleted, then $\mathbf{A}_{2b}$ will have dimension 89×388. Using (1), compute its H score as H_{2b}.

2(c) If $|H_1 - H_{2b}| < \theta$, stop the deletion process and go to step 3. Otherwise, update the H_1 score as H_{2b} and repeat steps 2(a)–2(c) with $\mathbf{A}_{2b}$ as input. Suppose the final output of the deletion process is $\mathbf{A}_3$.

Step 3: Insertion process. Start with input $\mathbf{A}_3$.

3(a) For each $j' \in \{\min(J) - 1, \max(J) + 1\}$, compute (3). If $C_{j'} < \alpha$, then insert this time point into the column list (J). Suppose the resulting matrix is $\mathbf{A}_{3a}$. Using (1), compute its H score as H_{3a}.

3(b) For each $i \notin I$, compute (2); if $R_i < \alpha$, then insert row (stock) i into the row list (I). Let $\mathbf{A}_{3b}$ denote the resulting matrix after inserting some stocks. Compute the H score of $\mathbf{A}_{3b}$ as H_{3b}.

3(c) If no row (stock) or column (time point) is inserted into the row or column list, stop the insertion process, and go to step 4. Otherwise, repeat steps 3(a) and 3(b). Suppose the final output of the insertion process is $\mathbf{A}_4$. This is the first identified bicluster.

Step 4. Identify the next bicluster: Eliminate the stocks in $\mathbf{A}_4$ from $\mathbf{A}_1$ and repeat steps 2–3. Let $\mathbf{A}_5$ denote the next identified bicluster.

Step 5. Stop search for biclusters: Let $\ell = 1, \ldots, L$ denote the number of identified biclusters for a trading day. Let S_ℓ denote the number of stocks in the ℓth bicluster within the trading day. Let U denote the total number of stocks (out of the S stocks) that belong in these L biclusters. If $U \leq \beta$, continue to repeat steps 1–4. Otherwise, stop identifying biclusters for that trading day. We repeat this process for each of the T trading days by repeating steps 1–5.

Parameter selection. The user-defined threshold α is the tolerance of insertion or deletion of the rows or columns. When α is large (small), the algorithm tends to give large (small) biclusters. We have used $\alpha = 1.2$ in our analysis, for which the distributions of the number and sizes of the biclusters are provided in Section 6.1. θ, the threshold of changes in the H score, serves as the stopping criterion in the deletion process. We have used $\theta = 1\mathrm{e}-12$. The threshold β enables us to stop identification of new biclusters within a trading day. Because it is not feasible to preset the number of biclusters for each trading day, the number of stocks explained by the biclusters is a natural choice to stop the algorithm. We have used $\beta = 80$ in our analysis.

Time complexity. Let S_ℓ denote the number of stocks in the ℓth bicluster within a trading day. Recall that S denotes the number of stocks, N denotes the number of time intervals and L denotes the number of identified biclusters for a trading day. If we fix the number of columns N, the time complexity of the CC-TSB algorithm is $\mathcal{O}(N^L S^L)$ (the order of the dimension of data matrix to the power of the number of biclusters) and that of our algorithm is $\mathcal{O}(N^L S \prod_{\ell=1}^{L-1}(S - \sum_{i=1}^{\ell} S_i)$, which is guaranteed to be faster than the CC-TSB algorithm. The ratio of time complexity of our algorithm to the CC-TSB algorithm is $\prod_{\ell=1}^{L-1}(1 - \sum_{i=1}^{\ell} S_i/S)$. Our algorithm can be much faster when S_ℓ is bigger for small ℓ; that is, the sizes of the biclusters that are identified earlier are relatively large. In the case that the size of each bicluster fixed at $S_\ell = S/L$, Figure 5 shows that the relative percentage of computing time of our algorithm over the CC-TSB algorithm decreases dramatically as the number of biclusters grows. When the number of biclusters is greater than six, the computing time of our algorithm is only 1% of the CC-TSB algorithm. For example, suppose $S_1 = 17$, $S_2 = 23$, $S_3 = 11$, $S_4 = 23$ and $S_5 = 8$; because $\sum_{\ell=1}^{5} S_\ell = 82$, the algorithm

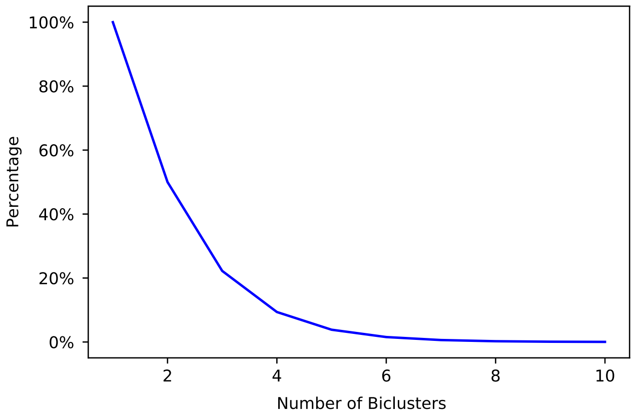


Figure 5. The relative percentage of computing time of CC-MDTSB over CC-TSB with the size of each bicluster fixed at S/L.

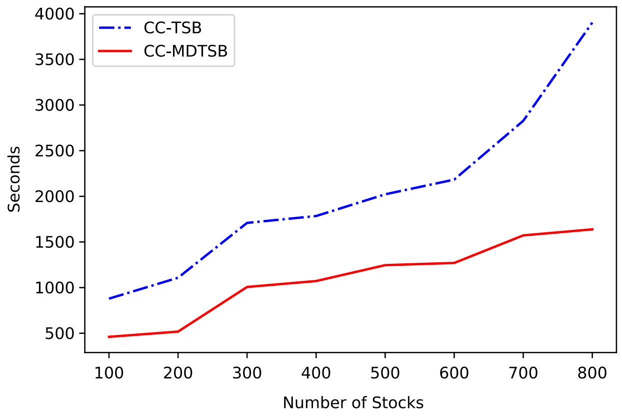


Figure 6. Computing time comparison of CC-MDTSB versus CC-TSB.

stops because the total number of identified stocks reaches the threshold of 80. The time complexity of CC-TSB algorithm is $\mathcal{O}(389^5 94^5)$. The time complexity of our algorithm is $\mathcal{O}(389^5(94)(94-17)(94-17-23)(94-17-23-11)(94-17-23-11-23))$. Our algorithm only spends 4.58% of the time taken by the CC-TSB algorithm, which saves a considerable amount of time. Figure 6 uses simulated data to show that our algorithm outperforms the CC-TSB algorithm for different dimensions. As the dimension increases, the two lines diverge, suggesting our CC-MDTSB algorithm is even faster in high-dimensional settings.

5 Forecasting comovement days

For a selected m-tuple of stocks (indexed by $F = \{f_1, f_2, \ldots, f_m\}$, for $m \geq 2$), let $w = 1, \ldots, W$ denote the calendar week, $T(w)$ denote the number of trading days in week w and $\tau_{\mathbf{F}}(w)$ denote the number of trading days for which the m-tuple falls within the same biclusters in week w. The comovement probability for an m-tuple of stocks in a given week w is computed as the proportion

$$\mathscr{P}_{\mathbf{F}}(w) = \frac{\tau_{\mathbf{F}}(w)}{T(w)}. \tag{4}$$

The cumulative comovement probability for an m-tuple *up to* a given week w_c is then computed as the proportion

$$P_{\mathbf{F}}(w_c) = \frac{\sum_{w=1}^{w_c} \tau_{\mathbf{F}}(w)}{\sum_{w=1}^{w_c} T(w)}. \tag{5}$$

For forecasting comovement days for any m-tuple of stocks, the values $P_{\mathbf{F}}(w_c)$ for $w_c = 1, \ldots, W$ are more stable than $\mathscr{P}_{\mathbf{F}}(w)$. We use $P_{\mathbf{F}}(w_c)$ for $w_c = 1, \ldots, G$ as our training sample and $P_{\mathbf{F}}(w_c)$ for $w_c = G+1, \ldots, W$ as our test sample. We employ the double exponential smoothing procedure (using HoltWinters::stats function in R) on the training data and forecast future values in the test sample by $\hat{P}_{\mathbf{F}}(w_c)$ for $w_c = G+1, \ldots, W$. Our goal is to predict $\tau_{\mathbf{F}}(w_c) = \sum_{w=1}^{w_c} \tau_{\mathbf{F}}(w)$ for the test period (see the numerator of (5)). We use algebra to solve the following identity:

$$\hat{P}_{\mathbf{F}}(w_c) = \frac{\sum_{w=1}^{G} \tau_{\mathbf{F}}(w) + \sum_{w=G+1}^{w_c} \hat{\tau}_{\mathbf{F}}(w)}{\sum_{w=1}^{G} T(w) + \sum_{w=G+1}^{w_c} T(w)} \tag{6}$$

to obtain

$$\hat{\tau}_{\mathbf{F}}(w_c) = \hat{P}_{\mathbf{F}}(w_c)\left(\sum_{w=1}^{G} T(w) + \sum_{w=G+1}^{w_c} T(w)\right) - \left(\sum_{w=1}^{G} \tau_{\mathbf{F}}(w) + \sum_{w=G+1}^{w_c-1} \hat{\tau}_{\mathbf{F}}(w)\right). \tag{7}$$

This enables us to infer the number of days over which a selected m-tuple of stocks comoves for any given w_c.

6 Results and discussion

In this section, we show biclustering results for 1-minute averaged raw returns and risk-adjusted returns. The data were described in Section 3. We also predict comovement probabilities of selected m-tuples of stocks in a given week and point out the connection between biclustering and diurnal effect. The comovement probabilities in this section are calculated using (5)–(7) for $W = 53$ (all the trading weeks in 2013), with two cases $m = 2$ (stock pairs) and $m = 3$ (trivariate stock groups).

6.1 Biclustering raw returns

We run our CC-MDTSB algorithm on the high-frequency data. Figure 7 indicates that the number of biclusters on each trading day ranges from 6 to 13, with most trading days having 8 or 9 biclusters. One interesting observation from our

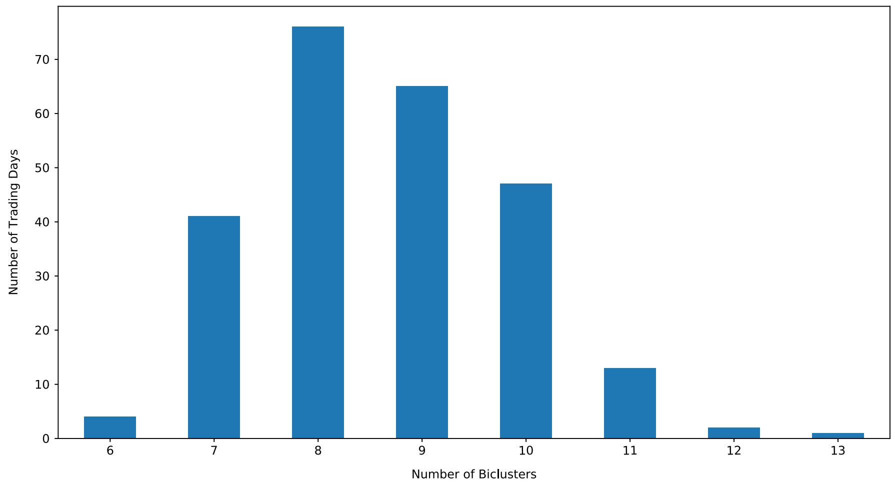


Figure 7. Distribution of number of biclusters.

Table I. Bivariate biclusters of full data.

Stock 1	Stock 2	$P_F(53)$ (%)
CVX	XOM	67.5
SO	AEP	55.8
PG	CL	54.6
JNJ	PEP	52.6
MMM	XOM	52.2
MMM	MCD	51.8
MMM	UPS	50.6
JNJ	MMM	50.2
NOV	SLB	49.4
CVX	MMM	48.6
JNJ	MCD	46.2
PEP	PG	45.8
APC	DVN	45.4
MMM	HON	45.0
JNJ	PG	44.2

results is that a few stocks in the financial sector, that is, BAC, C, COF, GS, JPM, MET, USB and WFC, do not belong to any identified bicluster on any trading day. This is probably due to their characteristics such as high liquidity and high volume.

Comovement probability. Our CC-MDTSB algorithm calculates the comovement probability for any m-tuple of stocks. Table I shows the top 15 pairs of stocks with the largest comovement probability, that is, the probability of trading days for which the two stocks are in the same bicluster (i.e., are comoving) in 2013. For example, the estimated comovement probability of two stocks in the (oil) energy sector, CVX (Chevron) and XOM (Exxon Mobil), is 0.675. As another example, JNJ (Johnson & Johnson) and PG (Procter & Gamble) move together on 44.2% of the trading days in 2013, possibly because they are both consumer-packaged goods companies.

Table II shows the top 15 trivariate stocks with the largest comovement probability. The comovement percentage of CVX (Chevron), MMM (Mining and Manufacturing Company) and XOM (Exxon Mobil), as a subgroup of energy companies, is 37.2% in 2013. The comovement probability of trivariate sets of stocks is overall much smaller than that of bivariate sets. It is expected that the comovement probability of more than three stocks will be even smaller. For brevity, we have only demonstrated the bivariate and trivariate cases in our results.

Biclusters and diurnal effect. Diurnal effect is an interesting stylized feature in the financial market, especially in high-frequency financial data, and refers to intra-day periodicity caused by trading patterns. Typically, trading is very active during the first half hour of a trading day. Volatility of stock prices is very high because of the information transmitted overnight. The trading activity tends to slow down as overnight information is digested over the day but picks up again when the market approaches the closing time and traders adjust their positions. Diurnal effect provides an explanation of why clustering algorithms are not appropriate or adequate for analysis of trading high-frequency financial data. The comovement pattern is disrupted by overnight information digestion and traders' activities of rebalancing their positions.

Our biclustering algorithm does not require taking all the time points into consideration when it studies similarities among stock prices, and it enables us to explore comovement of stocks in terms of stochastic patterns by reducing

Table II. Trivariate biclusters of full data.

Stock 1	Stock 2	Stock 3	$P_F(53)$ (%)
CVX	MMM	XOM	37.2
COP	CVX	XOM	30.6
PEP	PG	CL	28.3
JNJ	MMM	MCD	27.6
CVX	XOM	UPS	27.2
JNJ	PEP	PG	26.6
MMM	XOM	MCD	26.3
MMM	XOM	UPS	26.0
MMM	MCD	UPS	25.8
CVX	XOM	HON	25.6
JNJ	PG	CL	25.2
JNJ	MMM	XOM	24.6
MMM	UPS	HON	24.3
CVX	MMM	UPS	24.2
CVX	MMM	MCD	23.8

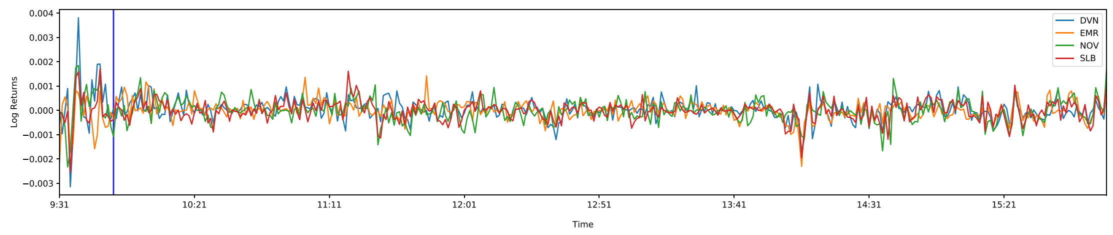


Figure 8. Comovement pattern of four stocks on January 3, 2013.

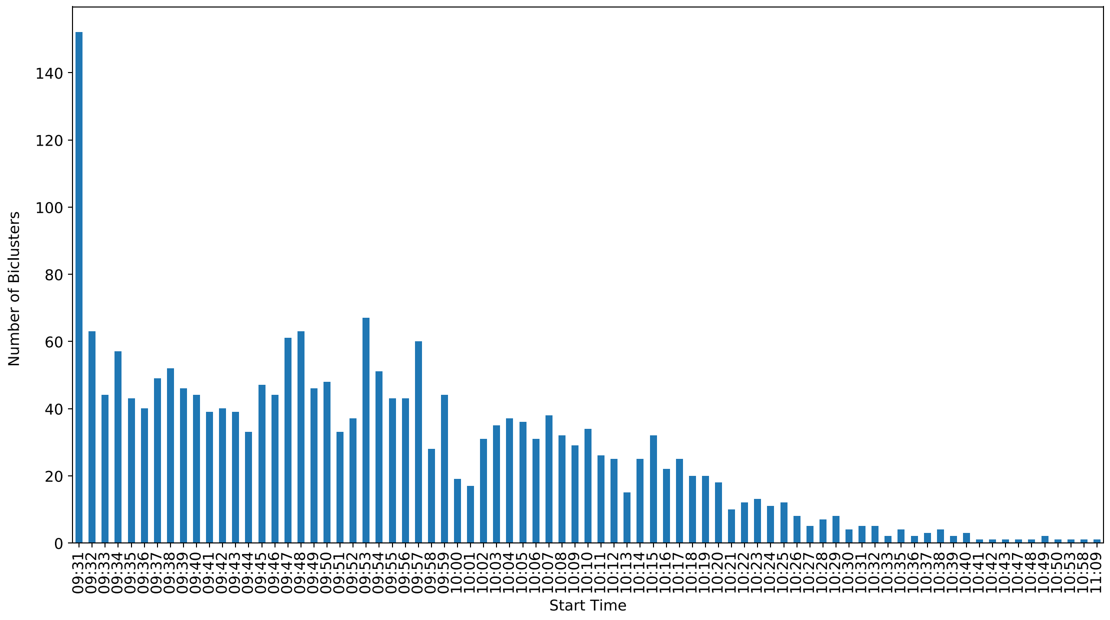


Figure 9. Start time distribution by number of biclusters.

the diurnal effect. Figure 8 shows the comovement pattern of one bicluster with four stocks on January 3, 2013. The area before the blue line represents the movement of the four stock prices during the first 20 minutes of the trading day, from 9:31 am to 9:50 am. This 20-minute time interval is eliminated by our CC-MDTSB algorithm because of very high volatility, and the four stock prices moved in very different directions during this period. This suggests that the comovement pattern was disrupted because of adjustment for overnight information, that is, the diurnal effect.

Figure 9 shows the distribution of the start time of biclusters over the 1-year period in 2013. We can see that most of the biclusters start a short period after the market opening. This illustrates that our CC-MDTSB algorithm is able to adjust for the diurnal effect without loss of information because of removal of the data from the opening period. Figure 10 shows the distribution of the end time of biclusters over the 1-year period in 2013. Only a small portion of the stocks stopped comovement with other stocks before 3:56 pm; this pattern could be due to the impact of traders' activity of rebalancing positions before the market closing.

Algorithm robustness. Some researchers exclude the data from the first half hour when analysing the high-frequency financial data because of the high level of noise in stock price movement. Here, we compare our analysis on the full data with that on the reduced data excluding transactions within the first half hour. Tables III and IV demonstrate that the top 15 sets of bivariate and trivariate stocks are very similar to those discovered in the full data. The comovement probability of CVX (Chevron) and XOM (Exxon Mobil) is still the highest. This indicates that our algorithm is robust to market noise from the diurnal effect. Moreover, our method is able to automatically adjust for the diurnal effect without the information loss because of removal of any period of data.

Forecasting comovement days. As an illustration, we use the stock pair CVX (Chevron) and XOM (Exxon Mobil) to predict the comovement days. Figure 11 shows that the double exponential smoothing procedure fits the data closely after 20 weeks. After 40 weeks, the fitted line and the raw data are almost overlapping. Table V shows the actual and predicted comovement probability up to each of the last 6 weeks. We can see that the predicted

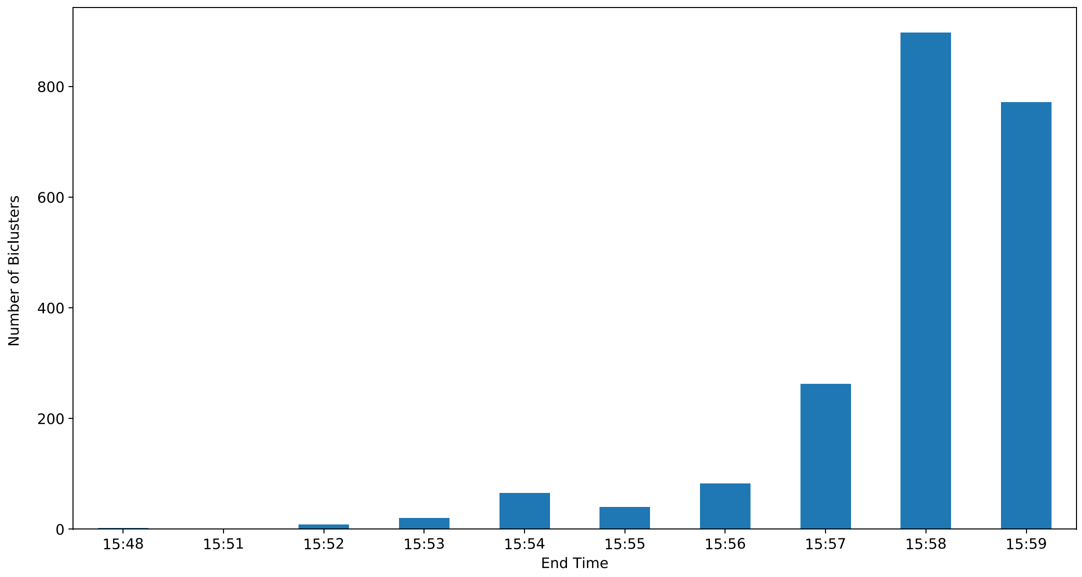


Figure 10. End time distribution by number of biclusters.

Table III. Bivariate biclusters of reduced data.

Stock 1	Stock 2	$P_F(53)$ (%)
CVX	XOM	70.7
PG	CL	58.6
JNJ	PEP	57.8
JNJ	MMM	57.4
MMM	MCD	57.0
MMM	UPS	57.0
SO	AEP	56.6
JNJ	MCD	55.8
MMM	XOM	55.8
CVX	MMM	55.4
MMM	HON	50.6
UPS	HON	49.4
CVX	UPS	48.2
COP	CVX	47.8
PG	PEP	47.8

probability is close to the actual probability. The estimated number of comovement trading days in the last 6 weeks using (7) is 11, which is the same as the actual number. Our prediction is satisfactory.

6.2 Biclustering volatility-adjusted returns

The 1-minute averaged log returns incorporate the volatility in the stock prices. The volatility-adjusted log returns represent the risk-adjusted asset log returns relative to the amount of risk the investment has taken over a given period

Table IV. Trivariate biclusters of reduced data.

Stock 1	Stock 2	Stock 3	$\mathbf{P}_{\mathbf{F}}(\mathbf{53})$ (%)
CVX	MMM	XOM	43.2
JNJ	MMM	MCD	37.1
CVX	XOM	UPS	34.9
COP	CVX	XOM	34.9
CVX	XOM	HON	34.5
JNJ	MMM	PEP	34.0
CVX	MMM	UPS	33.5
CVX	JNJ	XOM	33.2
MMM	XOM	UPS	33.0
JNJ	MCD	PEP	32.9
CVX	XOM	MCD	32.6
MMM	MCD	UPS	32.4
MMM	UPS	HON	32.3
MMM	XOM	MCD	31.9
JNJ	MMM	UPS	31.5

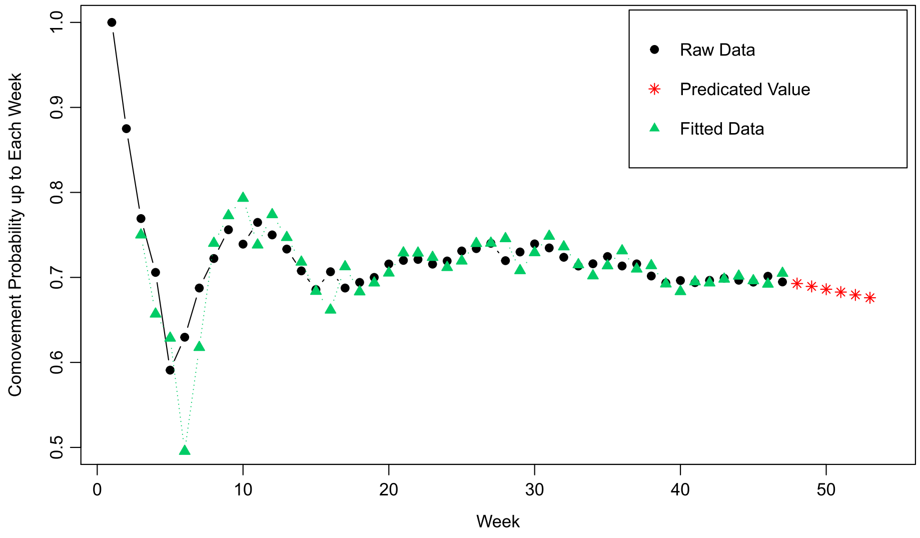


Figure 11. Holt–Winters double exponential smoothing of comovement probability of CVX and XOM up to each week.

of time. Let r_{ij} denote the averaged log return, σ_{ij} denote the volatility of stock i at time (minute) j, for $i = 1, \dots, S$ and $j = 1, \dots, N$, and n_{ij} be the number of transactions in the time interval $(j-1, j]$. The volatility-adjusted log return for stock i at time j is defined as

$$r_{\text{adj}\_ij} = \frac{r_{ij}}{\sigma_{ij}\sqrt{n_{ij}}}. \tag{8}$$

For example, the log return of APPL at 9:33 am is 0.00074, obtained by averaging the log returns of each second from 9:32 am to 9:33 am. The volatility of the log return of APPL from 9:32 am to 9:33 am is 0.000197, computed from

Table V. Actual versus predicted comovement probability up to each week.

Week w	$\mathbf{P}_{\mathbf{F}}(w)$ (%)	$\hat{\mathbf{P}}_{\mathbf{F}}(w)$(%)
48	69.87	69.28
49	68.80	68.94
50	69.04	68.61
51	68.44	68.27
52	68.02	67.94
53	67.47	67.61

Table VI. Bivariate volatility of adjusted log return.

Stock 1	Stock 2	$\mathbf{P}_{\mathbf{F}}(\mathbf{53})$ (%)
INTC	F	67.87
CSCO	F	61.45
CSCO	INTC	59.44
COP	XOM	57.43
JNJ	XOM	56.22
QCOM	XOM	54.22
KO	MO	53.82
CVX	XOM	51.41
FCX	XOM	51.41
COP	CVX	51.00
PFE	T	49.80
CAT	CVX	49.00
FCX	JNJ	48.19
AAPL	XOM	47.79
JNJ	QCOM	46.59

60 transactions. Because the unit of observation for this volatility is a second, we convert it to volatility in minutes by multiplying by the square root of 60, which is the number of transactions between 9:32 am and 9:33 am, to yield the volatility-adjusted return of $\frac{0.00074}{0.000197*\sqrt{60}} = 0.486$. We then apply our CC-MDTSB algorithm to the volatility-adjusted log returns.

Table VI shows the results for the top 15 pairs of stocks based on volatility-adjusted log returns. The pair of CVX (Chevron) and XOM (Exxon Mobil) again makes it to the top of the list. We also discover an interesting comovement pattern between CSCO (Cisco) and INTC (Intel) using the risk-adjusted returns. Biclustering and comovement patterns may be different after the volatility adjustment because of the extra information in the risk profiles of different assets.

7 Conclusion

High-frequency transaction-by-transaction data are widely available to investors and researchers with recent rapid technological development. We propose a multiple day time series biclustering (CC-MDTSB) algorithm in order to explore the microstructure of variability in stock prices on transaction-level intra-day data and to dynamically study

Algorithm 1 Multiple Day Time Series Biclustering Algorithm (CC-MDTSB)

Input: Stocks: $i = 1, \ldots, S$. Trading Days (blocks): $t = 1, \ldots, T$. Number of one-minute averages in a day: $j = 1, \ldots, N$. $\mathbf{A}$ is the $S \times N$ data matrix of the intra-day log returns of each stock. U is the number of stocks explained by the identified biclusters. α is the empirical threshold for deletion and insertion. β is the empirical threshold for S. θ is the empirical threshold for H score changes.

Output: $P_{\mathbf{F}}(w)$, for a given w.

```
1:  for number of trading days from 1 to $T$ do
2:      $I = 1$ to $S$.
3:      $J = 1$ to $N$.
4:      while $U \leq \beta$ do
5:          while The changes of $H$ score $\geq \theta$ do
6:              if $R_i \geq \alpha$, delete stock $i$ from $I$ and update $H$ score.
7:              if $C_j \geq \alpha$ for $j \in [\min(J), \max(J)]$, delete the time point $j$ from $J$ and update $H$ score.
8:          end while
9:          while some rows or columns inserted do
10:             if $C_j < \alpha$ for $j' \in [\min(J) - 1, \max(J) + 1]$, insert the time point $j'$ into $J$ and update $H$ score.
11:             if $R_i < \alpha$ for each $i \notin I$, add the stock $i$ into $I$ and update $H$ score.
12:         end while
13:         Report the $\ell$th bicluster as $\mathbf{A}(I, J)$.
14:         Assign the key $t\_\ell$ to $\mathbf{A}(I, J)$. $t$ is the trading date, $\ell$ is the bicluster number.
15:         Eliminate the stocks for $i \in I$ from $\mathbf{A}$.
16:     end while
17: end for
18: for m-tuple of stocks ($\mathbf{F}$) do
19:     Initial $P_{\mathbf{F}}(w) = 0$
20:     if $\mathbf{F} \in t\_\ell$ then
21:         $P_{\mathbf{F}}(w) + +$
22:     end if
23: end for
```

patterns of comovement over multiple trading days. Our algorithm is an extension of the CC-TSB algorithm of Zhang et al. (2005). For identifying biclusters within each trading day, our algorithm provides a faster alternative to the random replacement method in the CC-TSB algorithm. Another advantage of our algorithm is that it does not require prespecification of the number of biclusters for each trading day. Instead, we set a threshold on the number of stocks within the biclusters to yield a stopping criterion for multiple day analysis.

The computing time of our algorithm is guaranteed to be smaller than CC-TSB, which uses random replacement to identify excluded stocks. Instead, we implement a method that systematically reduces matrix dimension by eliminating submatrices corresponding to excluded stocks and times. The time saving is even more manifest in high-dimensional settings. In addition, we propose a fast and effective non-parametric double exponential smoothing procedure to estimate the comovement probability of any m-tuple of stocks conditional on the other stocks within the dynamic biclusters. Moreover, our method is also robust to the noise caused by the diurnal effect because of overnight information and traders' activities of rebalancing their positions.

(wileyonlinelibrary.com) https://doi.org/10.1002/sta4.176

The ISI's Journal for the Rapid Dissemination of Statistics Research

References

Babu, MS, Geethanjali, N & Satyanarayana, B (2012), 'Clustering approach to stock market prediction', *International Journal of Advanced Networking and Applications*, **3**(4), 1281–1291.

Chekouo, T, Murua, A & Raffelsberger, W (2015), 'The Gibbs-plaid biclustering model', *The Annals of Applied Statistics*, **9**(3), 1643–1670.

Cheng, Y & Church, GM (2000), Biclustering of expression data, in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, *AAAI Press,* Menlo Park, California, 2000:93–103.

Darie, M & Cosmin, SG (2009), 'A clustering of DJA stocks—the application in finance of a method first used in gene trajectory study', *Annals of the University of Oradea: Economic Science*, **4**(1), 1006–1011.

Dose, C & Cincotti, S (2005), 'Clustering of financial time series with application to index and enhanced index tracking portfolio', *Physica A: Statistical Mechanics and its Applications*, **355**(1), 145–151.

Hartigan, JA (1972), 'Direct clustering of a data matrix', *Journal of the American Statistical Association*, **67**(337), 123–129.

Harvill, JL, Kohli, P & Ravishanker, N (2016), 'Clustering nonlinear, nonstationary time series using BSLEX', *Methodology and Computing in Applied Probability*, **19**(3), 935–955.

Huang, QH (2011), 'Discovery of time-inconsecutive co-movement patterns of foreign currencies using an evolutionary biclustering method', *Applied Mathematics and Computation*, **218**(8), 4353–4364.

Laclau, C & Nadif, M (2017), 'Diagonal latent block model for binary data', *Statistics and Computing*, **27**(5), 1145–1163.

Lee, M, Shen, H, Huang, JZ & Marron, JS (2010), 'Biclustering via sparse singular value decomposition', *Biometrics*, **66**(4), 1087–1095.

Lee, S & Huang, JZ (2014), 'A biclustering algorithm for binary matrices based on penalized Bernoulli likelihood', *Statistics and Computing*, **24**(3), 429–441.

Manahov, V, Hudson, R & Gebka, B (2014), 'Does high frequency trading affect technical analysis and market efficiency? And if so, how?', *Journal of International Financial Markets, Institutions and Money*, **28**, 131–157.

Mankad, S & Michailidis, G (2014), 'Biclustering three-dimensional data arrays with plaid models', *Journal of Computational and Graphical Statistics*, **23**(4), 943–965.

Menkveld, AJ (2013), 'High frequency trading and the new market makers', *Journal of Financial Markets*, **16**(4), 712–740.

O'Hara, M (2015), 'High frequency market microstructure', *Journal of Financial Economics*, **116**(2), 257–270.

Pontes, B, Giráldez, R & Aguilar-Ruiz, JS (2015), 'Biclustering on expression data: a review', *Journal of Biomedical Informatics*, **57**, 163–180.

Sheng, Q, Moreau, Y & De Moor, B (2003), 'Biclustering microarray data by Gibbs sampling', *Bioinformatics*, **19**(suppl_2), ii196–ii205.

Tan, KM & Witten, DM (2014), 'Sparse biclustering of transposable data', *Journal of Computational and Graphical Statistics*, **23**(4), 985–1008.

Tola, V, Lillo, F, Gallegati, M & Mantegna, RN (2008), 'Cluster analysis for portfolio optimization', *Journal of Economic Dynamics and Control*, **32**(1), 235–258.

Wyse, J & Friel, N (2012), 'Block clustering with collapsed latent block models', *Statistics and Computing*, **22**(2), 415–428.

Xue, Y, Liu, Z, Luo, J, Ma, Z, Zhang, M, Hu, X & Kuang, Q (2015), 'Stock market trading rules discovery based on biclustering method', *Mathematical Problems in Engineering*, **2015**(1), 1–13.

Zhang, J (2010), 'A Bayesian model for biclustering with applications', *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **59**(4), 635–656.

Zhang, Y, Zha, H & Chu, CH (2005), A time-series biclustering algorithm for revealing co-regulated genes, in *Proceedings of International Conference on Information Technology: Coding and Computing, 2005, IEEE,* Las Vegas, NV, USA, Vol. 1, 32–37.

Zou, J, An, Y & Yan, H (2015), Volatility matrix inference in high-frequency finance with regularization and efficient computations, in *2015 IEEE International Conference on Big Data (Big Data)*, *IEEE*, Santa Clara, CA, USA, 2437–2444.

Zou, J & Huang, C (2016), Efficient portfolio allocation with sparse volatility estimation for high-frequency financial data, in *2016 IEEE International Conference on Big Data (Big Data)*, *IEEE*, Washington D.C., USA, 2332–2341.