Sluggish news reactions: A combinatorial approach for synchronizing stock jumps*

Nabil Bouamara†
Department of Accounting, Finance and Insurance, KU Leuven
Solvay Business School, Vrije Universiteit Brussel

Kris Boudt
Department of Economics, Ghent University
Solvay Business School, Vrije Universiteit Brussel
School of Business and Economics, Vrije Universiteit Amsterdam

Sébastien Laurent
Aix-Marseille University (Aix-Marseille School of Economics)
CNRS & EHESS
Aix-Marseille Graduate School of Management-IAE

Christopher J. Neely
Research Division, Federal Reserve Bank of St. Louis

November 17, 2021

Abstract

Sluggish news reactions manifest as gradual jumps and jump delays. In a panel of high-frequency intraday stock returns, these noisy jumps show up as a "sluggish cojump", *i.e.* jumps observed at close but distinct points in time. We synchronize these scattered jumps to recover the true common jump component. We apply our methods to investigate the local behavior of Dow 30 stock jumps in a short event window around an ETF jump.

Keywords: Asynchronicity; Cojumps; Combinatorics; High-frequency; Rearrangement

*Bouamara gratefully acknowledges support from the Flemish Research Foundation (FWO PhD fellowship #11F8419N) and the Platform for Education and Talent (Gustave Boël – Sofina fellowship). The R code to generate the toy examples is available upon request.

†Corresponding author: KU Leuven, Faculty of Economics and Business, Naamsestraat 69, 3000 Leuven, Belgium. Email: nabil.bouamara@kuleuven.be

1 Introduction

Cojump tests implicitly assume that jumps manifest synchronously across all observed stock prices. While transaction prices impound jumps asynchronously, the main response thus far has been to use a coarse sampling grid to guard against microstructure effects (see *e.g.*, Bollerslev, Law, and Tauchen, 2008; Lahaye, Laurent, and Neely, 2011; Li, Todorov, Tauchen, and Lin, 2019). Such a grid is restrictive in that it oversmooths actual changes and blurs the efficient common jump.

We offer an alternative strategy that synchronizes asset price jumps and recovers the efficient common jump. We synchronize the jumps in an ETF with the corresponding cojumps in the underlying stocks. The ETF tracks the index and should closely follow the true, latent value of a basket of stocks. Stock prices, however, vary in how quickly they include new information, because trading impounds that information and trading frequency varies between stocks. The asynchronicity in the impoundment of news causes the price of a synthetic stock portfolio to deviate from the price of the ETF. If the ETF price efficiently tracks the index, however, we can use the spread between the ETF price and a synthetically constructed index to measure the collective misalignment of the observed stock prices with their efficient levels. We think that it is likely that the ETF does efficiently track the index as it is a highly liquid and carefully watched asset.

To synchronize the stock jumps, we formulate the rearrangement of jumps in time as a combinatorial problem. Our approach is rooted in and extends the pioneering work of Puccetti and Rüschendorf (2012) and Embrechts, Puccetti, and Rüschendorf (2013) on rearrangements and the rearrangement algorithm. This algorithm is best known as an actuarial tool to compute bounds on portfolio risk measures, but it also has applications in other disciplines, such as operations research (see *e.g.*, Boudt, Jakobsons, and Vanduffel, 2018). Rearrangements can also effectively synchronize stock jumps and recover the efficient common jump, provided we constrain them from profligacy. Our rearrangement linear program imposes penalties to prevent rearrangements that are economically implausible. For example, we penalize large moves or the penalty might be asymmetric, with a greater penalty for moving them later in time than earlier. The approach is similar in spirit to the Lasso (Tibshirani, 1996) in that it penalizes complexity in the form of moving jumps.

The best rearrangement fixes observed, sluggish prices to correspond to how unobserved, efficient prices should behave. Researchers have tried to explain why market participants

are slow to incorporate news in the observed prices, suggesting infrequent trading and the corresponding Epps (1979) effect, trading frictions (Cohen et al., 1980), behavioural biases such as conservativeness (Edwards, 1982), different time resolutions of market participants (Corsi, 2009), asymmetry in information and magnitude of execution costs (Bandi, Pirino, and Renò, 2017), information acquisition and processing delays (Andersen, Li, Cebiroglu, and Hautsch, 2021), difficult-to-process information and investor inattention (Cohen, Malloy, and Nguyen, 2020), as well as others. In any case, jumps can be gradual (Barndorff-Nielsen, Hansen, Lunde, and Shephard, 2009) and jumps of individual assets typically follow those of the highly liquid market index during market-wide events (Li, Todorov, Tauchen, and Chen, 2017). The standard jump-diffusion model for stock prices (see Aït-Sahalia and Jacod, 2014) does not capture such complications. We model sluggish news reactions by extending the standard DGP with an additional noisy jump component. Rearrangements recover the efficient common jump in a simulation experiment. We also show the benefits of our approach using the example of a difficult-to-interpret FOMC statement.

We proceed as follows. Section 2 details the way we synchronize jumps using a simulated toy example. Section 3 illustrates an empirical example of a rearranged sluggish cojump in the Dow 30. Section 4 concludes and discusses future research.

2 Synchronizing jumps: A combinatorial problem

This section details the way we synchronize stock jumps. We formulate a data generating process that features gradual jumps and jump delays. We optimally rearrange jumps, penalizing rearrangements that are economically implausible. We use a simulated example to clarify the mechanics behind the rearrangements.

2.1 A DGP for sluggish news reactions

We formulate a data generating process for the sluggish prices of the stocks in the index. The price process features gradual jumps and jump delays.

2.1.1 The efficient stock price

We start by characterizing the efficient price process of the p stocks in the index. Let $\boldsymbol{X}_t = (X_{1,t}, ..., X_{p,t})^\top$ denote the logarithmic p-variate efficient stock price process. The

price process is defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F})_{t\geq 0}, \mathbb{P})$ and is adapted to the filtration $\mathcal{F}_t$ that represents information available to market participants at time t, with $t \geq 0$. We assume that $\boldsymbol{X}$ operates in an arbitrage-free, frictionless market, which implies that $\boldsymbol{X}$ is a semimartingale. Econometricians (see Aït-Sahalia and Jacod, 2014) model the stock prices $\boldsymbol{X}$ as a jump-diffusion, which includes a continuous Brownian component and a jump component:

$$\begin{aligned} \boldsymbol{X}_t &= \boldsymbol{X}_t^c + \boldsymbol{X}_t^d, \text{ with,} \\ \boldsymbol{X}_t^c &\equiv \boldsymbol{X}_0 + \int_0^t \boldsymbol{b}_s ds + \int_0^t \boldsymbol{\sigma}_s d\boldsymbol{W}_s, \\ \boldsymbol{X}_t^d &\equiv \textstyle\sum_{s\leq t} \Delta \boldsymbol{X}_s, \end{aligned} \tag{1}$$

in which $t \geq 0$, $\boldsymbol{b}$ is the drift process, $\boldsymbol{\sigma}$ is the stochastic (co)volatility process, $\boldsymbol{W}$ is a multivariate Brownian motion, and $\Delta \boldsymbol{X}_t \equiv \boldsymbol{X}_t - \boldsymbol{X}_{t-}$, with $\boldsymbol{X}_{t-}$ the left limit at time t, denotes the jumps of $\boldsymbol{X}$ at time t. A stock's growth prospects generates a jump in the stock price. Big news and major economic events generate jumps in multiple stock prices.

2.1.2 The observed stock price

In practice we do not observe the efficient price process in continuous time. Instead we observe discretely sampled, noisy prices. Microstructure effects, such as tick size, discrete observations, bid-ask spreads, adverse selection, liquidity and inventory control produce market microstructure noise (see *e.g.*, Christensen, Oomen, and Podolskij, 2014; Lee and Mykland, 2012). Observed prices are also sluggish. When news arrives, it takes some time for the market to reach a consensus about the new equilibrium level, creating another type of noise. News is incorporated asynchronously into different asset prices through transactions.

We model the observed log-price process $\boldsymbol{Y}_t = (Y_{1,t}, ..., Y_{p,t})^\top$ of the p stocks as the sum of a contaminated Brownian component and a contaminated jump component:

$$\begin{aligned} \boldsymbol{Y}_{i\Delta_n} &= \boldsymbol{Y}_{i\Delta_n}^c + \boldsymbol{Y}_{i\Delta_n}^d, \text{ with,} \\ \boldsymbol{Y}_{i\Delta_n}^c &\equiv \boldsymbol{X}_{i\Delta_n}^c + \boldsymbol{u}_{i\Delta_n} \\ \boldsymbol{Y}_{i\Delta_n}^d &\equiv \textstyle\sum_{h\Delta_n \leq i\Delta_n} \Delta \boldsymbol{Y}_{h\Delta_n}. \end{aligned} \tag{2}$$

There are two kinds of noise. Microstructure noise $\boldsymbol{u}$ contaminates the efficient price process $\boldsymbol{X}$, but is typically too small to contaminate the discontinuous part $\boldsymbol{X}^d$. It cannot generate gradual jumps, nor can it generate jump delays. We capture the mistimed or mismeasured

jumps in a separate noisy jump component $\boldsymbol{Y}^d$, which emulates a sluggish news reaction by spreading the jump across several time intervals.

The observed price process (2) combines the frictions and the sampling frequency. We sample discretely at equispaced intervals $i\Delta_n$, with $i = 0, ..., \lfloor T/\Delta_n \rfloor$, across some fixed time span T, in which $\lfloor \cdot \rfloor$ denotes the floor function. There is less noise at a lower frequency Δ_n (like in Bollerslev et al., 2008; Lahaye et al., 2011; Li et al., 2019) but data at such lower frequencies tend to oversmooth actual changes and blurs the efficient common jump. This article synchronizes the asynchronous jumps in the observed prices to recover the efficient common jump.

2.1.3 A simulated example

We simulate data from the new DGP to clarify the mechanics behind our rearrangement procedure. We simulate second-by-second ($\Delta_n = 1/23,401$) log-prices for 1 stock ($p = 1$), $X_{i\Delta_n}$, across one trading day ($T = 1$) from a jump-diffusion model, with zero drift and constant variance, $\sigma_t^2 = 0.039$ corresponding to an annualized return volatility of about 20%. We contaminate the efficient price process with microstructure noise and a sluggish news reaction.

We draw the efficient jump process from a compound Poisson process:

$$X^d_{i\Delta_n} \equiv \sum_{j=1}^{N^J} I_{\{U_j \cdot T \leq i\Delta_n\}} \Delta X_j, \tag{3}$$

in which $I(\cdot)$ is an indicator function, N^J governs the total number of jumps that occur across a day, U_j are the random arrival times of the jumps and the sequence of normally distributed random variables, $\Delta X_1, ..., \Delta X_{N^J}$, are the corresponding jump sizes.

We add *i.i.d.* microstructure noise u, with $E[u] = 0$ and $E[u^2] = \omega^2$, in which we select ω^2 by fixing the noise ratio to $\gamma = (n\omega^2 / \int_0^1 \sigma_s^2 ds)^{1/2} = 0.5$, to contaminate the efficient price (Christensen et al., 2014).

The new information which generates the jump in the efficient price is not immediately impounded in the stock's observed price. To simulate such a sluggish news reaction, we draw a contaminated jump process that spreads each efficient jump across multiple time intervals:

$$Y^d_{i\Delta_n} \equiv \sum_{j=1}^{N^J} I_{\{U_j \cdot T \leq i\Delta_n\}} \sum_{d=0}^{N^D_j} I_{\{W_{j,d} \cdot T \leq i\Delta_n\}} \Delta L_{j,d} \, \Delta X_j, \tag{4}$$

in which N^J governs the total number of jumps that occur across a day, U_j are the random arrival times of the jumps, the sequence of normally distributed random variables, $\Delta X_1, ..., \Delta X_{N^J}$, are the corresponding jump sizes, and, for each jump j, a step function spreads each jump size across multiple time intervals; N_j^D governs the total number of steps in the step function, $W_{j,0}, ..., W_{j,N_j^D}$ are the step arrival times and $\Delta L_{j,0}, ..., \Delta L_{j,N_j^D}$ are increments in the step sizes.

Each stock jump has its own learning step process, $L_{j,i\Delta_n}$, which captures the speed with which observed prices incorporate new information. The learning variable rises from 0 to 1 as information about the efficient jump is fully incorporated in the stock price. The information increments $\Delta L_{j,d}$ add chunks of the efficient jump size ΔX_j to the observed jump process Y^d.

Suppose a jump j arrives at the random arrival time U_j with a jump size ΔX_j (3). To simulate the learning step process (4) for this particular jump, $L_{j,i\Delta_n}$, we first draw the number of steps, N_j^D, from a binomial distribution: $N_j^D \sim \text{Bin}(\text{Number of trials} = 5,\ \text{Success probability} = 0.4)$, for j fixed.

Each of the steps has a random width. We draw the waiting times between the steps from an exponential distribution: $w_{j,d} \sim \lceil \text{Exp}[\text{Rate} = 1/(15\, N_j^D)] \rceil$, for j fixed and $d = 1, ..., N_j^D$, in which $\lceil \cdot \rceil$ is the ceiling operator. The step arrival times $W_{j,d}$ in the indicator function of the compound information process are the cumulative sums of the waiting times starting at the arrival time of the efficient jump U_j: $W_{j,0} \cdot T \equiv U_j \cdot T$; $W_{j,1} \cdot T \equiv U_j \cdot T + w_{j,1}$; $\ldots$; $W_{j,N_j^D} \cdot T \equiv U_j \cdot T + w_{j,1} + ..., w_{j,N_j^D}$. The starting point is the arrival time of the efficient jump. The end point tells us when the information has been fully impounded. The total delay with which the efficient stock jump is impounded in the observed price is the sum of the waiting times: $D_j = \sum_{d=1}^{N_j^D} w_{j,d}$. If it takes more steps to incorporate the jump, the total delay will be longer.

The steps also have a random size that correspond to accumulated information impounded in the observed price. To extract the sizes $L_{j,d}$ of each step d, with $d = 0, ..., N_j^D$, we sample realizations from a Brownian bridge according to the step arrival times, $W_{j,0}, ..., W_{j,N_j^D}$. The Brownian bridge $\Lambda_{j,t}$ is a continuous-time, stochastic process which is pinned at both ends of the delay interval, at $t = U_j \cdot T$ and $t = U_j \cdot T + D_j$:

$$\Lambda_{j,t} = B_{j,t} - \frac{t}{U_j \cdot T + D_j} \left(1 - B_{j,U_j \cdot T + D_j)}\right), \qquad t \in [U_j \cdot T, U_j \cdot T + D_j]$$

in which $B_{j,t}$ is a standard, univariate Wiener process, with $B_{j,0} = 0$. A standard Wiener

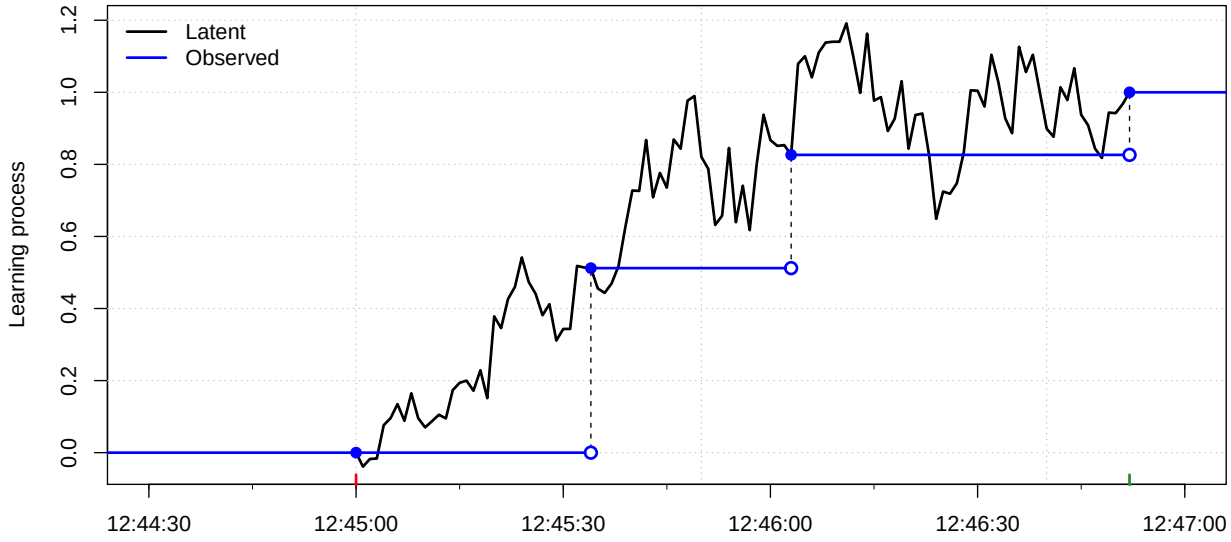
process is tied down to the origin, but the other points are not restricted. The Brownian bridge is pinned at both ends. Just as pylons support a literal bridge, the pylons in the Brownian bridge make sure that the process evolves from the first pylon $\Lambda_{j,U_j \cdot T} = 0$ to the second pylon $\Lambda_{j,U_j \cdot T + D_j} = 1$. The Brownian bridge resembles the latent learning process. When news is released, the market processes and accumulates information, including under- and overreactions. After some time the information is fully impounded in the price.

The Brownian bridge gradually moves from 0 to 1 – non-monotonically – but we observe its values at discrete intervals. Investors impound big chunks of new information in the observed price at intermittent intervals. We extract the discrete learning process, $L_{j,d}$, with $d = 0, 1, ..., N_j^D$, by sampling the step sizes from the Brownian bridge process $\Lambda_{j,t}$ at discrete points, $W_{j,0}, W_{j,1}, ..., W_{j,N_j^D}$. Because we accumulate new information in the price, we use the increments in the draws, $\Delta L_{j,d}$, to construct the indicator function in (4).

Figure 1 plots an example of a stochastic learning process for one stock jump delay. The efficient stock jump arrives at $i\Delta_n = 11,701$ or 12:45:00. The waiting times between each of three steps are equal to $w_{1,1} = 35, w_{1,2} = 29, w_{1,3} = 49$ seconds. In the following $D_j = 112$ seconds, the Brownian bridge (in black) process evolves from 0 to 1. If $\Lambda_{j,i\Delta_n} < 1$, the observed price jump underreacts and does not (completely) incorporate the new information. If $\Lambda_{j,i\Delta_n} > 1$, the observed price overreacts to the jump. At the end of the interval, when $\Lambda_{j,i\Delta_n} = 1$, the observed stock jump equals the efficient stock jump. We do not observe this learning process in continuous time. Rather, we sample the step sizes at the waiting times, leading to the step function (in blue). The waiting times relate to the step widths and are equal to $W_{1,0} \cdot T = 11,701 + 0$, $W_{1,1} \cdot T = 11,701 + 34$, $W_{1,2} \cdot T = 11,701 + 63$ and $W_{1,3} \cdot T = 11,701 + 112$. The sizes are equal to $L_{1,0} = 0.000$, 0.512, 0.826, and 1.000. The information increments are equal to $\Delta L_{1,0} = 0.000$, $\Delta L_{1,1} = 0.512$, $\Delta L_{1,2} = 0.314$, and $\Delta L_{1,3} = 0.174$. The information increments should sum to 1.

Figure 2 shows that learning can manifest as a gradual jump in the observed price. The observed price sluggishly catches up with the new equilibrium level in the top panel. The other panels decompose the price process into its continuous Brownian component and its jump component. The middle panel compares the efficient continuous price with the one contaminated by market microstructure noise. The bottom panel compares the efficient, sudden jump with the contaminated, gradual jump.

Figure 1: A learning step process when news arrives



2.2 The jump-event matrix

We investigate the relationship between jumps in an ETF and the corresponding cojumps in the underlying stocks. When multiple stocks react sluggishly to new information, the jumps are asynchronous across stocks and such jumps will generally not coincide with jumps in the ETF. We construct a jump-event matrix to take a closer look at the local behavior of the asset jumps.

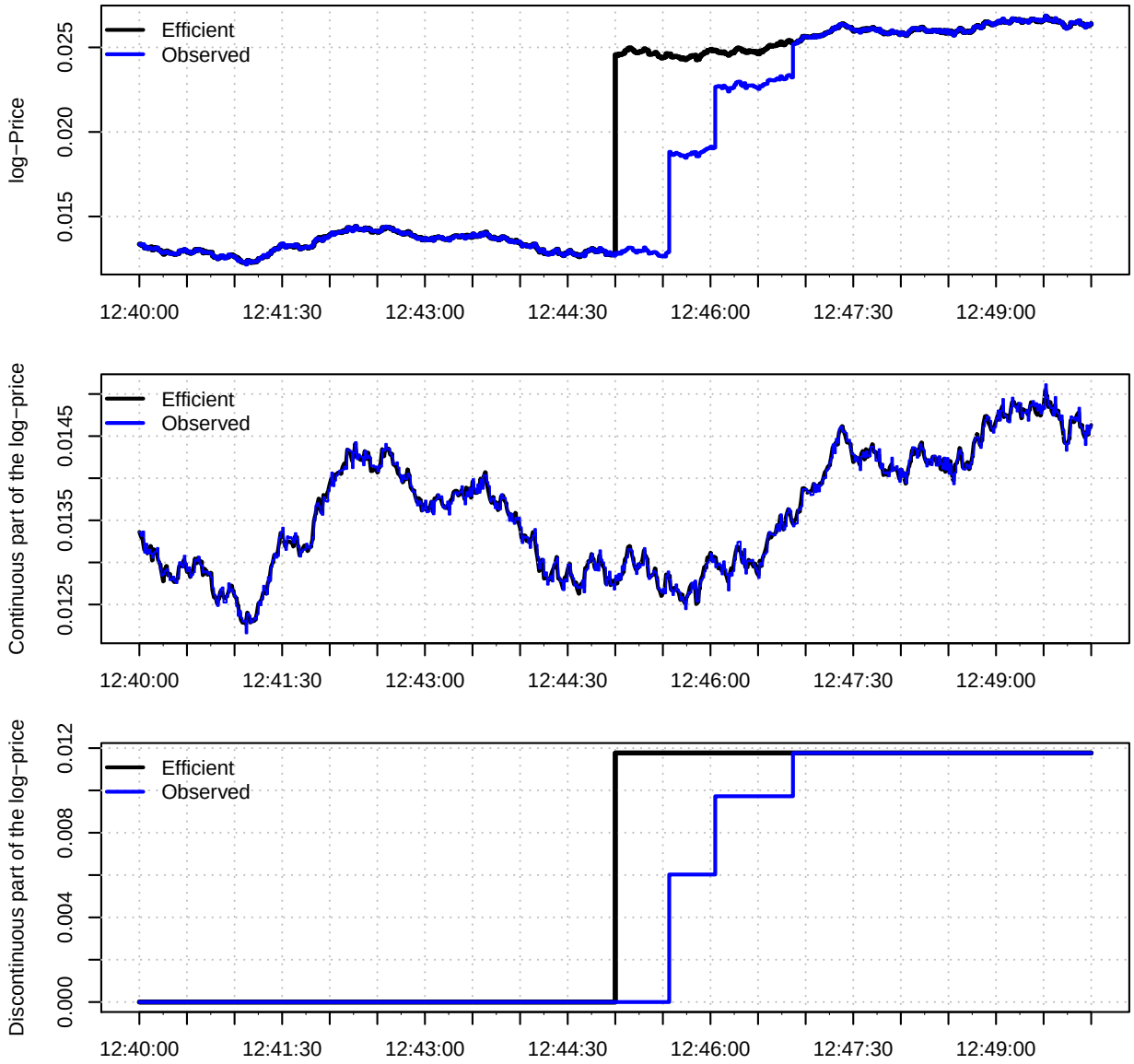
2.2.1 Price and return spreads

An ETF log-price process, Z_t, tracks an index of the p stocks. The log-price Z - which is assumed to equal to the observed log-price - is a linear combination of the efficient log-prices of the stocks in (1).[1] We assume that the ETF price process efficiently incorporates its jump component, moving in lockstep with a portfolio of efficiently priced stocks:

$$Z_t = \sum_{k=1}^{p} w_{k,t} X_{k,t} \tag{5}$$

[1]To simplify our notation, we rely on a weighted average of individual log-returns as opposed to simple returns, but this problem is considered as minor in empirical applications (Jondeau, Poon, and Rockinger, 2007, pp.9).

Figure 2: A stock's price reacts sluggishly to news



The synthetic index portfolio, $S_{i\Delta_n}$, tracks the evolution of the implied stock prices in (2):

$$S_{i\Delta_n} = \sum_{k=1}^{p} w_{k,i\Delta_n} Y_{k,i\Delta_n} \tag{6}$$

We should expect the ETF log-price Z to nearly equal the synthetic log-price S in the absence of sluggish prices. Asynchronicity in the impoundment of news causes the price of the synthetic index portfolio to deviate from the price of an ETF tracking the index.

To clarify the workings of our procedures, we consider a stylistic 3-stock universe ($p = 3$) and a corresponding ETF. The stock names A, B and C correspond to the indices $k = 1, 2$ and 3. The ETF is an equally weighted average of the stocks' efficient prices. We lower

the sampling frequency to one minute ($\Delta_n = 1/391$).

Figure 3 shows the efficient and observed log-prices of the assets. The news is incorporated asynchronously into the prices. Stock A jumps gradually and is 2 minutes late, stock B's jump is 1 minute late and stock C does not jump. The implied – inefficient – price of the basket of stocks deviates from the efficient ETF price.

The deviation or spread in prices is the difference between the observed prices on a synthetic index of stocks $S_{i\Delta_n}$ and the prices of an observable ETF trading the index $Z_{i\Delta_n}$:

$$\delta^p_{i\Delta_n} := S_{i\Delta_n} - Z_{i\Delta_n}. \tag{7}$$

The price spread is the vertical distance between the price of the ABC portfolio and the ABC ETF in the last panel of Figure 3. In the first periods, the prices are almost the same, producing a negligible deviation in prices. At 12:45, the ETF price jumps to the new log-price level 0.011 while the prices of the individual stocks do not move much. After a couple of minutes, the prices of the portfolio of individual stocks catch up to the new equilibrium.

Investors only observe the contaminated stock prices and the ETF price process that tracks the index. If the ETF price efficiently tracks the index, however, we can use the spread between the ETF price and a synthetically constructed index to measure the collective misalignment of the observed stock prices with their efficient levels. We think that it is likely that the ETF does efficiently track the index as it is a highly liquid and carefully watched asset.

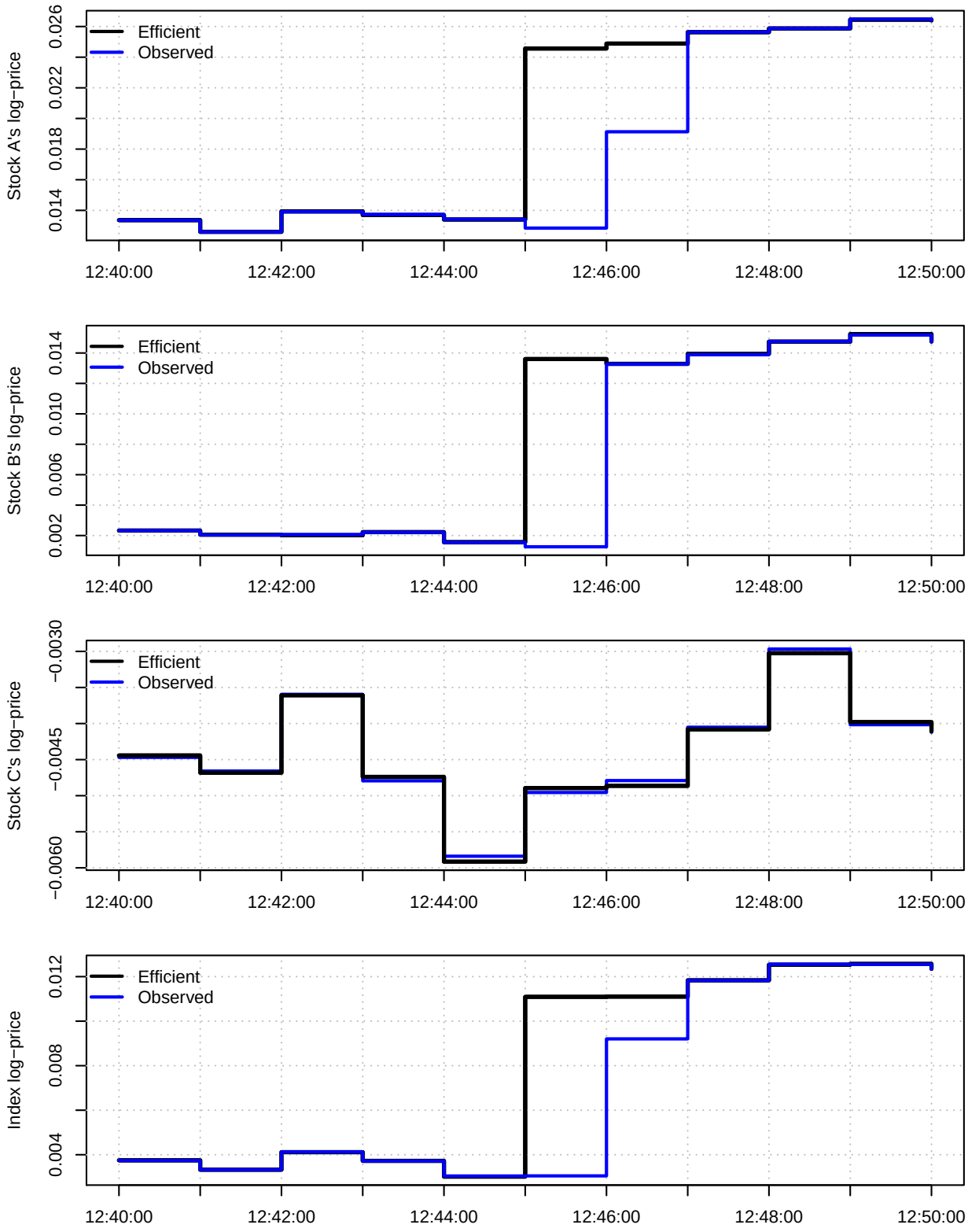
The difference between the price of the synthetic index and the ETF price equals microstructure noise component minus the discontinuous component which is not yet included in the observed price:

$$S_{i\Delta_n} - Z_{i\Delta_n} = \sum_{k=1}^{p} w_{k,i\Delta_n} u_{k,i\Delta_n} + \sum_{k=1}^{p} w_{k,i\Delta_n} \left(Y^d_{k,i\Delta_n} - X^d_{k,i\Delta_n} \right)$$

The sluggish components of jumps are much larger than microstructure noise, so the asynchronicity in the impoundment of news dominates the large price spreads.

In practice, we look at the stock returns to focus on the new information impounded in the stock price. The failure of investors to absorb jump-relevant information retards the price changes $\Delta^n_i Y := Y_{i\Delta_n} - Y_{(i-1)\Delta_n}$. In a panel of stock returns, these large returns appear to be asynchronous across stocks.

Figure 3: Stock prices vary in how quickly they impound new information



Consider the asynchronicity in the following time series of return vectors of the stocks and the ABC ETF:

$$\Delta_i^n Y_1 = \begin{bmatrix} \vdots \\ -0.018 \\ -0.031 \\ -0.057 \\ \underline{0.629} \\ \underline{0.651} \\ \vdots \end{bmatrix}, \Delta_i^n Y_2 = \begin{bmatrix} \vdots \\ 0.015 \\ -0.067 \\ -0.029 \\ \underline{1.201} \\ 0.062 \\ \vdots \end{bmatrix}, \Delta_i^n Y_3 = \begin{bmatrix} \vdots \\ -0.120 \\ -0.104 \\ 0.088 \\ 0.017 \\ 0.074 \\ \vdots \end{bmatrix}, \text{ and } \Delta_i^n Z = \begin{bmatrix} \vdots \\ -0.039 \\ -0.071 \\ \underline{0.807} \\ 0.001 \\ 0.073 \\ \vdots \end{bmatrix}, \quad (8)$$

in which the returns are reported in percentages and the large jump returns are underlined.

The spread in returns is the difference between the observed returns on a synthetic index of stocks $\Delta_i^n S := \sum_{k=1}^p w_{k,i\Delta_n} \Delta_i^n Y_k$ and the returns of an observable ETF trading the index $\Delta_i^n Z := Z_{i\Delta_n} - Z_{(i-1)\Delta_n}$ or, equivalently, the percentage change of the price spread in (7):

$$\delta^r_{i\Delta_n} := \delta^p_{i\Delta_n} - \delta^p_{(i-1)\Delta_n} = \Delta_i^n S - \Delta_i^n Z. \quad (9)$$

The sum of the first three columns of the matrix below, $\delta^r_{i\Delta_n}$, shows the equally weighted, linear combination of the log-returns of stocks A, B, and C (the first three columns in (8)). The linear combination is the return on the synthetic ABC portfolio. The fourth column on the left side of the equal sign is the log-return of the ABC ETF (the last column in (8)). Their difference, on the right hand side of the equation, is the return spread (9) in each of the five periods.

$$\delta^r_{i\Delta_n} = \begin{bmatrix} \vdots \\ [1/3.(-0.018 + 0.015 - 0.120)] - (-0.039) \\ [1/3.(-0.031 - 0.067 - 0.104)] - (-0.071) \\ [1/3.(-0.057 - 0.029 + 0.088)] - (\ \underline{0.807}) \\ [1/3.(\ \underline{0.629} + \underline{1.201} + 0.017)] - (\ 0.001) \\ [1/3.(\ \underline{0.651} + 0.062 + 0.074)] - (\ 0.073) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \\ \vdots \end{bmatrix}.$$

In the first two periods, the returns to the synthetic portfolio are almost the same as the returns to the ETF, producing only a small deviation. In the third period, the ETF price jumps by 0.807%, while the prices of the individual stocks do not move much. In the fourth and fifth periods, the prices of the portfolio of individual stocks catch up to the ETF jump, but the returns of the synthetic ABC portfolio are not offset by the returns of the ABC ETF.

2.2.2 Jump vectors and the jump-event window

News and economic events generate price jumps, *i.e.* very large returns. Jump tests, *e.g.*, Lee and Mykland (2008) and Mancini (2001), classify observed returns (not the efficient returns) as either discontinuous or continuous:

$$\Delta_i^n Y_k = \Delta_i^n J_k + \Delta_i^n C_k \text{ and } \Delta_i^n Z \ = \Delta_i^n \widetilde{J} \ + \Delta_i^n \widetilde{C} \ , \tag{10}$$

in which $\Delta_i^n Y_k$, for $k = 1, ..., p$, the ith return of the one-dimensional observed stock log-price process, Y_k, is the sum of a discontinuous stock return $\Delta_i^n J_k := \Delta_i^n Y_k \cdot I(\text{Jump}_{i\Delta_n})$ and a continuous stock return $\Delta_i^n C_k := \Delta_i^n Y_k \cdot I(\text{No Jump}_{i\Delta_n})$, in which $I(\cdot)$ is an indicator function. A similar classification applies to the ETF return $\Delta_i^n Z$.

Back to our 3-stock stock universe. The ABC ETF returns equal the sum of the continuous and discontinuous (jump) returns:

$$\Delta_i^n Z = \begin{bmatrix} \vdots \\ -0.039 \\ -0.071 \\ \underline{0.807} \\ 0.001 \\ 0.073 \\ \vdots \end{bmatrix} \text{ is the sum of } \Delta_i^n \widetilde{J} = \begin{bmatrix} \vdots \\ 0.000 \\ 0.000 \\ \underline{0.807} \\ 0.000 \\ 0.000 \\ \vdots \end{bmatrix} \text{ and } \Delta_i^n \widetilde{C} = \begin{bmatrix} \vdots \\ -0.039 \\ -0.071 \\ 0.000 \\ 0.001 \\ 0.073 \\ \vdots \end{bmatrix}.$$

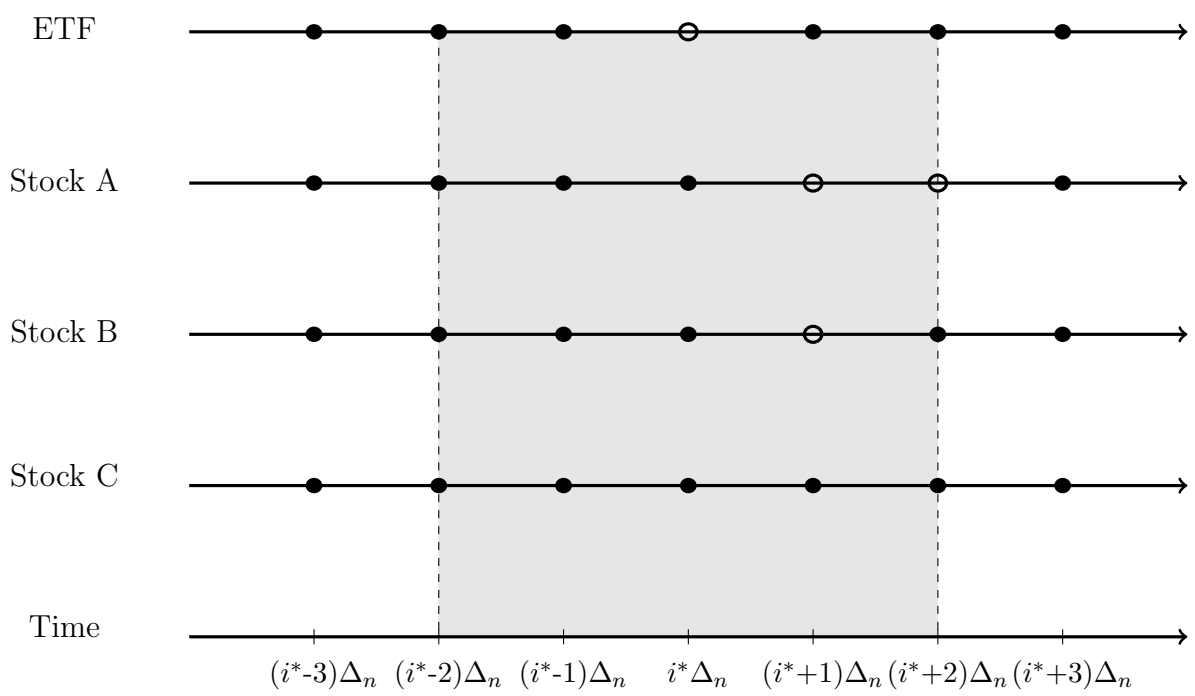
The continuous and jump return vectors are mutually exclusive. The jump return vector is sparse.

We assume that the ETF immediately impounds latent common jump in stock prices. For a given ETF index jump at time $i^*\Delta_n$, we look at the jump vectors of the individual stocks within a window of observations:

$$[\Delta_i^n J_k]_{i \in \mathcal{W}_n}, \text{ with } k = 1, ..., p, \tag{11}$$

in which $\Delta_i^n J_k$ is the vector of jump returns for stock k, with $k = 1, ..., p$, and $\mathcal{W}_n := [I_1\Delta_n, I_2\Delta_n]$ is an event window of size $h \equiv I_2\Delta_n - I_1\Delta_n + 1$. To clarify the mechanics behind our procedures, we use a stylized example with an event window from two minutes before to two minutes after the index jump, $I_1\Delta_n = (i^* - 2)\Delta_n$ and $I_2\Delta_n = (i^* + 2)\Delta_n$. For the empirical application in Section 3, we broaden the event window from ten minutes before to ten minutes after the index jump.

Figure 4: Jumps are observed at close but distinct points in time



The following vectors are the stock jump vectors in a window two minutes before and after the index jump:

$$[\Delta_i^n J_1]_{i\in\mathcal{W}_n} = \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ \underline{0.629} \\ \underline{0.651} \end{bmatrix}, [\Delta_i^n J_2]_{i\in\mathcal{W}_n} = \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ \underline{1.201} \\ 0.000 \end{bmatrix} \text{ and } [\Delta_i^n J_3]_{i\in\mathcal{W}_n} = \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

Stock A jumps gradually - in two increments in periods 4 and 5 - and is two minutes late, Stock B jumps one minute late and Stock C does not jump in the event window.

The five-minute event window is shaded in Figure 4. We depict jumps as open dots and non-jumps as closed dots. This stylized example captures the central problem in the analysis of common jumps. If news reached the entire market instantly and trading were continuous, jumps in a group of stocks should presumably occur simultaneously with the ETF index jump. Sluggish price changes lead to delays in observed jumps, however. The jumps are observed at close but distinct points in time. We call this phenomenon a "sluggish cojump". We synchronize these stock jumps to recover the efficient common jump.

2.2.3 Decomposition of the spread

Large spreads between the ETF and a synthetic stock portfolio occur when information is impounded into stocks with (variable) lags. To focus on the effect of asynchronicity in the impoundment of news on the spread, we break up the return of synthetic stock portfolio $\Delta_i^n S$ into its discontinuous and continuous part:

$$\Delta_i^n S := \sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n Y_k = \sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n J_k + \sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n C_k. \tag{12}$$

For example, the equally weighted ABC portfolio return $\Delta_i^n S$ is the sum of weighted discontinuous stock returns and weighted continuous stock returns:

$$\sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n J_k = \begin{bmatrix} \vdots \\ 1/3.(\quad 0.000 + 0.000 + 0.000) \\ 1/3.(\quad 0.000 + 0.000 + 0.000) \\ 1/3.(\quad 0.000 + 0.000 + 0.000) \\ 1/3.(\quad \underline{0.629} + \underline{1.201} + 0.000) \\ 1/3.(\quad \underline{0.651} + 0.000 + 0.000) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.610 \\ 0.217 \\ \vdots \end{bmatrix}$$

$$\sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n C_k = \begin{bmatrix} \vdots \\ 1/3.(-0.018 + 0.015 - 0.120) \\ 1/3.(-0.031 - 0.067 - 0.104) \\ 1/3.(-0.057 - 0.029 + 0.088) \\ 1/3.(\quad 0.000 + 0.000 + 0.017) \\ 1/3.(\quad 0.000 + 0.062 + 0.074) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ -0.041 \\ -0.067 \\ 0.001 \\ 0.006 \\ 0.045 \\ \vdots \end{bmatrix}.$$

The return spread (9) now equals a linear combination of the weighted stock jumps, the weighted continuous stock returns and the ETF returns:

$$\delta_{i\Delta_n}^r \overset{(9)}{:=} \Delta_i^n S - \Delta_i^n Z \overset{(12)}{=} \sum_{k=1}^{p} \underbrace{w_{k,i\Delta_n} \Delta_i^n J_k}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} + \underbrace{\sum_{k=1}^{p} \underbrace{w_{k,i\Delta_n} \Delta_i^n C_k}_{\substack{\text{Weighted} \\ \text{continuous} \\ \text{stock returns}}} - \underbrace{\Delta_i^n Z}_{\text{ETF returns}}}_{\text{Target}}. \tag{13}$$

We want a target to hit when we synchronize the individual stock jumps. The target will be the difference between the weighted continuous stock returns and the ETF returns. If both the stock jumps and the ETF impound information at the same time – the stock jumps

offset the target – the spread in returns should be small, containing only microstructure noise (see 2.2.1). Sluggish stock price jumps cause large return spreads. By rearranging the stock jumps, we can minimize the variance of the return spreads, *i.e.* the right side of (13).

2.2.4 Constructing the jump-event matrix

To synchronize the individual stock jumps, we translate the decomposition of return spreads (13) into a easier-to-handle object: a $h \times q$ jump-event matrix J_n, where h is the number of periods in the window around the jump and $q-1$ is the number of stock jumps. This object highlights how stock jumps influence the return spreads:

$$J_n = (\gamma_{il}) := \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n}, \tag{14}$$

in which $i = 1, ..., h$ and $l = 1, ..., q$. The first $q-1$ columns consist of the vectors of weighted stock discontinuous returns $w_{i\Delta_n} \Delta_i^n J := (w_{1,i\Delta_n} \Delta_i^n J_1, ..., w_{q-1,i\Delta_n} \Delta_i^n J_{q-1})$ sampled within a window of h observations around the ETF jump. The weighted jump vectors $w_{i\Delta_n} \Delta_i^n J$ consist of the p stock jump vectors $\Delta_i^n J_k$, for $k = 1, ..., p$ (11), but we reorganize the stock jump vectors. If a stock jumps gradually, like in stock A, separate jumps appear in different columns. We only allow for one non-zero element in the jump vectors. We exclude the jump vectors for which the stock does not jump, like in stock C. The qth column is the target vector, $T_{i\Delta_n} := (\sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n C_k) - \Delta_i^n Z$, which is the difference between the continuous returns of the synthetic stock index portfolio, $\sum_{k=1}^{p} w_{k,i\Delta_n} \Delta_i^n C_k$, and the ETF return $\Delta_i^n Z$. Note that we structure the jump-event matrix such that its row-sums are equal to the return spreads (13) across the event window:

$$J_n^+ := J_n \times 1_q = [\delta_{i\Delta_n}^r]_{i \in \mathcal{W}_n} \tag{15}$$

in which 1_q is a column-vector of ones.

The jump-event matrix (14) in our example looks like:

$$J_n = \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{\phantom{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}}_{\text{Target}}, \text{with row-sums } J_n^+ = \begin{bmatrix} -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \end{bmatrix}.$$

The first three columns of the jump-event matrix correspond to the jump vectors of the three individual stock jumps across the event window. The first two columns contain the gradual jumps of Stock A and the third column contains the delayed jump of stock B. The stock jump sizes are generally weighted according to their shares of the index, but this example uses an equiweighted index for simplicity. The fourth column of the jump-event matrix is a target vector that contains the difference between the continuous returns of the stocks and the ETF returns. The return spread is the sum of the row-sums of the weighted discontinuous stock returns and the target column, *i.e.* the row-sums of the jump-event matrix (15). As before (in 2.2.1), there are large spikes in the return spread in a small window around the ETF jump. Stocks A and B do not jump until periods 4 and 5.

2.3 Rearranging the elements within the jump-event matrix

If prices respond sluggishly to news, a common jump in the efficient price for a substantial share of the market manifests in asynchronous jumps. After decomposing the stock returns into jump and non-jump moves, the jumps can be rearranged in time to offset the target column, thereby minimizing the variance in the return spreads and recovering the covariance of the efficient cojump.

2.3.1 Permutations

The minimization of the variance of the row-sums J_n^+ vector (15) by rearranging observed stock jumps in a jump-event matrix (14) is a combinatorial problem. We rearrange the h elements in each of weighted stock jump vectors (the first $q-1$ columns in the jump-event matrix J_n) to offset the elements in the target vector (the last column in the jump-event matrix J_n), so that we minimize the variance of the row-sums of the jump-event matrix. Each rearrangement is defined by a particular permutation π_l of h elements of the lth

column: $\pi_l : \{1, ..., h\} \to \{1, ..., h\}$, with $l = 1, ..., q$. The permutation π_l is represented compactly by:

$$\begin{pmatrix} \pi_l(1) & \pi_l(2) & ... & \pi_l(h) \end{pmatrix}$$

For example, consider the following permutation π_1 for the first jump of stock A, $[w_{1,i\Delta_n} \Delta_i^n J_1]_{i \in \mathcal{W}_n}$. This permutation swaps the third and the fourth observations in the jump vector:

$$\pi_1 = \begin{pmatrix} 1 & 2 & \underline{4} & \underline{3} & 5 \end{pmatrix}. \tag{16}$$

This rearrangement of the weighted jump vectors for stock A's jump can be expressed in the following way:

$$[w_{1,i\Delta_n} \Delta_i^n J_1]_{i \in \mathcal{W}_n} = \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ \underline{0.210} \\ 0.000 \end{bmatrix} \xrightarrow{\text{Rearrangement}} [w_{1,i\Delta_n} \Delta_i^n J_1^\pi]_{i \in \mathcal{W}_n} = \begin{bmatrix} 0.000 \\ 0.000 \\ \underline{0.210} \\ 0.000 \\ 0.000 \end{bmatrix},$$

which shifts the stock jump back in time within the event window $\mathcal{W}_n$.

The rearrangements of all columns are summarized in the vector of q permutations $\pi := (\pi_1, ..., \pi_q)$. Because we want a target to hit when we synchronize the individual stock jumps, we do not change the rearrangement in the qth column; the target vector.

2.3.2 The return spread after rearrangement

Every rearrangement yields a new jump-event matrix:

$$J_n := \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n} \xrightarrow{\text{Rearrangement}} \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J^\pi}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n} := J_n^\pi \tag{17}$$

in which J_n is the observed jump-event matrix (14), $w_{i\Delta_n} \Delta_i^n J^\pi := (w_{1,i\Delta_n} \Delta_i^n J_1^\pi, ..., w_{q-1,i\Delta_n} \Delta_i^n J_{q-1}^\pi)$ is the vector of *rearranged* weighted stock jump returns.

The return spreads, the vector of row-sums of the rearranged jump-event matrix J_n^π, are now a function of the arrangement of the stock jumps and thus the stock jump arrival times:

$$J_n^{\pi,+} := J_n^\pi \times 1_q = [\delta_{i\Delta_n}^{r,\pi}]_{i \in \mathcal{W}_n} \tag{18}$$

The permutation π_1 (16) rearranges the jump-event matrix, switching the 3rd and 4th rows of the first column:

$$J_n = \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} \xrightarrow{\text{Rearrangement}} \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ \underline{0.210} & 0.000 & 0.000 & -0.807 \\ 0.000 & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} = J_n^{\pi}.$$

The permutation π_1 also changes the third and fourth row-sum of the jump-event matrix:

$$J_n^{+} = \begin{bmatrix} -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \end{bmatrix} \xrightarrow{\text{Rearrangement}} \begin{bmatrix} -0.002 \\ 0.003 \\ -0.597 \\ 0.404 \\ 0.189 \end{bmatrix} = J_n^{\pi,+}.$$

The variability in the return spreads is smaller after this rearrangement.

2.3.3 The best rearrangement

The best rearrangement rearranges jumps to minimize the variability of the return spreads to reflect that the latent prices of the price series of the stocks and the ETF move in lockstep. This reduction in variability is known as "flattening".

Flattening the return spreads, *i.e.* the row-sums of the jump-event matrix, means that the stochastic variables in the separate columns of the jump-event matrix should be completely mixable. Mathematically, a q-dimensional distribution function $F(Q_1, ..., Q_q)$ on $\mathbb{R}$ is q-completely mixable if there exist q random variables $Q_1, ..., Q_q$ identically distributed as F such that:

$$\text{Prob}\left(Q_1 + Q_2 + ... + Q_q = \text{constant}\right) = 1$$

The sums of random variables should approximate a constant. Complete mixability is a concept of negative dependence. If we only have $q = 2$ variables, complete mixability implies countermonotonicity and perfect negative dependence. A completely mixable dependence structure minimizes the variance of the sum of the random variables with given

marginal distributions. The jump-event matrix consists of realizations of random variables (the stock jump returns) and a difference of random variables (the target). We want the sums of the stock jumps to hit the target. That is, we swap values in each column associated to the stock jumps so that the row-sums in the jump-event matrix have minimal variability.

The combinatorial optimization problem is as follows. We rearrange the elements in the jump-event matrix to minimize the variability of the row-sums of the rearranged matrix:

$$\min_{\pi} V\left(J_n^{\pi,+}\right), \text{ with } J_n^{\pi,+} := \sum_{m=1}^{q+1} (\gamma_{im}^{\pi}), \tag{19}$$

in which the row-sums $J_n^{\pi,+}$ are return spreads expressed as a function of the arrangement of stock jumps, $V(\cdot)$ is a scalar-valued function that measures the variability of the vector of row-sums. Possible candidates for the variability measure are the variance and the range. The rearrangement switches the stock jumps to offset the target column. The result of this optimization synchronizes scattered jumps so that they minimize the difference between the price of the synthetic index portfolio and that of the ETF.

The rearrangement problem (19) is rooted in the pioneering work of Puccetti and Rüschendorf (2012) and Embrechts et al. (2013) on rearrangements and their rearrangement algorithm. This algorithm is best known as an actuarial tool to compute bounds on portfolio risk measures, but it also has applications in other disciplines, such as operations research (see *e.g.*, Boudt et al., 2018).

Rearrangements can also effectively synchronize stock jumps and recover the efficient common jump. Optimally rearranging the jumps in the previous example, by minimizing the variability of the row-sums, produces the following transformation:

$$J_n = \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} \xrightarrow[\text{rearrangement}]{\text{The best}} \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ \underline{0.210} & \underline{0.217} & \underline{0.400} & -0.807 \\ 0.000 & 0.000 & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} = J_n^{\pi}.$$

Rearrangement combines two small jumps of stock A and shifts the jumps of stock B one period backward, aligning the jumps in time.

The best rearrangement tempers the variability in the return spreads:

$$J_n^+ = \begin{bmatrix} -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \end{bmatrix} \xrightarrow[\text{rearrangement}]{\text{The best}} \begin{bmatrix} -0.002 \\ 0.003 \\ 0.020 \\ 0.004 \\ -0.028 \end{bmatrix} = J_n^{\pi,+}$$

Figure 5 shows the implied prices after this new arrangement of jump returns. The news gets incorporated asynchronously into the observed stock prices. That is, the observed (blue) prices of stock A and stock B deviate from their efficient (black) values in the first and second panels. As a result, the observed price of the ETF deviates from its efficient value in the fourth panel. An optimal rearrangement of the stock jumps recovers the jump in the efficient price path. The rearranged price paths (green) are now much closer to the efficient price paths (black) in the 1st, 2nd and 4th panel.

2.4 Rearrangement Linear Program

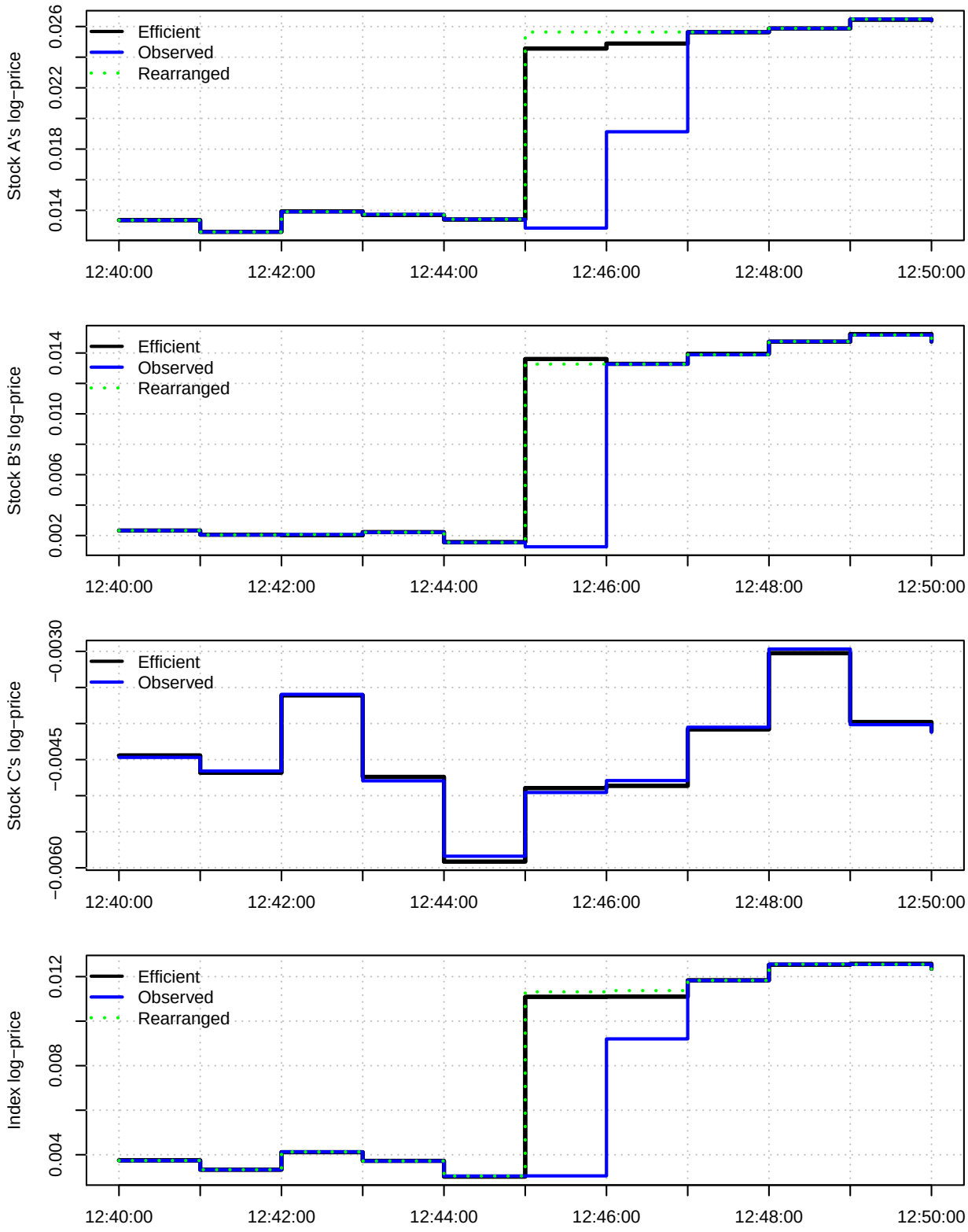
We formulate a linear program to solve the combinatorial problem in (19) to achieve the best rearrangement. Our program imposes penalties to prevent rearrangements that are economically implausible. For example, we penalize large moves or the penalty might be asymmetric, with a greater penalty for moving them later in time than earlier, or it could penalize moves that take stock jumps farther away from the ETF jump. The approach is similar in spirit to the Lasso (Tibshirani, 1996) in that it penalizes complexity in the form of moving jumps.

2.4.1 The vanilla rearrangement algorithm

Several heuristics exist to flatten the row-sums of a matrix. Our rearrangement linear program extends the rearrangement algorithm of Puccetti and Rüschendorf (2012) and Embrechts et al. (2013). The algorithm rearranges the elements in each column of the matrix, so that each column becomes oppositely ordered to the sum of the other columns. It loops over each column one-by-one. Our benchmark is the standard rearrangement algorithm with a fixed target in the last column (see *e.g.*, Bernard et al., 2018, 2017).

Suppose that we want to rearrange the jump-event matrix of size $h \times q$. The algorithm is as follows:

Figure 5: Rearrangement recovers the efficient stock jumps



1. Randomly shuffle the elements (in each of the first $q-1$ columns) to obtain the starting matrix of the algorithm. The random shuffle already flattens (*i.e.*, reduce the variability of) the row-sums.

$$J_n = \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} \xrightarrow[\text{Shuffle}]{\text{Random}} \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ \underline{0.210} & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & \underline{0.400} & -0.807 \\ 0.000 & \underline{0.217} & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}}$$

2. Iteratively rearrange the l-th column of the rearranged matrix J_n^π so that it becomes oppositely ordered to the sum of the other columns, for $l = 1, ..., q-1$. We never rearrange the target column, $l = q$. For $l = 1$, the algorithm immediately matches the stock jump with the ETF jump:

$$\underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ \underline{0.210} & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & \underline{0.400} & -0.807 \\ 0.000 & \underline{0.217} & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} \xrightarrow[\text{order}]{\text{Oppositely}} \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ \underline{0.210} & 0.000 & \underline{0.400} & -0.807 \\ 0.000 & \underline{0.217} & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}}$$

3. Repeat Step 2 until no further changes occur. That is, until a matrix J_n^π is found with each column oppositely ordered to the sum of the other columns. The matrix will have row-wise sums with minimal variance. All jumps align in the third row, at the time of the ETF jump, and the variance of the row-sums drops from 0.266 to 0.000:

$$\cdots \xrightarrow[\text{order}]{\text{Oppositely}} \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ \underline{0.210} & \underline{0.217} & \underline{0.400} & -0.807 \\ 0.000 & 0.000 & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \underbrace{}_{\text{Target}} = J_n^\pi$$

The rearrangement algorithm matches the synthetic index portfolio's jumps to the ETF jumps. But there are no constraints on the type of rearrangements that take place. The algorithm can move any element anywhere as long as it improves the objective. For example, it can move jumps either forward or backward in time. To constrain the procedure from economically implausible rearrangements, we next extend the rearrangement algorithm into a linear program that penalizes such undesirable actions.

2.4.2 Permutations as a decision variable

Permutations summarize the positions of the elements in the rearranged jump-event matrix (2.3.1). The rearrangement algorithm (2.4.1) permutes the elements in each column (with the exception of the target column) in each iteration, but it is not possible directly constrain where the algorithm can move the elements. It directly changes the elements in the jump-event matrix, by making sure that each column is oppositely ordered to the sum of the other columns. Instead of directly changing the jump-event matrix, we treat the permutations as a decision variable and constrain such moves.

We impose constraints on the rearrangements by using the column representation of permutation matrix. The $h \times h$ permutation matrix for one permutation, π_l, is a permutation of the columns of the identity matrix I_h:

$$P_{\pi_l} = (p_{ii'}) = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1h} \\ p_{21} & p_{22} & \dots & p_{2h} \\ \vdots & & \ddots & \\ p_{h1} & p_{h2} & \dots & p_{hh} \end{bmatrix} \tag{20}$$

For each i, $p_{ii'}$ is 1 if $i' = \pi_l(i)$ and is 0 otherwise. The entries of the ith row are all zero except for a 1 that appears in column $\pi_l(i)$:

$$P_{\pi_l} = \begin{bmatrix} \boldsymbol{e}_{\pi_l(1)} \\ \boldsymbol{e}_{\pi_l(2)} \\ \vdots \\ \boldsymbol{e}_{\pi_l(n)} \end{bmatrix}$$

in which $\boldsymbol{e}_{i'}$, a standard basis vector, denotes a row-vector of length h with a 1 on position i' and a 0 on every other position.

The permutation in our example, $\pi_1 = \begin{pmatrix} 1 & 2 & \underline{4} & \underline{3} & 5 \end{pmatrix}$, yields the following permuta-

tion matrix:

$$P_{\pi_1} = \begin{bmatrix} \boldsymbol{e}_{\pi_1(1)} \\ \boldsymbol{e}_{\pi_1(2)} \\ \vdots \\ \boldsymbol{e}_{\pi_1(n)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_4 \\ \boldsymbol{e}_3 \\ \boldsymbol{e}_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{1} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

in which column i' of the I_5 identity matrix now appears as the column $\pi(i')$ of P_{π_1}.

We can track the new positions of each element in the columns of the permutation matrix (20) by tracking how far they deviate from the diagonal. The fourth element, which is on position $(4,4)$ in I_5, shifts one spot backwards by shifting one step upward in the permutation matrix. The third element, which is on position $(3,3)$ in I_5, shifts one step forward by shifting one step downward in the permutation matrix. We can directly impose penalties on where we rearrange the elements.

Our rearrangement problem (19) rearranges multiple columns. We have a permutation matrix for each of the q columns in the jump-event matrix. We concatenate the permutation matrices together in the $h \times (hq)$ co-permutation matrix:

$$\Pi = (p_{lii'}) = \left[P_{\pi_1}, P_{\pi_2}, ..., P_{\pi_q}\right] \tag{21}$$

with $l = 1, ..., q$ and $i, i' = 1, ..., h$.

To find the best rearrangement of the jump-event matrix, $J_n = (\gamma_{il})$, with $i = 1, ..., h$ and $l = 1, ..., q$, where h is the number of periods in the window around the ETF jump and $q - 1$ is the number of stocks, we can rewrite the return spreads of the as a function of these permutation matrices:

$$J_n^{\pi,+} = \Pi \times \text{vec}(J_n) \tag{22}$$

or, equivalently:

$$\begin{bmatrix} J_{n,1}^{\pi,+} \\ J_{n,2}^{\pi,+} \\ J_{n,3}^{\pi,+} \\ \vdots \\ J_{n,h}^{\pi,+} \end{bmatrix} = \left[\begin{array}{cccc|cccc|c|cccc} p_{111} & p_{112} & \cdots & p_{11h} & p_{211} & p_{212} & \cdots & p_{21h} & \cdots & p_{q11} & p_{q12} & \cdots & p_{q1h} \\ p_{121} & p_{122} & \cdots & p_{12h} & p_{221} & p_{222} & \cdots & p_{22h} & \cdots & p_{q21} & p_{q22} & \cdots & p_{q2h} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & & \vdots & & \ddots & \vdots \\ p_{1h1} & p_{1h2} & \cdots & p_{1hh} & p_{2h1} & p_{2h2} & \cdots & p_{2hh} & \cdots & p_{qh1} & p_{qh2} & \cdots & p_{qhh} \end{array}\right] \times \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \\ \vdots \\ \underline{\gamma_{h1}} \\ \vdots \\ \gamma_{hq} \end{bmatrix}$$

in which $\Pi = \left[P_{\pi_1}, P_{\pi_2}, ..., P_{\pi_q}\right]$, with P_{π_1} short notation for the $h \times h$ permutation matrix for the first column of the jump event matrix. The vectorized version of the observed

jump-event matrix $J_n = (\gamma_{il})$, with $i = 1, ..., h$ and $l = 1, ..., q$, is a stacked column vector of dimension $hq \times 1$.

The linear program rearranges elements by changing the ones in co-permutation matrix instead of directly changing the jump-event matrix. The result of the matrix product is a $h \times 1$ column vector, including the row-sums of the rearranged jump-event matrix.

Back to our example. The vectorized jump-event matrix stacks the columns on top of one another. To render the observed jump matrix, without any rearrangements, the permutations P_{π_l} should be the identity matrices I_h, for all columns $l = 1, ..., q$:

$$\Pi = \left[\begin{array}{ccccc|ccccc|ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]; \text{vec}(J_n) = \left[\begin{array}{r} 0.000 \\ 0.000 \\ 0.000 \\ 0.210 \\ 0.000 \\ \hline 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.217 \\ \hline 0.000 \\ 0.000 \\ 0.000 \\ 0.400 \\ 0.000 \\ \hline -0.002 \\ 0.003 \\ -0.807 \\ 0.004 \\ -0.028 \end{array}\right].$$

The row-sums are equal to:

$$J_n^{\pi,+} = \Pi \times \text{vec}(J_n) = \begin{bmatrix} 0.000 + 0.000 + 0.000 - 0.002 \\ 0.000 + 0.000 + 0.000 + 0.003 \\ 0.000 + 0.000 + 0.000 - 0.807 \\ 0.210 + 0.000 + 0.400 + 0.004 \\ 0.000 + 0.217 + 0.000 - 0.028 \end{bmatrix} = \begin{bmatrix} -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \end{bmatrix}.$$

Note that the row-sums are the same as the observed return spreads in 2.2.4, which were obtained by calculating the row-sums of the jump-event matrix.

It is straightforward to see that the rearranged matrix is the observed jump-event matrix:

$$J_n^{\pi} = \left[\begin{array}{cccc} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{array}\right], \quad \begin{array}{c} \\ \\ \\ \\ \underbrace{} \\ \text{Rearranged weighted discontinuous stock returns} \end{array} \quad \begin{array}{c} \\ \\ \\ \\ \underbrace{} \\ \text{Target} \end{array}$$

2.4.3 Objective function

Return spreads are a function of the co-permutation matrix (22). The goal of rearrangement is to flatten the row-sums $J_{n,i}^{\pi,+}$, for $i = 1, ..., h$. The algorithm minimizes the variance of the row-sums (see 2.4.1). To reframe the problem into a linear program, we minimize the range of the row-sums.

We need the order statistics to calculate the range. For a particular co-permutation matrix Π, we have the corresponding row-sums, $J_{n,1}^{\pi,+}, J_{n,2}^{\pi,+}, ..., J_{n,h}^{\pi,+}$. We denote the order statistics as $J_{n,(1)}^{\pi,+}, J_{n,(2)}^{\pi,+}, ..., J_{n,(h)}^{\pi,+}$, in which the subscript (i) enclosed in parentheses indicates the ith order statistic of the sample.

The smallest order statistic is the minimum of the sample and the largest order statistic is the maximum of the sample:

$$J_{n,(1)}^{\pi,+} = \min\{J_{n,1}^{\pi,+}, J_{n,2}^{\pi,+}, ..., J_{n,h}^{\pi,+}\} \text{ and } J_{n,(h)}^{\pi,+} = \max\{J_{n,1}^{\pi,+}, J_{n,2}^{\pi,+}, ..., J_{n,h}^{\pi,+}\} \tag{23}$$

The sample range is the difference between the maximum and the minimum order statistic:

$$R(J_{n,1}^{\pi,+}, J_{n,2}^{\pi,+}, ..., J_{n,h}^{\pi,+}) = J_{n,(h)}^{\pi,+} - J_{n,(1)}^{\pi,+} \tag{24}$$

The best rearrangement minimizes the variability – the range – of the row-sums:

$$\underset{\Pi, J_{n,(1)}^{\pi,+}, J_{n,(h)}^{\pi,+}}{\text{Minimize}} \; R(J_{n,1}^{\pi,+}, J_{n,2}^{\pi,+}, ..., J_{n,h}^{\pi,+}) \tag{Objective}$$

in which Π is the co-permutation matrix, which defines arrangement of the elements in the rearranged jump-event matrix. The row-sums $J_{n,i}^{\pi,+}$ are a linear function of the co-permutation matrix and the jump-event matrix matrix (22) and $J_{n,(1)}^{\pi,+}$ and $J_{n,(h)}^{\pi,+}$ are latent

smallest and largest order statistics (23). The linear program minimizes the range by choosing the co-permutation matrix that minimizes the difference between the maximum and minimum feasible row-sum. For a perfect rearrangement, the smallest and largest row-sum are equal after convergence, $J^{\pi,+}_{n,(1)} = J^{\pi,+}_{n,(h)}$, which flattens the row-sums.

2.4.4 Rearrangements in a linear program

To fully replicate the machinery of the rearrangement algorithm, we introduce constraints on the decision variables by constraining the feasible choices of the co-permutation matrix. We acknowledge that the minimum and maximum row-sums are latent and connect them to the co-permutation matrix.

A. The ordering constraint

While minimizing the range, the linear program looks for a feasible arrangement in the co-permutation matrix Π. Each arrangement has a corresponding set of row-sums $J^{\pi,+}_{n,i}$. To minimize the range we span latent bounds on these row-sums:

$$J^{\pi,+}_{n,(1)} \leq J^{\pi,+}_{n,i} \leq J^{\pi,+}_{n,(h)}, \text{ for } i = 1, ..., h. \tag{C.1}$$

We look for the rearrangement with the smallest range. The constraint also makes sure that we do not swap the minimum and the maximum row-sum.

B. The permutation constraint

We define a permutation in the linear program. Three types of constraints on the co-permutations Π suffice: a selection constraint, a row constraint and a column constraint.

The selection constraint. Each permutation matrix (20) has h ones to make sure that there we select exactly h elements in each column of the jump-event matrix:

$$\sum_{l=1}^{q} \sum_{i=1}^{h} \sum_{i'=1}^{h} p_{lii'} = hq \tag{C.2a}$$

The row and column constraints. For the row and column constraint, we take a closer look at the permutation matrix for the first column of the jump event matrix:

$$P_{\pi_1} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1h} \\ p_{21} & p_{22} & \cdots & p_{2h} \\ \vdots & & \ddots & \vdots \\ p_{h1} & p_{h2} & \cdots & p_{hh} \end{bmatrix}$$

The row-sums and column-sums of this matrix should equal one. The combination of these constraints makes sure that – the rearrangement of a jump vector (in our case $[w_{1,i\Delta_n}\Delta_i^n J_1]_{i\in\mathcal{W}_n}$) – all numbers are selected only once across all observations i, with $i = 1, ..., h$. We impose the row-constraint on each permutation matrix in the optimization problem:

$$\sum_{i'=1}^{h} p_{lii'} = 1, \text{ for } i = 1, ..., h \text{ and } l = 1, ..., q \tag{C.2b}$$

Suppose for the sake of argument that the row-sums are not equal to one, then we could have either multiple or zero elements in the rearranged matrix on a particular position:

$$\left[\begin{array}{ccccc|c} \underline{1} & \underline{1} & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & \dots \end{array}\right] \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \\ \vdots \\ \underline{\gamma_{h1}} \\ \vdots \end{bmatrix} = \begin{bmatrix} (\gamma_{11} + \gamma_{21}) & + \dots \\ 0 & + \dots \\ \gamma_{31} & + \dots \\ \gamma_{41} & + \dots \\ \gamma_{51} & + \dots \end{bmatrix}$$

This solution matrix is not feasible since the rearrangement is not a permutation. The rearrangement selects the sum of two elements for the position $(1, 1)$, namely $\gamma_{1,1} + \gamma_{21}$ and no element for the position $(2, 1)$. The linear program imposes a row-constraint for each of the separate observations $i = 1, ..., h$ and each column, $l = 1, ..., q$. The linear program also imposes a column-constraint on each permutation matrix in the optimization problem:

$$\sum_{i=1}^{h} p_{lii'} = 1, \text{ for } l = 1, ..., q \text{ and } i' = 1, ..., h \tag{C.2c}$$

Suppose the row-sums are equal to 1, but the column-sums are not equal to one. Then we can have the same number in a column in the rearranged matrix.

$$\left[\begin{array}{ccccc|c} \underline{1} & 0 & 0 & 0 & 0 & \dots \\ \underline{1} & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & \dots \end{array}\right] \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \\ \vdots \\ \underline{\gamma_{h1}} \\ \vdots \end{bmatrix} = \begin{bmatrix} \gamma_{11} & + \dots \\ \gamma_{11} & + \dots \\ \gamma_{31} & + \dots \\ \gamma_{41} & + \dots \\ \gamma_{51} & + \dots \end{bmatrix},$$

which, again, is not a permutation. We select the same element twice, γ_{11} and we place it on two positions, $(1, 1)$ and $(2, 1)$. We did not select γ_{21}. The linear program imposes a column constraint for each column in the permutation matrix, with $i' = 1, ..., h$, and for each column in the jump-event matrix, with $l = 1, ..., q$.

C. The target

The jump-event matrix includes a non-rearrangable target column. By convention, the target vector is in the last column of the jump-event matrix and its permutation matrix is last in the co-permutation matrix. To extract it we set the other permutation matrices to zero:

$$\left[\begin{array}{ccccc|ccccc|c|ccccc} \underline{0} & 0 & 0 & 0 & 0 & \underline{0} & 0 & 0 & 0 & 0 & \dots & \underline{1} & 0 & 0 & 0 & 0 \\ 0 & \underline{0} & 0 & 0 & 0 & 0 & \underline{0} & 0 & 0 & 0 & \dots & 0 & \underline{1} & 0 & 0 & 0 \\ 0 & 0 & \underline{0} & 0 & 0 & 0 & 0 & \underline{0} & 0 & 0 & \dots & 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & \underline{0} & 0 & 0 & 0 & 0 & \underline{0} & 0 & \dots & 0 & 0 & 0 & \underline{1} & 0 \\ 0 & 0 & 0 & 0 & \underline{0} & 0 & 0 & 0 & 0 & \underline{0} & \dots & 0 & 0 & 0 & 0 & \underline{1} \end{array}\right] \times \left[\begin{array}{c} \vdots \\ \hline \gamma_{1q} \\ \gamma_{2q} \\ \vdots \\ \gamma_{hq} \end{array}\right] = \left[\begin{array}{c} 0+0+0+\ldots+\gamma_{1q} \\ 0+0+0+\ldots+\gamma_{2q} \\ \vdots \\ 0+0+0+\ldots+\gamma_{hq} \end{array}\right]$$

The linear program imposes that the permutation matrix corresponding to the qth column in the jump-event matrix, P_{π_q} to remain an identity matrix I_n. The elements cannot deviate from the diagonal:

$$(\text{diag}(P_{\pi_q}))_i = 1, \text{ for } i = 1, ..., h \tag{C.3}$$

D. Example: The rearrangement with the smallest range

The rearrangement linear program solves the combinatorial problem in (19) to achieve the best rearrangement. We use Gurobi Optimizer version 9.1.0 build v9.1.0rc0 (mac64) to optimize the linear program. When the linear program rearranges the jump-event matrix, the stock jumps align in time and the range drops from 1.421 to 0.048. The rearrangement takes about 0.019 seconds.

$$J_n = \underbrace{\left[\begin{array}{ccc|c} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 0.000 & -0.807 \\ \underline{0.210} & 0.000 & \underline{0.400} & 0.004 \\ 0.000 & \underline{0.217} & 0.000 & -0.028 \end{array}\right]}_{\substack{\text{Weighted} \\ \text{discontinuous} \\ \text{stock returns}}} \xrightarrow[\text{rearrangement}]{\text{The best}} \underbrace{\left[\begin{array}{ccc|c} 0.000 & 0.000 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.000 & 0.003 \\ \underline{0.210} & \underline{0.217} & \underline{0.400} & -0.807 \\ 0.000 & 0.000 & 0.000 & 0.004 \\ 0.000 & 0.000 & 0.000 & -0.028 \end{array}\right]}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{discontinuous} \\ \text{stock returns}}} = J_n^{\pi}.$$

2.4.5 Penalties on the rearrangements

The vanilla rearrangement algorithm (2.4.1) and unconstrained rearrangement linear program permutes elements in each column without regard to where the new elements land.

We introduce penalties for rearrangements we consider economically implausible. For example, we do not want the elements to stray too far from their initial position.

A permutation matrix allows us to track the new positions of the elements. The corresponding effect of one permutation, $\pi_1 = \begin{pmatrix} 1 & 2 & \underline{4} & \underline{3} & 5 \end{pmatrix}$, on the permutation matrix $P_{\pi_1} = (p_{ii'})$ is:

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \underline{1} & 0 & \dots \\ 0 & 0 & \underline{1} & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & \dots \end{array}\right] \times \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ 0.210 \\ \underline{0.000} \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.000 & + \dots \\ 0.000 & + \dots \\ 0.210 & + \dots \\ 0.000 & + \dots \\ 0.000 & + \dots \end{bmatrix}$$

If we were to follow the first stock jump, which is element p_{144} in the co-permutation matrix, it shifts from a position on the diagonal $(4, 4)$ in the identity matrix to a higher position $(3, 4)$. The changes occur in the vertical, i, dimension. The jump shifts 1 spot backward. We can follow each element in their respective columns to see the corresponding change in position. The changes in position for each element in the column are equal to $\{0, 0, +1, -1, 0\}$. One swap results in 2 moves, one forward and one backward.

A. Distance

We track the distance traveled of an element compared to its original position in the identity matrix. We denote the distance metric for the permutation in one column formally as $d(P_{\pi_l})$, with $l = 1, ..., q$. The distance matrix D_n tracks the total number of shifts in each permutation matrix. If we are interested in the total number of moves compared to the identity matrix, the distance matrix is equal to:

$$D_n = (d_{ii'}) \begin{bmatrix} 0 & 1 & \cdots & h-2 & h-1 \\ 1 & 0 & \cdots & h-3 & h-2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h-2 & h-3 & \cdots & 0 & 1 \\ h-1 & h-2 & \cdots & 1 & 0 \end{bmatrix} \tag{25}$$

Keeping the elements on the diagonals results in a zero distance.

For example, the distance matrix for a column in our jump-event matrix which consists

of $h = 5$ elements is:

$$D_n = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & \underline{\underline{1}} & 2 \\ 3 & 2 & \underline{1} & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

If the elements in the permutation matrix remain on the diagonal, the distance is equal to zero.

The total number of moves for the lth column in the jump-event matrix is equal to the following matrix product:

$$d(P_{\pi_l}) = \text{vecr}\left(P_{\pi_l}\right) \times \left(\text{vecr}\left(D_n\right)\right)^\top, \text{ for } l = 1, ..., q \tag{26}$$

The vectorization, vecr$(\cdot)$ sticks the rows together, leading to a $1 \times hq$ row-vector. We transpose the second term after vectorization to get a $hq \times 1$ column-vector. The result is a row-wise multiplication of the elements, which equals the total number of swaps. The total distance does not distinguish between the direction of the move.

Swapping two elements in the permutation $\pi_1 = \begin{pmatrix} 1 & 2 & \underline{4} & \underline{3} & 5 \end{pmatrix}$ leads to 2 total shifts:

$$d(P_{\pi_1}) = \text{vecr}\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{\underline{1}} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right) \times \text{vecr}\left(\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & \underline{\underline{1}} & 2 \\ 3 & 2 & \underline{1} & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}\right)^\top = 2$$

B. Backward and forward moves

The distance matrix (25) treats the moves as symmetric. By taking the upper triangular portion D_n^U and lower triangular portions D_n^L of the distance matrix D_n, we can focus on backward and forward moves Upward moves in the permutation matrix are backward moves in time, downward moves in the permutation matrix are forward moves in time. For the above permutation the total number of backward moves is equal to 1:

$$d(P_{\pi_1}) = \text{vecr}\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{\underline{1}} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right) \times \text{vecr}\left(\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & \underline{\underline{1}} & 2 \\ 0 & 0 & \underline{0} & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right)^\top = 1$$

C. The moves of one particular element

The distance matrix summarizes the number of moves for one stock jump vector – *i.e* within one column of the jump-event matrix. A swap of a jump and a non-jump in one column of the jump-event matrix results in 2 moves. We can focus on the move of just one observation, such as the arrival of jump, by setting the elements in the columns $i' \neq i^*$, with i^* the original arrival time of the jump, in the permutation matrix to zero. If we were to focus on the moves of the jump in the first column which arrives originally during the 4th period, we set the columns 1, 2, 3 and 5 in the permutation to zero:

$$d(P_{\pi_1}) = \text{vecr}\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{\underline{1}} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right) \times \text{vecr}\left(\begin{bmatrix} 0 & 0 & 0 & \underline{3} & 0 \\ 0 & 0 & 0 & \underline{2} & 0 \\ 0 & 0 & 0 & \underline{\underline{1}} & 0 \\ 0 & 0 & 0 & \underline{0} & 0 \\ 0 & 0 & 0 & \underline{1} & 0 \end{bmatrix}\right)^{\top} = 1$$

D. Example: The penalized best rearrangement

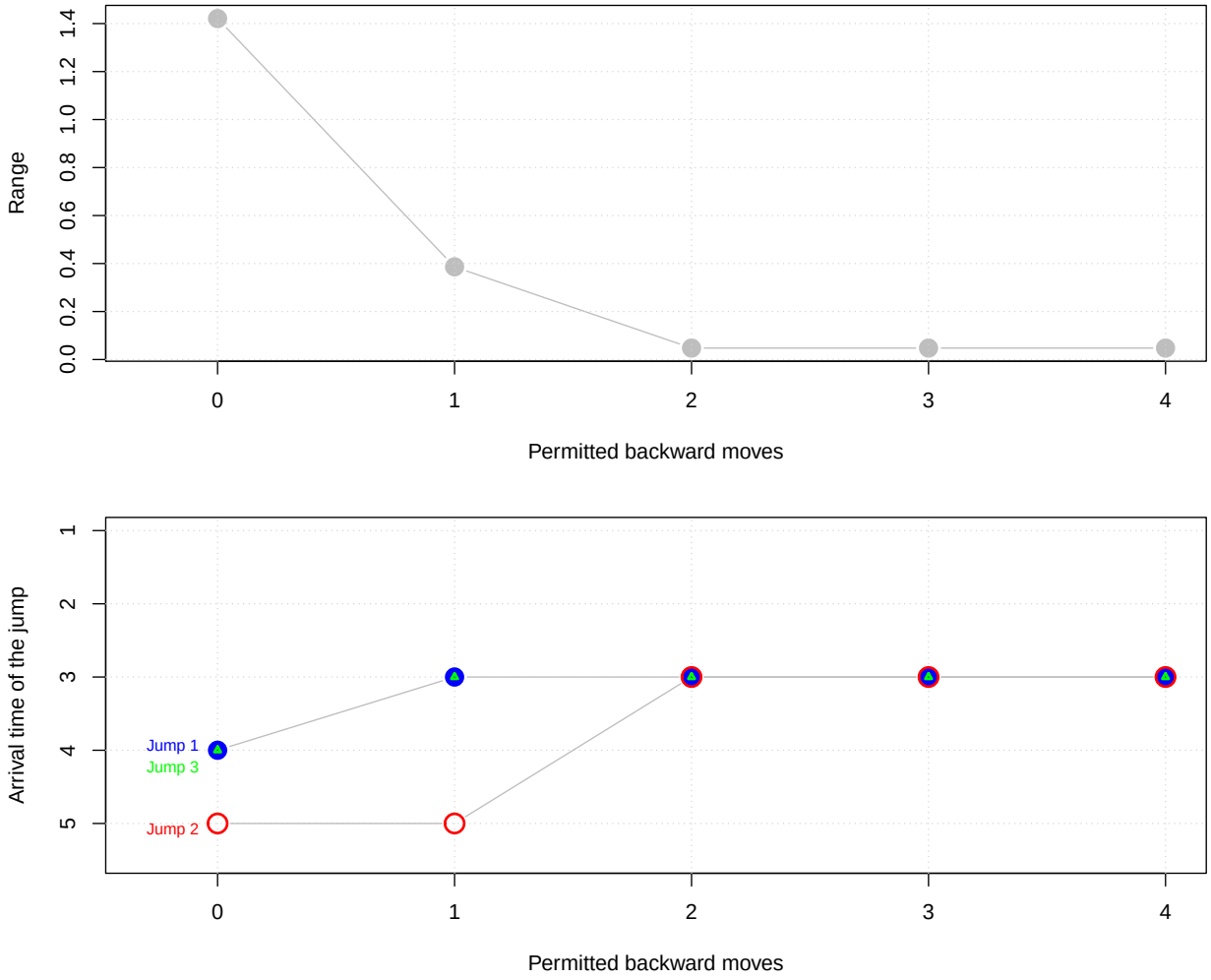
The linear program imposes bounds on the estimated coefficients (the arrival times) of a combinatorics problem by imposing a shift constraint. We only allow for backward moves in this article because we assume that stock prices are sluggish and lag the highly liquid and carefully watched ETF, they do not lead it. Therefore, we only permit stock jumps to be moved to an earlier time, not a later time:

$$d(P_{\pi_l}) \leq c, \text{ for } l = 1, ..., q-1 \tag{P1}$$

in which c is the maximum permitted size of the move. We do not impose a shift constraint for the target column, with $l = q$, because the linear program keeps it fixed either way. If we have a fixed constant c, this results in a linear program with $q-1$ constraints. We solve the linear program for multiple upper boundaries to obtain figures like the one below.

Figure 6 shows the range, *i.e.*, flatness, and the jump arrival times as a function of permitted moves. We allow for at most 4 shifts backwards and solve the linear program for each shift constraint. When we impose no moves in the jump-event matrix, the linear program returns back the observed jump-event event matrix, with the a gradual jump of stock A at times 4 and 5, that is, the 4th and 5th row in the jump-event matrix. When we allow for more shifts, the return spreads flatten. The smallest range for our toy example occurs when we allow for two backward shifts, in which the arrival times of all the jumps align in the third period.

Figure 6: Range of the return spreads and jump arrival times as a function of permitted moves



3 Empirics

We apply our rearrangement procedure to investigate the cross-section of Dow 30 stock jumps during ETF jump events.

3.1 Data set: The Dow and the DIA ETF

We extract the equity data from the NYSE Trade and Quote (TAQ) database and include trade data with millisecond precision timestamps. The proxy for the market is the SPDR Dow Jones Industrial Average ETF (DIA), an index fund tracking the daily movement of the Dow Jones Industrial Average. We study the response of an unbalanced panel of

Dow 30 stocks.[2] We construct a synthetic index comprised of the Dow 30 stocks like in Bollerslev et al. (2008).

The data cover the period the beginning of 2007 until the beginning of 2020. Our sample period includes several episodes of exceptional turbulence, such as global housing and credit crisis, the European sovereign debt crisis, the bail-out of Greece, and the Japanese earthquake, the recent covid-19 crisis, as well as other major political and economic events that have caused big moves on the financial market. The data set includes a lot of market jump events with a diverse set of economic forces behind them.

We adopt the pre-filtering routine of Barndorff-Nielsen et al. (2009). We remove banking holidays, half-trading days, any day where there more than two hour time difference between consecutive trades and periods of malfunctioning such as the 2010 flash crash. We sample prices at the one-minute frequency.

3.2 Rearranging stock jumps: An empirical illustration

As an illustration of the local behaviour of the cross-section of jumps, we investigate an example of a sluggish cojump in the DIA basket instrument and some of its underlying components. On September 18, 2007 the Federal Reserve announced[3] rate cuts; a bold action but risky action according to market anecdotes[4].

Figure 7 shows that when one examines high-frequency data, one often finds that the ETF and a synthetic stock portfolio do not always move perfectly in lockstep, especially during the arrival of news. We plot the price path of the Dow ETF and a synthetic Dow 30 portfolio, the spreads in the prices (7) and the returns (9) right before and after the FOMC Statement. We mark the ETF jump with a red circle.[5] The ETF jumps at 14:16,

[2]The 43 unique tickers are AA, AAPL, AIG, AXP, BA, BAC, C, CAT, CSCO, CVX, DD, DIS, DOW, DWDP, GE, GM, GS, HD, HON, HPQ, IBM, INTC, JNJ, JPM, KFT, KO, MCD, MMM, MO, MRK, MSFT, NKE, PFE, PG, T, TRV, UNH, UTX, V, VZ, WBA, WMT and XOM.

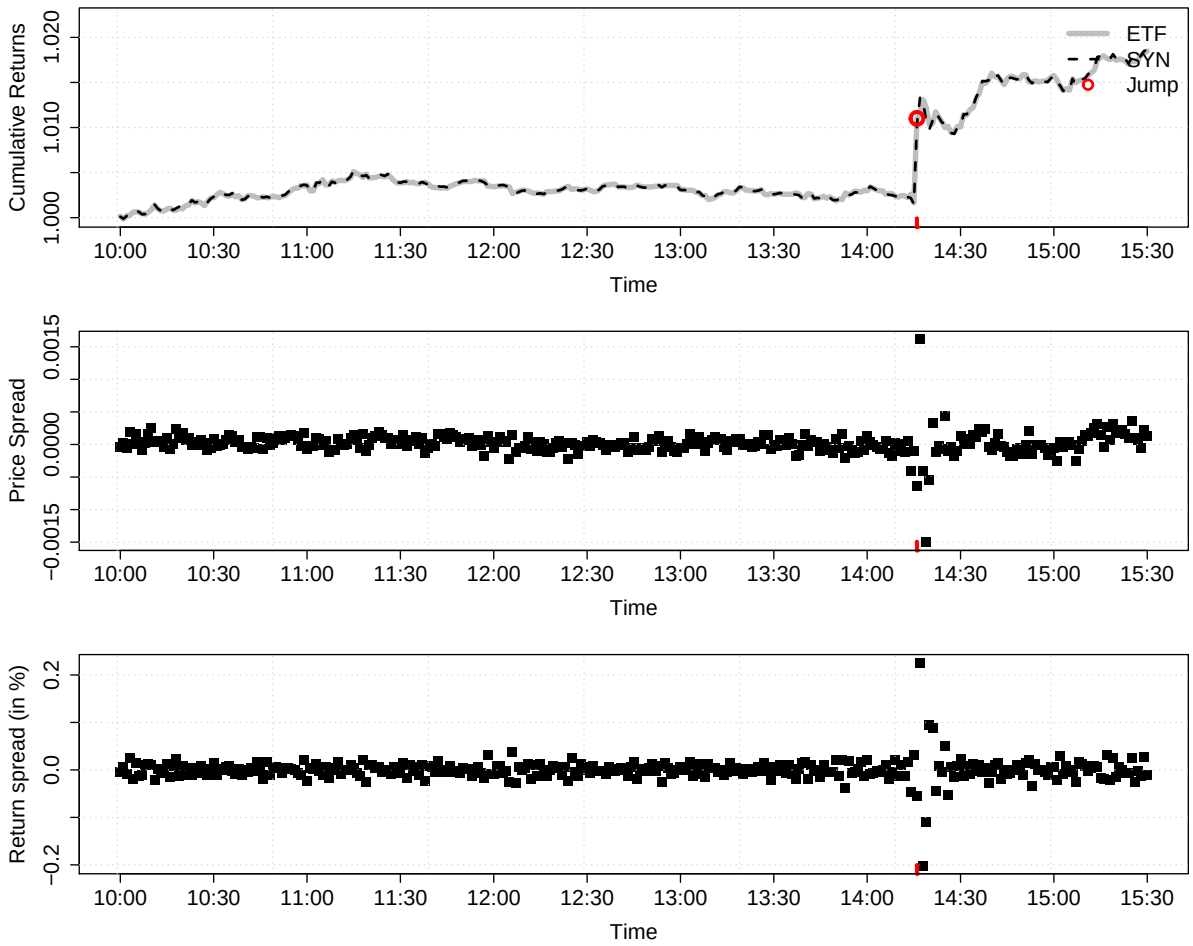
[3]Press release and the related FOMC Meeting Statement.

[4]See, for example, the coverage in The Economist and the Financial Times: Bernanke's bounty, Instant reaction: Response to the Fed, The Short View: Fed decision — it's all about game theory, Overview: US equities and oil surge after rate cuts, Cheering greets Fed announcement, Fed must weigh inflation against recession, Bold Fed goes for half-point cut, Bank acts boldly to avert recession risk, Fed cut: Pundits speak, Fed slashes rates, and Feeling ecstatic? Mind the e-Ben-der

[5]We apply Mancini (2001)'s univariate jump test to identify jumps. We set the threshold level for

U.S. Eastern Time, one minute after the release of the statement, with a size equal to 0.938 percent. If news reached the entire market instantly and trading were continuous, jumps in a group of stocks should presumably occur simultaneously with the ETF index jump and the spreads should be randomly scattered. The ETF jump in the example, however, is followed by an increase in the spreads. Markets took some time to incorporate and adjust the information on the Fed's economic plans into the stock's observed prices.

Figure 7: An increase in the spread right after an FOMC Statement



We zoom in on the jump vectors using a jump-event matrix across an window from ten minutes before to ten minutes after the index jump. There are 46 stock jumps in the event window, of which 27 occur at the same time as the ETF jump, meaning that

identifying ETF jump events like in Li et al. (2019): one-minute increments which exceed in absolute value 7 local volatility estimates.

19 stock jumps are lagging the ETF jump. We use the rearrangement linear program to synchronize the asynchronous stock jumps. We impose two additional constraints. Because some stocks already jump with the index, we impose that the stocks that already jump with the index, cannot move. We also impose that the linear program cannot rearrange sluggish stock jumps before the ETF jump, because we assume that no stock jumps earlier than it should, that is, before the highly liquid and carefully watched ETF.

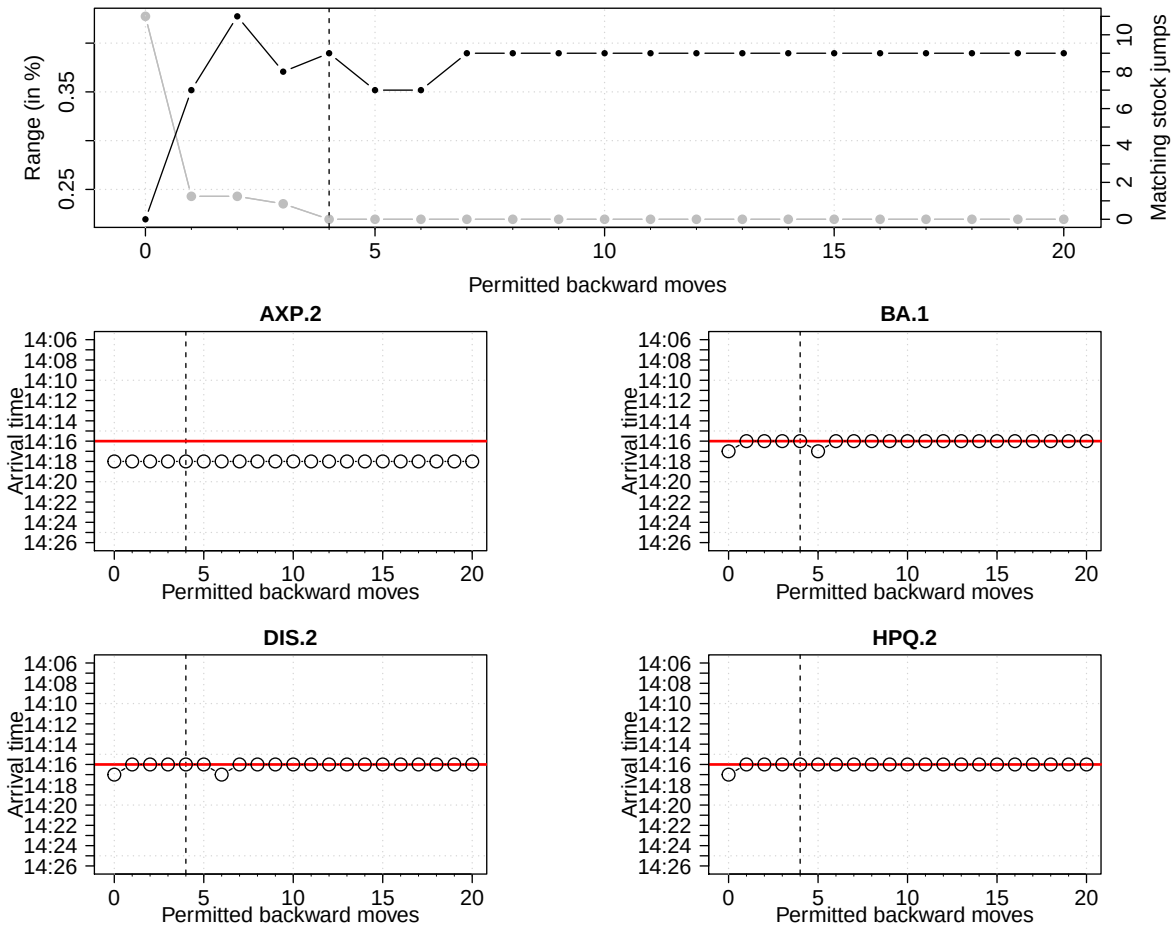
Figure 8 shows the rearrangement of the asynchronous stock jumps for this event. The top panel shows the range of the return spreads (in gray) and the number of matched stocks (in black) as a function of the permitted moves. The range is a downward curve – as we allow for more changes in the data – and we use it to determine the number of permitted backward moves a jump can make. The range is minimal as soon as the linear program can rearrange the stocks 4 steps backwards (dashed line). This rearrangement matches 9 out of 19 scattered stock jumps with ETF, recovering the common jump in the stocks. The other panels show the implied arrival time of some of the individual stock jumps. We mark the ETF jump arrival time with a red line. Some stock jumps like the second one of AXP never match the ETF jump, but some stock jumps do stick to the red line after rearrangement and now jump at the same time as the index.

4 Concluding remarks

Sluggish news reactions manifest as gradual jumps and jump delays. In a panel of high-frequency intraday stock returns, these noisy jumps show up as a "sluggish cojump", *i.e.* jumps observed at close but distinct points in time. We introduce the tools to synchronize the scattered jumps in a panel of stock prices and recover the efficient common jump. The intuition behind our approach if fairly simple: we rearrange the stock jumps in time to reflect that the latent prices of the stocks and the ETF move in lockstep.

The ideas and results in this paper are suggestive of an empirical extension which exploits the information in the rearranged stock jumps. The information in the rearranged stock jumps is likely to improve risk management decisions. Casual simulations suggest that asynchronicity in the stock jumps influences realized variances, covariances and betas (see *e.g.*, Bollerslev et al., 2020, 2021); as expected, the realized measures shoot up after synchronization. A thorough analysis must, however, await future work.

Figure 8: Rearrangement of the asynchronous stock jumps



References

Aït-Sahalia, Y. (2004). Disentangling diffusion from jumps. *Journal of Financial Economics 74*(3), 487–528.

Aït-Sahalia, Y. and J. Jacod (2014). *High-frequency financial econometrics*. Princeton University Press.

Andersen, T. G., I. Li, Archakov, G. Cebiroglu, and N. Hautsch (2021). Local mispricing and microstructural noise: A parametric perspective. *Journal of Econometrics (Forthcoming)*.

Bandi, F. M., D. Pirino, and R. Renò (2017). Excess idle time. *Econometrica 85*(6), 1793–1846.

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2009). Realized kernels in practice: Trades and quotes. *Econometrics Journal 12*(3), C1–C32.

Bernard, C., O. Bondarenko, and S. Vanduffel (2018). Rearrangement algorithm and maximum entropy. *Annals of Operations Research 261*(1), 107–134.

Bernard, C., L. Rüschendorf, and S. Vanduffel (2017). Value-at-risk bounds with variance constraints. *Journal of Risk and Insurance 84*(3), 923–959.

Bollerslev, T., T. H. Law, and G. Tauchen (2008). Risk, jumps, and diversification. *Journal of Econometrics 144*(1), 234–256.

Bollerslev, T., A. J. Patton, and R. Quaedvlieg (2020). Realized semicovariances. *Econometrica (Forthcoming)*, 1–44.

Bollerslev, T., A. J. Patton, and R. Quaedvlieg (2021). Realized semibetas: Disentangling "good" and "bad" downside risks. *Journal of Financial Economics (Forthcoming)*.

Boudt, K., E. Jakobsons, and S. Vanduffel (2018). Block rearranging elements within matrix columns to minimize the variability of the row sums. *4OR 16*(1), 31–50.

Boudt, K. and M. Petitjean (2014). Intraday liquidity dynamics and news releases around price jumps: Evidence from the djia stocks. *Journal of Financial Markets 17*, 121–149.

Christensen, K., R. C. Oomen, and M. Podolskij (2014). Fact or friction: Jumps at ultra high frequency. *Journal of Financial Economics 114*(3), 576–599.

Cohen, K. J., G. A. Hawawini, S. F. Maier, R. A. Schwartz, and D. K. Whitcomb (1980). Implications of microstructure theory for empirical research on stock price behavior. *Journal of Finance 35*(2), 249–257.

Cohen, L., C. Malloy, and Q. Nguyen (2020). Lazy prices. *Journal of Finance 75*(3), 1371–1415.

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics 7*(2), 174–196.

Edwards, W. (1982). Conservatism in human information processing. In D. Kahneman, P. Slovic, and A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases*, Chapter 25, pp. 359–369. Cambridge university press.

Embrechts, P., G. Puccetti, and L. Rüschendorf (2013). Model uncertainty and VaR aggregation. *Journal of Banking & Finance 37*(8), 2750–2764.

Epps, T. W. (1979). Comovements in stock prices in the very short run. *Journal of the American Statistical Association 74*(366a), 291–298.

Hasbrouck, J. (2003). Intraday price formation in us equity index markets. *Journal of Finance 58*(6), 2375–2400.

Jondeau, E., S.-H. Poon, and M. Rockinger (2007). *Financial modeling under non-Gaussian distributions*. Springer Science & Business Media.

Lahaye, J., S. Laurent, and C. J. Neely (2011). Jumps, cojumps and macro announcements. *Journal of Applied Econometrics 26*(6), 893–921.

Lee, S. S. and P. A. Mykland (2008). Jumps in financial markets: A new nonparametric test and jump dynamics. *Review of Financial Studies 21*(6), 2535–2563.

Lee, S. S. and P. A. Mykland (2012). Jumps in equilibrium prices and market microstructure noise. *Journal of Econometrics 168*(2), 396–406.

Li, J., V. Todorov, G. Tauchen, and R. Chen (2017). Mixed-scale jump regressions with bootstrap inference. *Journal of Econometrics 201*(2), 417–432.

Li, J., V. Todorov, G. Tauchen, and H. Lin (2019). Rank tests at jump events. *Journal of Business & Economic Statistics 37*(2), 312–321.

Mancini, C. (2001). Disentangling the jumps of the diffusion in a geometric jumping brownian motion. *Giornale dell'Istituto Italiano degli Attuari 64*(19-47), 44.

Puccetti, G. and L. Rüschendorf (2012). Computation of sharp bounds on the distribution of a function of dependent risks. *Journal of Computational and Applied Mathematics 236*(7), 1833–1840.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.