Sluggish news reactions: A combinatorial approach for synchronizing stock jumps

Authors	Nabil Bouamara, Kris Boudt, Sébastien Laurent, and Christopher J. Neely
Working Paper Number	2024-006A
Creation Date	March 2024
Citable Link	https://doi.org/10.20955/wp.2024.006
Suggested Citation	Bouamara, N., Boudt, K., Laurent, S., Neely, C.J., 2024; Sluggish news reactions: A combinatorial approach for synchronizing stock jumps, Federal Reserve Bank of St. Louis Working Paper 2024-006. URL https://doi.org/10.20955/wp.2024.006

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

Nabil Bouamara
Research Division, National Bank of Belgium

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 Univ., CNRS, AMSE, Marseille, France
Aix-Marseille Graduate School of Management-IAE

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

March 26, 2024

Abstract

Stock prices often react sluggishly to news, producing gradual jumps and jump delays. Econometricians typically treat these sluggish reactions as microstructure effects and settle for a coarse sampling grid to guard against them. Synchronizing mistimed stock returns on a fine sampling grid allows us to better approximate the true common jumps in related stock prices.

Keywords: Asynchronicity; Cojumps; High-frequency data; Microstructure noise; Realized Covariance; Rearrangement

JEL: C02, C58, G11, G14

---

*We have received helpful comments and suggestions from Andres Algaba, Geert Dhaene, Jean-Yves Gnabo, Roxana Halbleib, Ilze Kalnina, Nathan Lassance, Oliver Linton, André Lucas, Kristien Smedts, Lisa Van den Branden, Steven Vanduffel, Brecht Verbeken, and the conference and seminar participants at KU Leuven, Vrije Universiteit Brussel, Vrije Universiteit Amsterdam, the Computational and Financial Econometrics Conference (2021), the Belgian Financial Research Forum (2023), the Quantitative and Financial Econometrics Conference (2023), and the Society of Financial Econometrics Summer School (2023). Nabil Bouamara gratefully acknowledges support from the Flemish Research Foundation (FWO fellowship #11F8419N) and the Platform for Education and Talent (Gustave Boël – Sofina fellowship). Sébastien Laurent has received fundings from the french government under the "France 2030" investment plan managed by the French National Research Agency (reference: ANR-17-EURE-0020 and ANR-21-CE26-0007-01) and from Excellence Initiative of Aix-Marseille University - A*MIDEX. The views expressed in this paper are those of the author(s) and do not necessarily reflect the views of the Federal Reserve System, the Board of Governors, the regional Federal Reserve Banks, or the National Bank of Belgium. Federal Reserve Bank of St. Louis Working Papers are preliminary materials circulated to stimulate discussion and critical comment.

1 Introduction

Major economic news, such as FOMC announcements, natural disasters, or geopolitical conflicts, impact financial markets by triggering jumps in related high-frequency stock prices (Li et al., 2017, document empirical examples of this phenomenon). Statistical tests for these common jump events, known as "cojump tests", implicitly assume that jumps happen at the same time across all relevant assets but, in fact, jumps occur asynchronously in transaction data. Empirical studies support this view: stock prices can move sluggishly (Bandi et al., 2017), jumps may develop gradually (Barndorff-Nielsen et al., 2009), jumps of less-liquid individual assets typically lag those of the more-liquid market index (Li et al., 2017), and at an ultra-high frequency, cojumps are spurious (Bajgrowicz et al., 2016). Most researchers have dealt with this problem by settling for a coarse sampling grid (Barndorff-Nielsen et al., 2009; Bollerslev et al., 2008; Lahaye et al., 2011; Li et al., 2019). Such a coarse grid guards against microstructure effects, the frictions with which actual trades take place, but it is restrictive in that it oversmooths actual changes (Aït-Sahalia, 2004).

We propose an alternative approach to recover the common jump in a basket of stocks on a fine sampling grid. Our method assumes that the exchange-traded fund (ETF) price tracks the latent, equilibrium (also often called "efficient") value of a stock index, and that sluggish news impoundment creates a spread between the value of a synthetic index and the efficient ETF price. We rearrange stock price jumps in time to minimize this spread and thus (approximately) recover the latent, common, efficient jump.

Combinatoric methods optimally rearrange asynchronous jumps. We extend the pioneering work of Puccetti and Rüschendorf (2012) and Embrechts et al. (2013) on rearrangements. Their algorithm is best known for actuarial applications, but can also be applied to other disciplines, such as option pricing (*e.g.*, Bondarenko and Bernard, 2023) and operations research (*e.g.*, Boudt et al., 2018). Rearrangements can also synchronize stock jumps on a fine sampling grid, provided we penalize economically implausible rearrangements. For example, we only allow for a rearrangement of jumps backward in time because we assume that stock prices are sluggish and lag the highly liquid and carefully watched ETF, they do not lead it. To operationalize this, we introduce the "Rearrangement Mixed-Integer Linear Program."

We apply our methods to investigate the reactions of Dow 30 stock prices in event windows around DIA ETF jumps. For example, the Federal Reserve announced rate cuts on September 18, 2007, at 14:15 US Eastern Time, after which markets took up to five minutes to incorporate the Fed's news into the Dow 30 stock's prices. Rearrangements synchronize 19 (out of 23) scattered stock jumps with the ETF jump, approximately recovering the common jump in the stocks. This is not an isolated incident: our linear program rearranges stock jumps during 180 separate cojump events.

Synchronizing mistimed stock returns improves estimates of the daily realized covariance matrix and improves the out-of-sample financial performance compared to using raw returns. While other estimators, like the multivariate realized kernel in Barndorff-Nielsen et al. (2011) or a Cholesky factorization in Boudt et al. (2017), protect against mild market microstructure noise and the Epps (1979) effect (that is, the downward bias in covariance estimates due to asynchronous trading), they do not protect against the underestimation of jump dependence due to asynchronous jumps.

We proceed as follows. Section 2 details the synchronization method using a toy example. Section 3 illustrates an empirical example of a rearranged sluggish cojump in the Dow 30 and includes a portfolio allocation exercise. Section 4 concludes.

2 Synchronizing jumps: A combinatorial problem

Trading is non-synchronous at high frequencies; This leads to so-called "stale" prices at irregularly spaced times, differing across assets. Addressing asynchronicity through the coordinated collection of multivariate data has been an active area of research in financial econometrics in recent years, see *e.g.*, Barndorff-Nielsen et al. (2011) or Boudt et al. (2017) and the references therein, and the topic has been integral to covariance estimations since at least Epps (1979). Nonetheless, existing sampling schemes like refresh-time sampling (Barndorff-Nielsen et al., 2011), are not designed to handle price jumps, because stale prices are not the only cause of asynchronous jumps. Sometimes asset prices may be "sluggish"; news might be impounded in prices with a delay, although the asset is trading. To address this problem, we synchronize the timing of multivariate jumps using what we call "Jump Sampling". This technique refines the detection of high-frequency cojumps and the realized covariance matrix.

Figure 1 compares refresh-time sampling to jump sampling in the presence of asynchronous observations and jumps. It draws inspiration from Figure 1 in Barndorff-Nielsen et al. (2011), illustrating refresh-time in a situation with three assets (without jumps). We expand upon this concept to include scenarios with asynchronous price jumps, focusing on three assets: a basket instrument and its two underlying stocks. In each asset's case, the filled dots in Figure 1 indicate the updates in posted prices, and an open dot pinpoints the time at which the price jumps. Vertical dashed lines represent the sampling times generated from the three assets, using refresh-time sampling. For example, the first black dot represents the time it has taken for all three assets to trade. But because asynchronous jumps are typically not solely attributed to (il)liquidity issues[1], refresh-time sampling does not resolve the asynchronicity inherent in the jumps.

To confront asynchronicity, our new jump sampling scheme generally rearranges mistimed jumps to occur simultaneously with the ETF jump, as shown in the lower section of the diagram. In what follows, we detail how we optimally rearrange jumps, penalizing economically implausible rearrangements. A simulated example clarifies the mechanics of the rearrangements.

2.1 A DGP for sluggish news reactions

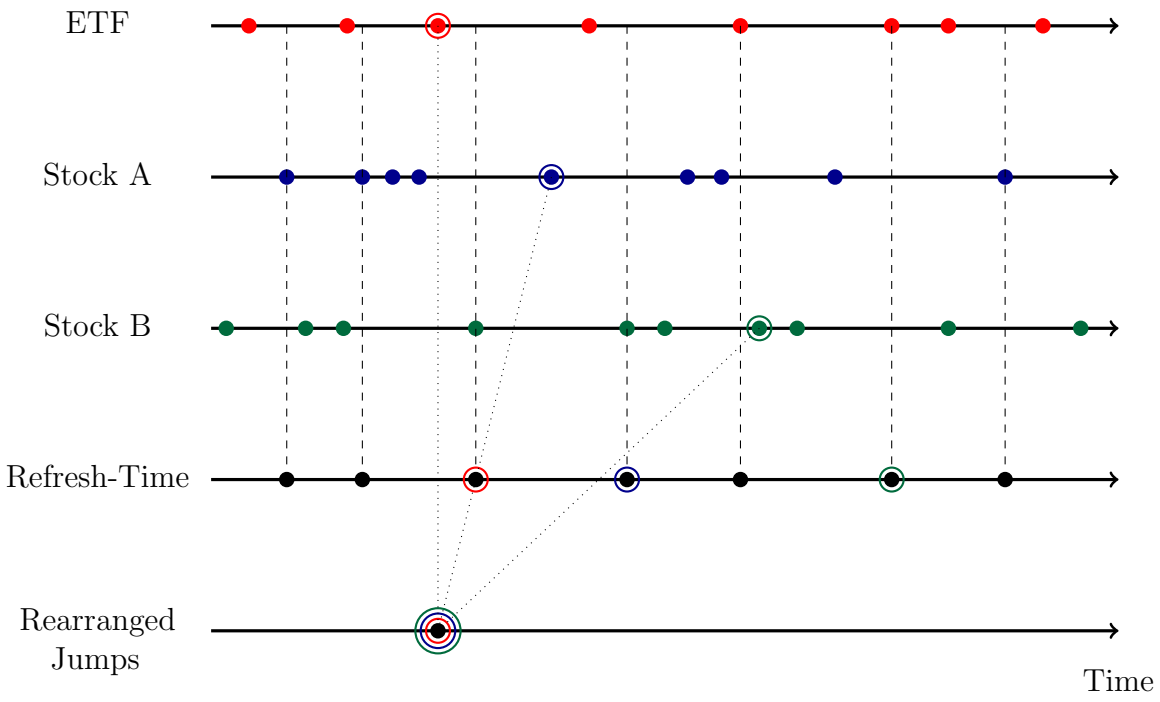
We assume a data generating process with gradual and delayed jumps for the prices of the stocks in the index.[2] Trading frictions may prevent the observed prices from immediately reflecting a jump in the underlying equilibrium price. The conventional martingale-plus-noise price model does not capture such complications, which are important in the empirical analysis of multivariate jump processes.

Let $X_t = (X_{1,t}, ..., X_{p,t})^\top$ denote the logarithmic p-variate, equilibrium (or so-called "efficient") price of the p stocks in the market index. 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 infor-

[1] Asynchronous jumps in asset prices happen for several reasons, not just differences in liquidity. Assets may react differently to news based on the traders involved, their beliefs, and actions. For example, major news like the FOMC statement on September 18, 2007, discussed in Section 3.2, triggered an increase in trading, unlike the typical activity seen throughout the rest of the day (not reported). This suggests that time variation in trading intensity contributes to asynchronous jumps.

[2] Gradual jumps are when the prices exhibit strong linear trends for periods of a few minutes (Barndorff-Nielsen et al., 2009). Jump delays are when jumps of individual assets follow those of the highly liquid market index during market-wide events (Li et al., 2017).

Figure 1: Jump sampling



Note: This diagrams compares refresh-time sampling to jump sampling in the presence of asynchronous observations and jumps. In each asset's case, the filled dots indicate when the posted prices were updated, and the open dot represents the time at which the price jumps. The vertical dashed lines represent the sampling times generated from the three assets using refresh-time sampling. Jump sampling rearranges mistimed jumps to occur simultaneously with the ETF jump.

mation available to market participants at time t, with $t \geq 0$. We assume that X operates in an arbitrage-free, frictionless market, which implies that X is a semimartingale. Econometricians (see Aït-Sahalia and Jacod, 2014) model stock prices as a jump-diffusion process:

$$\begin{aligned} X_t &= X_t^c + X_t^d, \text{ with,} \\ X_t^c &:= X_0 + \int_0^t b_s ds + \int_0^t \sigma_s dW_s, \\ X_t^d &:= \textstyle\sum_{s \leq t} \Delta X_s, \end{aligned} \tag{1}$$

in which $t \geq 0$, b is the drift, σ is the stochastic (co)volatility, W is a multivariate Brownian motion and $\Delta X_t := X_t - X_{t-}$, with X_{t-} the left limit at time t, denotes the jumps of X at time t. A stock's growth prospects generates a jump in single stock price (Christensen et al., 2023; Lee and Mykland, 2008) but major economic news, such as pre-scheduled announcements, natural disasters or geopolitical conflicts, trigger common (*i.e.*, synchronous) jumps in related stock prices (see Li et al., 2017, for examples).

In practice we observe discretely sampled, noisy transaction prices, rather than the latent process in (1). Frictions such as tick size, discrete observations, bid-ask spreads, adverse selection, liquidity and inventory control produce market microstructure noise (see *e.g.*, Christensen et al., 2014; Lee and Mykland, 2012; Li and Linton, 2022). Additionally, prices may also be sluggish; market participants must trade to reveal private information and reach a

consensus about the impact of some piece of news. If trades do not occur at the time of a jump in the underlying efficient price, then observed reactions can be delayed. That is, jumps in observed prices can lag those in the efficient price, even if market participants are constantly aware of fundamentals.

We model the observed log price process $Y_t = (Y_{1,t}, ..., Y_{p,t})^\top$ of the p stocks as contaminated version of (1), observed at discrete time points $i\Delta_n$, with $i = 0, ..., \lfloor T/\Delta_n \rfloor$, across a time span T, where $\lfloor \cdot \rfloor$ denotes the floor function:

$$\begin{aligned} &Y_{i\Delta_n} = Y^c_{i\Delta_n} + Y^d_{i\Delta_n}, \text{ with,} \\ &Y^c_{i\Delta_n} := X^c_{i\Delta_n} + u_{i\Delta_n}, \\ &Y^d_{i\Delta_n} := \textstyle\sum_{h\Delta_n \leq i\Delta_n} \Delta Y_{h\Delta_n}. \end{aligned} \tag{2}$$

There are two kinds of noise in (2): microstructure noise and mistimed jumps. Microstructure noise u contaminates the efficient price process X, but is typically too small to substantially contaminate the discontinuous part X^d. It cannot generate gradual jumps (as in Barndorff-Nielsen et al., 2009) or jump delays (as in Li et al., 2017). We capture the mistimed or mismeasured jumps in a separate noisy jump component Y^d, allowing a sluggish news reaction, which spreads the stock jump across several time intervals.

Figure 2 shows a simulated sample path of this new DGP (2) for second-by-second prices of a single stock. The gray path in the top panel of Figure 2 illustrates that the efficient stock price jumps at 12:45 in reaction to news. In the following 112 seconds, the observed price (in black) catches up with the new equilibrium level (in gray) by gradually matching the jump. The middle and bottom panels respectively decompose the price process into its continuous Brownian component and its discrete jump component. The middle panel compares the efficient, continuous price trajectory with that contaminated by mild market microstructure noise. The bottom panel compares the efficient, abrupt jump with the contaminated, more gradual jump. Our assumed DGP uses a step function to model the incorporation of news into observed prices. Appendix A shows how this assumed DGP spreads the jump across several time intervals.

The observed price process (2) combines the frictions and the sampling frequency. There is less noise on a coarse sampling grid, *i.e.*, at a lower sampling frequency Δ_n, but data at such lower frequencies tend to oversmooth actual changes (as in Bollerslev et al., 2008; Lahaye et al., 2011; Li et al., 2019). The finer the sampling grid, the higher the probability that a jump can be recognized as such (Aït-Sahalia, 2004).

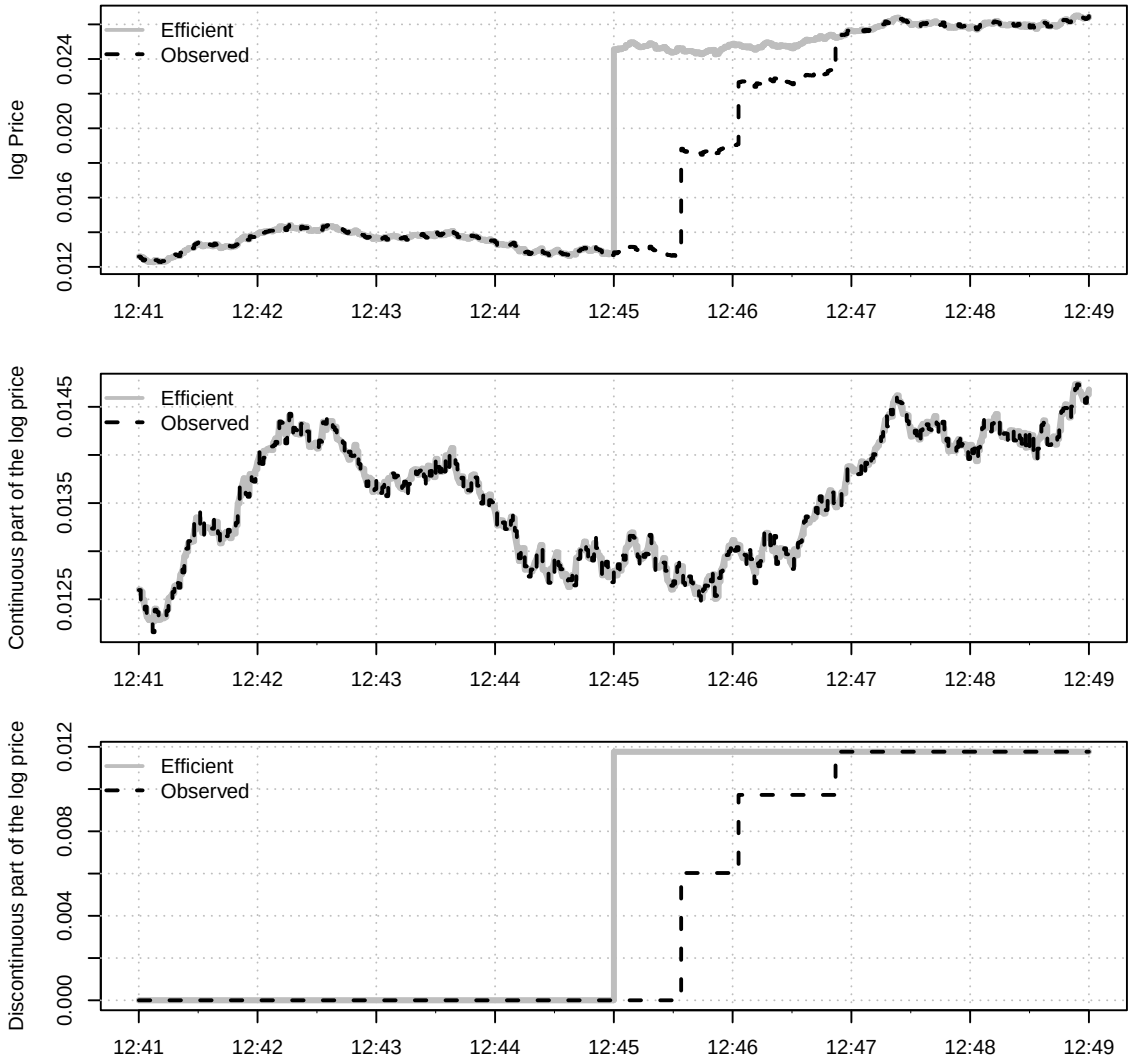
2.2 Collecting asynchronous jumps in a jump-event matrix

When multiple stocks react sluggishly to new information, their jumps will generally be asynchronous on a fine sampling grid, and, because of this, these jumps will not all coincide with the jump in the price of an index-tracking ETF. Empirical evidence corroborates this prediction: jumps of less-liquid individual assets typically lag those of the more-liquid market index (Li et al., 2017) and the ETF jumps more often than a synthetically constructed index of stocks (Bollerslev et al., 2008).

2.2.1 The spread measures sluggishness in high-frequency data

Let $w_{k,t}$, with $k = 1, ..., p$, be the weights allocated to each stock in the market index at each moment in time. The price of the synthetically constructed index portfolio, $S_{i\Delta_n}$, is a linear

Figure 2: The discontinuous component of a stock's price may react sluggishly to news



Note: We plot the decomposition of the efficient (1) and observed (2) log price process for one sample path during a short event window. We simulate second-by-second ($\Delta_n = 1/23,401$) efficient and observed prices for 1 stock across one trading day. The top panel shows the log price. The middle panel shows the continuous part of the log price, and the bottom panel shows the discontinuous part of the log price. The efficient stock price jumps at $i\Delta_n = 11,701$ or 12:45:00. Mild market microstructure noise contaminates the continuous part of the price process, while delays in the jump process contaminate the discontinuous part of the price process.

combination of the observed stock prices in (2), sluggishly incorporating its jump component:

$$S_{i\Delta_n} = \sum_{k=1}^{p} w_{k,i\Delta_n} Y_{k,i\Delta_n}. \tag{3}$$

An ETF log price process, Z_t, tracks an index of the p stocks. We assume that the observed log price, Z_t, replicates a portfolio of efficiently priced stocks (1), efficiently incorporating

their jumps[3]:

$$Z_t = \sum_{k=1}^{p} w_{k,t} X_{k,t}. \tag{4}$$

The deviation or spread in prices is the difference between the observed prices on a synthetic index of stocks (4) and the prices of an observable ETF trading the index (3):

$$\delta^p_{i\Delta_n} := S_{i\Delta_n} - Z_{i\Delta_n}. \tag{5}$$

Similarly, we can also define the spread in returns as the difference between the observed returns on a synthetic index of stocks $\Delta^n_i S := \sum_{k=1}^{p} w_{k,i\Delta_n} \Delta^n_i Y_k$ and the returns of an observable ETF trading the index $\Delta^n_i Z := Z_{i\Delta_n} - Z_{(i-1)\Delta_n}$ or, equivalently, the percentage change of the price spread in (5):

$$\delta^r_{i\Delta_n} := \delta^p_{i\Delta_n} - \delta^p_{(i-1)\Delta_n} = \Delta^n_i S - \Delta^n_i Z. \tag{6}$$

In the absence of microstructure noise or sluggish prices, we would expect the ETF log price Z to equal the synthetic log price S. However, these types of noise cause the price of the synthetic index portfolio to deviate from the presumably efficient price of an ETF tracking the index.[4] The sluggish jump component is much larger than microstructure noise, so asynchronous impoundment of news drives the spread in prices. Hence, the spread (5) between the ETF price and a synthetically constructed index acts as a measure of "sluggishness" – the collective misalignment of noisy stock prices with their efficient levels. The goal is to rearrange these jumps in time, essentially changing their time labels, to minimize the spread and approximately recover the latent efficient price.

To illustrate our procedures, we consider a stylistic 3-stock universe ($p = 3$) and a corresponding ETF, in which stock prices vary in how quickly they impound news. The stock names A, B and C correspond to the indices $k = 1, 2$ and 3. The ABC ETF price is an equally weighted average of the underlying stocks' efficient prices (4) and the synthetic ABC portfolio is an equally weighted average of the stocks' observed prices (3). The sampling frequency is one minute (i.e., $\Delta_n = 1/391$).

Consider the following time series, where each row in the vectors represents a consecutive time interval, showing a segment of the returns of the three stocks and the ABC ETF:

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

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

[4]Within our theoretical model descriptions, the difference between the price of the synthetic index and the ETF price (5) equals a microstructure noise component minus the discontinuous component that has not yet been impounded into the observed prices:

$$S_{i\Delta_n} - Z_{i\Delta_n} = \sum_{k=1}^{p} w_{k,i\Delta_n} u_{k,i\Delta_n} + \sum_{k=1}^{p} w_{k,i\Delta_n} \left(Y^d_{k,i\Delta_n} - X^d_{k,i\Delta_n} \right). \tag{7}$$

This sharp theoretical decomposition is unobservable to the econometrician in empirical data.

in which the returns are reported in percentages and the jump returns are underlined. The timings of these jumps shows that stock prices asynchronously incorporate news. Stock A's price (first column) takes 2 minutes longer than the ETF's (last column) to complete its jump, Stock B's jump (second column) is delayed by one minute, stock C does not jump at all (third column).

These delays cause the implied (inefficient) returns of the synthetic ABC portfolio to deviate from the (efficient) returns of the ABC ETF. The sum of the first three columns of the matrix below, which calculates the return spread $\delta^r_{i\Delta_n}$, represents the returns of the synthetic ABC portfolio, calculated as the equally weighted, linear combination of the log returns of stocks A, B, and C (the first three columns in equation (8)). 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 equation (9), is the return spread (6), in each of the five periods.

$$\delta^r_{i\Delta_n} = \underbrace{\begin{bmatrix} & & \vdots \\ 1/3.(-0.018+0.015-0.120) & - & (-0.039) \\ 1/3.(-0.031-0.067-0.104) & - & (-0.071) \\ 1/3.(-0.057-0.029+0.088) & - & \underline{0.807} \\ 1/3.(\underline{0.629}+\underline{1.201}+0.017) & - & 0.001 \\ 1/3.(\underline{0.651}+0.062+0.074) & - & 0.073 \\ & & \vdots \end{bmatrix}}_{\text{ABC portfolio returns}} = \underbrace{\begin{bmatrix} \vdots \\ -0.002 \\ 0.003 \\ -0.807 \\ 0.614 \\ 0.189 \\ \vdots \end{bmatrix}}_{\text{Return spreads}}. \tag{9}$$

In the first two periods, the returns of the synthetic index portfolio are almost the same as the returns to the ETF, producing only a small deviation on the right-hand side. Here, only microstructure noise separates the two prices. In the third period, the ETF price jumps by 0.807%, while the prices of the individual stocks do not move much, leading to a large negative spread. In the fourth and fifth periods, the prices of the portfolio of individual stocks catch up to the ETF jump, leading to large positive spreads.

This stylized example captures the central problem in the analysis of common jumps on a fine sampling grid. If news reached the entire market instantly, was interpreted homogeneously, and trading were continuous, jumps in a group of stocks should presumably occur simultaneously with the ETF index jump and the spreads should be small and random. Sluggish price changes lead to asynchronous jumps, however, and the spread temporarily expands and contracts again.

2.2.2 Decomposition of the spread

To isolate the effect of asynchronous jumps on the spread (6), we separate the return of the synthetic stock index portfolio, into discontinuous (jump) and continuous parts. The return of this portfolio, $\Delta^n_i S$, is the weighted average of the observed returns from each individual stock in the portfolio, $\sum_{k=1}^{p} w_{k,i\Delta_n} \Delta^n_i Y_k$, in which $\Delta^n_i Y_k$, for $k = 1, ..., p$, is the Δ^n_i-th observed return of the kth stock.

Jump tests, such as the one developed by Lee and Mykland (2008), help us flag large observed stock returns as jumps. Based on this, we categorize observed stock returns $\Delta^n_i Y_k$ into two distinct categories:

$$\Delta^n_i Y_k = \Delta^n_i J_k + \Delta^n_i C_k. \tag{10}$$

This stock return is composed of a jump return, defined as $\Delta_i^n J_k := \Delta_i^n Y_k \cdot I(\text{Jump}_{i\Delta_n})$, or a continuous return $\Delta_i^n C_k := \Delta_i^n Y_k \cdot I(\text{No Jump}_{i\Delta_n})$. The indicator function $I(\cdot)$ flags whether the jump test flags a jump. These return types are mutually exclusive, meaning a stock return at any given time interval Δ_i^n can be classified as either but not both.

After this decomposition, the return spread (6) equals a linear combination of the weighted stock jump returns, the weighted continuous stock returns, and the ETF returns:

$$\delta^r_{i\Delta_n} \overset{(6)}{:=} \Delta_i^n S - \Delta_i^n Z \overset{(10)}{=} \sum_{k=1}^{p} \underbrace{w_{k,i\Delta_n} \Delta_i^n J_k}_{\substack{\text{Weighted} \\ \text{jump} \\ \text{stock returns}}} + \underbrace{\sum_{k=1}^{p} \underbrace{w_{k,i\Delta_n} \Delta_i^n C_k}_{\substack{\text{Weighted} \\ \text{continuous} \\ \text{stock returns}}} - \underbrace{\Delta_i^n Z}_{\text{ETF returns}}}_{\text{Target}} . \tag{11}$$

We want to optimally change the time labels of the jump returns of individual stocks (the first term) to offset the ETF jump (included in the second term). The "target" consists of the ETF return (with a sign flip) and the continuous stock returns (maintaining their relationship). Recall, if both the stocks and the ETF impound news at the same time, the spread in returns should be small, containing only microstructure noise.

2.2.3 Construction of the jump-event matrix

To help us synchronize jumps within an event window, we create a jump-event matrix Γ. This matrix is an easier-to-handle representation of the decomposition outlined in (11) and has the following structure:

$$\Gamma = (\gamma_{il}) := \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J}_{\substack{\text{Weighted} \\ \text{jump} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n}, \tag{12}$$

The matrix Γ consists of elements γ_{il}, with $i = 1, ..., h$ and $l = 1, ..., q$, where h is the number of time periods in the event window around the ETF jump, and $q-1$, is the number of jumps in individual stocks within the time interval, not the number of stocks. Some stocks may jump more than once or not at all. The event window, $\mathcal{W}_n := [I_1\Delta_n, I_2\Delta_n]$, covers a range of $h := I_2\Delta_n - I_1\Delta_n + 1$ time periods. For the empirical application in Section 3, we use an event window from five minutes before to five minutes after the ETF jump, $I_1\Delta_n = (i^* - 5)\Delta_n$ and $I_2\Delta_n = (i^* + 5)\Delta_n$.

The matrix's first $q-1$ columns consist of vectors of weighted stock jump returns sampled within this event window. In practice, we first collect the p jump vectors of individual stocks within a window of observations:

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

in which $\Delta_i^n J_k$ is the vector of jump returns for stock k, with $k = 1, ..., p$. These vectors are then reorganized in the matrix. If a stock jumps multiple times, as does stock A, separate jumps appear in different columns. Each jump vector contains one and only one non-zero element. We exclude the jump vectors for which the stock does not jump, as in the case of stock C. The matrix's 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 returns $\Delta_i^n Z$.

The row-sums of this matrix show its practical value. Within Γ, we rearrange the stock jumps, but not the elements of the target column. Each row-sum, a linear combinations of stock jumps, the stock's continuous returns and the ETF returns, can easily reflect the spreads for stock jumps with different time labels (11):

$$\Gamma^{+} := \sum_{m=1}^{q} (\gamma_{im}) = [\delta^{r}_{i\Delta_n}]_{i \in \mathcal{W}_n}. \tag{13}$$

In our ABC example, the jump-event matrix (12) is:

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

The first three columns of the jump-event matrix correspond to the three individual stock jumps across an event window from two minutes before to two minutes after the ETF jump. Note that the number of columns in the first block corresponds to the number of identified jumps in individual stocks, not the number of individual stocks. 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 equally weighted index for simplicity. The fourth column in 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 jump stock returns and the target column, *i.e.* the row-sums of the jump-event matrix (13). As before (in 2.2.1), there is a negative and a positive spike in the return spread in a small window around the ETF jump in period 3, because stocks A and B do not jump until periods 4 and 5.

2.3 Rearranging the elements within the jump-event matrix

After decomposing the stock returns into jump and non-jump returns, we can rearrange the stock jumps in the jump-event matrix to offset the target column. The rearrangement is a combinatorial optimization problem. We seek to rearrange the stock jumps in the jump-event matrix (each individual column in the first block of columns in the jump-event matrix) to offset the elements in the target vector (the last column), which minimizes the variability of return spreads (the row-sums).

A rearrangement of a specific column in the $h \times q$ jump-event matrix is determined by a permutation π_l of its h elements, with $l = 1, ..., q$. This permutation π_l is represented compactly as a vector mapping the original order into a new order:

$$\pi_l := \begin{pmatrix} 1 & 2 & ... & h \\ \pi_l(1) & \pi_l(2) & ... & \pi_l(h) \end{pmatrix}. \tag{14}$$

Each column in the jump-event matrix has a permutation. The collection of these permutations for all q columns is denoted by $\pi := (\pi_1, ..., \pi_q)$.

A rearrangement of the jump-event matrix, as defined in (12), is a new matrix, denoted with a π in its superscript:

$$\Gamma := \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J}_{\substack{\text{Weighted} \\ \text{jump} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n} \xrightarrow{\text{Rearrangement}} \Gamma^\pi := \Big[\underbrace{w_{i\Delta_n} \Delta_i^n J^\pi}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{jump} \\ \text{stock returns}}}, \underbrace{T_{i\Delta_n}}_{\text{Target}} \Big]_{i \in \mathcal{W}_n}. \tag{15}$$

In this new matrix, the elements γ_{il}^π correspond to the new configuration of the matrix. The vector $w_{i\Delta_n} \Delta_i^n J^\pi := (w_{1,i\Delta_n} \Delta_i^n J_1^\pi, ..., w_{q-1,i\Delta_n} \Delta_i^n J_{q-1}^\pi)$ consists of the *rearranged* weighted stock jump returns. By changing the order of elements within the jump return vectors, this new version of the matrix reflects a different arrangement of stock jumps. The target column, however, remains unchanged: we do not swap any elements in the qth column in the jump-event matrix, and it does not carry a π in its superscript.

The row-sums of the rearranged jump-event matrix Γ^π are the new return spreads:

$$\Gamma^{\pi,+} := \sum_{m=1}^{q} (\gamma_{im}^\pi). \tag{16}$$

These row-sums are influenced by the specific arrangement (and thus timing) of the stock jumps. This makes them particularly valuable for examining various scenarios and understanding the impact of different timings.

Returning to the ABC example: let's consider a specific permutation π_1 (see (14)), which swaps the 3rd and the 4th observations in the first column in the jump-event matrix:

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

This swap rearranges the jump-event matrix, switching the 3rd and 4th rows of the first column, shifting a weighted stock jump of stock A one period back in time (and a zero forward in time):

$$\Gamma^\pi = \underbrace{\begin{bmatrix} 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 \\ \underline{0.210} & 0.000 & 0.000 \\ 0.000 & 0.000 & \underline{0.400} \\ 0.000 & \underline{0.217} & 0.000 \end{bmatrix}}_{\substack{\text{Rearranged} \\ \text{weighted} \\ \text{jump} \\ \text{stock returns}}} \underbrace{\begin{bmatrix} -0.002 \\ 0.003 \\ -0.807 \\ 0.004 \\ -0.028 \end{bmatrix}}_{\text{Target}}, \text{with row-sums } \Gamma^{\pi,+} = \begin{bmatrix} -0.002 \\ 0.003 \\ -0.597 \\ 0.404 \\ 0.189 \end{bmatrix},$$

After this rearrangement, the third and fourth row-sum of the jump-event matrix also change, and the variability in the row-sums is slightly reduced, because the ETF jump also occurs in the third observation.

2.4 The best rearrangement of the jump-event matrix

Under the assumption that the latent prices of the price series of the stocks and the ETF move in lockstep, the best jump rearrangement minimizes the variability of the return spreads, which measures jump misalignment. This reduction in spread variability is known as "flattening". Formally, we aim to minimize the range of these return spreads, i.e., the row-sums $\Gamma^{\pi,+}$:

$$\min_\pi R\left(\Gamma^{\pi,+}\right). \tag{17}$$

The range $R(\cdot)$ is the difference between the highest and lowest row-sums in our rearranged data:

$$R\left(\Gamma^{\pi,+}\right) = \Gamma^{\pi,+}_{(h)} - \Gamma^{\pi,+}_{(1)}$$

in which the subscript (i) in parentheses indicates the ith order statistic of the sample of row-sums. By minimizing R, the optimization problem finds the best alignment of individual stock jumps with the ETF jump. By doing so, the return spreads will be nearly the same (ideally zero) across all rows, reflecting a coordinated market reaction.

The combinatorial optimization problem (17) is rooted in the pioneering work of Puccetti and Rüschendorf (2012) and Embrechts et al. (2013) on rearrangements and the Rearrangement Algorithm; looping over each column in a matrix to order it oppositely to the sum of the other columns (see Appendix B for a run-through). This algorithm is best known as an actuarial tool to bound portfolio risk, but also has applications in other disciplines, such as such as option pricing (*e.g.*, Bondarenko and Bernard, 2023) and operations research (*e.g.*, Boudt et al., 2018).

Their rearrangement algorithm, however, does not constrain the rearrangements. For example, it can move jumps either forward or backward in time. We introduce the "Rearrangement Mixed-Integer Linear Program" to constrain the procedure from economically implausible rearrangements.

2.4.1 The idea behind the RMILP

In the domain of operations research, one often encounters the challenge of optimizing an objective function within a set of constraints: "Find a vector $\boldsymbol{x}$ that minimizes $\boldsymbol{c}^\top \boldsymbol{x}$, with $\boldsymbol{c}$ a given vector, subject to some constraints on $\boldsymbol{x}$." The Rearrangement Mixed-Integer Linear Program (RMILP) we introduce here adheres to this principle: it chooses arguments which minimize an objective function, subject to linear constraints, and it allows for penalties.

Permutations as permutation matrices

To rearrange (i.e., permute) the h observations in a single column in the jump-event matrix, we use the column representation of a permutation matrix. Given that the jump-event matrix has 4 columns, our solution will involve 4 permutation matrices, one for each column.

The $h \times h$ permutation matrix, denoted as P_{π_l}, reorders the columns of the identity matrix I_h to represent a permutation π_l, as we defined earlier in (14):

$$P_{\pi_l} = (p_{ii'}) = \begin{bmatrix} \boldsymbol{e}_{\pi_l(1)} \\ \boldsymbol{e}_{\pi_l(2)} \\ \vdots \\ \boldsymbol{e}_{\pi_l(h)} \end{bmatrix}. \tag{18}$$

For each i, the element $p_{ii'}$ is 1 if $i' = \pi_l(i)$, and 0 otherwise. This means that the entries of the ith row are all zero except for a 1 that appears in column $\pi_l(i)$. We can think of these rows as basis vectors, where $\boldsymbol{e}_{i'}$ is a row-vector of length h with a 1 on position i' and a 0 on every other position. Note that the arguments in the subscripts of the basis vectors match the permutation order in (14).

This representation is crucial for the RMILP because a permutation matrix (18) can track how far each element shifted by tracking the deviations from the diagonal. Such tracking is essential for imposing penalties for movement. For example, we can impose a maximum

distance from the diagonal to not let the jumps stray too far in the event window (see Section 2.4.3 for further elaboration).

The permutation discussed in the example of Section 2.3, π_1 switches the 3rd and 4th rows of the 1st column in a jump-event matrix, and corresponds to the following 5×5 (there are 5 rows in the jump-event matrix) permutation matrix:

$$P_{\pi_1} = \begin{bmatrix} \boldsymbol{e}_{\pi_1(1)} \\ \boldsymbol{e}_{\pi_1(2)} \\ \boldsymbol{e}_{\pi_1(3)} \\ \boldsymbol{e}_{\pi_1(4)} \\ \boldsymbol{e}_{\pi_1(5)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_4 \\ \boldsymbol{e}_3 \\ \boldsymbol{e}_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{1} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where column i' of the I_5 identity matrix now appears as the column $\pi(i')$ of P_{π_1}. The changes occur in the vertical, i, dimension: upward moves in the permutation matrix are backward moves in time, while downward moves in the permutation matrix are forward moves in time. The fourth element, which is on position $(4,4)$ in I_5, shifts one spot backward in time 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 in time by shifting one step downward in the permutation matrix. In other words, P_{pi_1} switches the period 3 return with the period 4 return.

There is a distinct permutation matrix for each of the q columns in the jump-event matrix, i.e., for each stock jump and the target. These matrices are then concatenated to form the co-permutation matrix Π of dimension $h \times (hq)$:

$$\Pi = (p_{lii'}) = \left[P_{\pi_1}, P_{\pi_2}, ..., P_{\pi_q}\right], \tag{19}$$

with $l = 1, ..., q$, $i, i' = 1, ..., h$ and P_{π_1} short notation for the $h \times h$ permutation matrix for the first column in the jump-event matrix. Similarly, $P_{\pi_1}, P_{\pi_1}, ..., P_{\pi_q}$ are the permutation matrices for the 2nd, 3rd, ..., and qth columns, respectively.

Extracting the row-sums with permutation matrices

To produce the return spreads, we start by transforming the jump-event matrix into a single long column vector and then premultipy the (vectorized) jump-event matrix by the co-permutation matrix:

$$\Gamma^{\pi,+} = \Pi \times \text{vec}(\Gamma), \tag{20}$$

or, equivalently:

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

in which vec(Γ), the vectorized version of the observed jump-event matrix $\Gamma = (\gamma_{il})$, with $i = 1, ..., h$ and $l = 1, ..., q$, is a stacked column vector of dimension $hq \times 1$. The co-permutation matrix Π rearranges this vector. The result of the matrix product in (20) is a column vector with h entries (the left side of the equal sign). Each entry is a row-sum from the rearranged jump-event matrix.

Flattening the row-sums by choosing the best permutation matrices

Recall that linear programs are problems that can be expressed in canonical form as: "Find a vector $\boldsymbol{x}$ that minimizes $\boldsymbol{c}^\top \boldsymbol{x}$, with $\boldsymbol{c}$ a given vector, subject to some constraints on $\boldsymbol{x}$." Because we cannot directly express the objective function, R, as a function of the permutation matrices in Π, the RMILP steps slightly outside the bounds of conventional linear programming to get the appropriate form indirectly.[5] The RMILP aims to find the co-permutation matrix Π, along with two auxiliary boundary values, L (lower) and U (upper), to minimize the interval $U - L$:

$$\begin{aligned} &\text{Find } \Pi, L, U \text{ that minimizes } U - L, \\ &\qquad \text{subject to } L \leq J^{\pi,+}_{n,i} \leq U, \text{ for } i = 1, ..., h, \end{aligned} \tag{21}$$

in which Π is the co-permutation matrix, which defines the rearranged jump-event matrix, and L and U define lower and upper bounds of an unknown interval.

This objective function alone does not directly minimize the range of the row-sums. We link L and U with Π by constraining the decision variables with inequality constraints: L and U mark the bounds within which all row-sums $J^{\pi,+}_{n,i}$ fall. Seemingly trivial, this constraint indirectly sets the smallest possible (the minimum) and largest possible (maximum) row-sum.[6] By minimizing $U - L$, subject to the inequality constraint, the RMILP squeezes these boundaries together as closely as possible. In doing so, the RMILP ensures that the optimal solutions for L and U (denoted L^* and U^*) equal the smallest and largest row-sums:

$$L^* = J^{\pi,+}_{n,(1)} \text{ and } U^* = J^{\pi,+}_{n,(h)}, \tag{22}$$

A simple proof by contradiction suffices to confirm this statement: if L^* and U^* were not equal to the smallest and largest row-sums, one could still further minimize $U - L$.

The program in (21) chooses candidates for three decision variables – Π, L, and U– to minimize the range of the row-sums. The optimization problem thus becomes a "mixed-integer" linear program, with Π composed of binary elements $(0, 1)$ and continuous values for L and U.

2.4.2 Other necessary constraints for the RMILP

In the RMILP, we constrain the choices of the co-permutation matrix (19) using two additional types of constraints: the permutation constraint and a target constraint. The permutation constraint requires that each permutation matrix is composed entirely of zeros and ones, and that each row and column in the permutation matrix sums to one. An identity matrix satisfies this constraint, for example, as does an identity matrix with its rows rearranged in any order. The target constraint prohibits changes in the order of the elements in the target column and must be explictly encoded into the program. The combination of these minimal conditions produces meaningful rearrangements that effectively minimize the range.

---

[5] While the matrix product (20) can easily extract each individual row-sum, i.e., $J^{\pi,+}_{n,1}, J^{\pi,+}_{n,2}, ..., J^{\pi,+}_{n,h}$ (without brackets in the indices), there is no possible choice of the co-permutation matrix Π which produces the maximum or minimum of the row-sums specified in our objective function, i.e., $J^{\pi,+}_{n,(1)}$ and $J^{\pi,+}_{n,(h)}$ (with brackets) through the matrix product.

[6] The row-sums $J^{\pi,+}_{n,i}$ in the middle of the constraint result from the matrix product in (20) for a given arrangement of jumps. The left inequality in the constraint of (21) defines a lower boundary in a set of row-sums (by definition each individual row-sum should greater than or equal to the minimum) and the right inequality defines upper boundary in a set of row-sums (by definition each individual row-sum should be less than or equal to the maximum). The outer boundaries, the lower boundary should be less than or equal to the upper boundary, ensures that the minimum is always smaller than the maximum, as by definition.

The permutation constraints

To define a proper co-permutation matrix (19), we set up equality constraints in the linear program:

$$\text{subject to } \sum_{l=1}^{q}\sum_{i=1}^{h}\sum_{i'=1}^{h} p_{lii'} = hq, \tag{23}$$

$$\sum_{i'=1}^{h} p_{lii'} = 1, \text{ for } i = 1, ..., h \text{ and } l = 1, ..., q, \tag{24}$$

$$\sum_{i=1}^{h} p_{lii'} = 1, \text{ for } l = 1, ..., q \text{ and } i' = 1, ..., h. \tag{25}$$

Constraint (23) requires that each permutation matrix (18) contains h ones, corresponding to selecting all (exactly h) elements in each column in the jump-event matrix. Thus, the total sum across all q permutation matrices in the co-permutation matrix equals hq. Constraint (24) imposes that the rows of each permutation matrix to sum to one. Without this, we could end up selecting multiple or no elements in certain positions of the rearranged matrix. Constraint (25) applies to the columns of the permutation matrix. It guarantees that no number is repeated within any column in the rearranged matrix.

The target

We also impose an equality constraint on the permutation matrix of the last column in the jump-event matrix P_{π_q} that ensures that the RMILP does not rearrange the elements in the target column:

$$\text{subject to } (\text{diag}(P_{\pi_q}))_i = 1, \text{ for } i = 1, ..., h, \tag{26}$$

In other words, the permutation matrix (18) associated with the qth column in the jump-event matrix P_{π_q} must be an identify matrix.

2.4.3 Adapting the RMILP with additional constraints

Linear programming allows us to prevent economically implausible rearrangements. For example, we can penalize large moves or moves in a particular direction with the aid of a distance matrix. This matrix, D_n, tracks how far each rearrangement in any permutation matrix P_{π_l} (18) deviates from its original order (an identity matrix). D_n, an $h \times h$ matrix, is structured as:

$$D_n = (d_{ii'}) \begin{bmatrix} 0 & 1 & \cdots & h-2 & h-1 \\ 1 & 0 & \cdots & h-3 & h-2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h-2 & h-3 & \cdots & 0 & 1 \\ h-1 & h-2 & \cdots & 1 & 0 \end{bmatrix}, \tag{27}$$

with zeros on the diagonal and increasing values as we move away from the diagonal.

We only allow stock jumps to be moved backward in time because we assume that stock prices are sluggish and lag the highly liquid and carefully watched ETF, they do not lead it. We can restrict stock jumps to earlier times only, using the upper triangular portion of

D_n to monitor backward shifts in P_{π_l}. The upper triangular portion of the distance matrix tracks the backward distance traveled of *all* elements in a column in the jump-event matrix, because the permutation matrix (18) also tracks the rearrangements of the zeros. To only track the move of a stock jump, we disable the irrelevant columns of the distance matrix: if the price jumps in period $i = i^*$, we set the columns $i' \neq i^*$ of D_n to zero.

The distance of the move for the lth column in the jump-event matrix is equal to the following matrix product of a vectorized permutation matrix and a vectorized distance matrix:

$$d(P_{\pi_l}) = \text{vecr}\left(P_{\pi_l}\right) \times \text{vecr}\left(D_n\right)^\top, \text{ for } l = 1, ..., q. \tag{28}$$

The vectorization, $\text{vecr}(\cdot)$ concatenates the rows of a matrix (unlike standard vectorization, which stacks columns), producing a $1 \times hq$ row-vector. We transpose the second term after vectorization to get a $hq \times 1$ column-vector. The matrix product gives us the total number of relevant shifts.

We limit the length of jump moves in our rearrangements with an inequality constraint:

$$\text{subject to } d(P_{\pi_l}) \leq c, \text{ for } l = 1, ..., q-1, \tag{29}$$

in which $c \geq 0$ is the maximum permitted length of the backward move and $d(P_{\pi_l})$ is the total distance traveled by a jump within the lth column in the jump-event matrix. For example, we could allow each jump to move backward in time a maximum of one or two minutes and solve the RMILP for each of these possible jump-length constraints. The qth column is exempt from this constraint, as the target column remains fixed per constraint (26). In practical applications, we could even tailor constraints for each stock based on its liquidity.

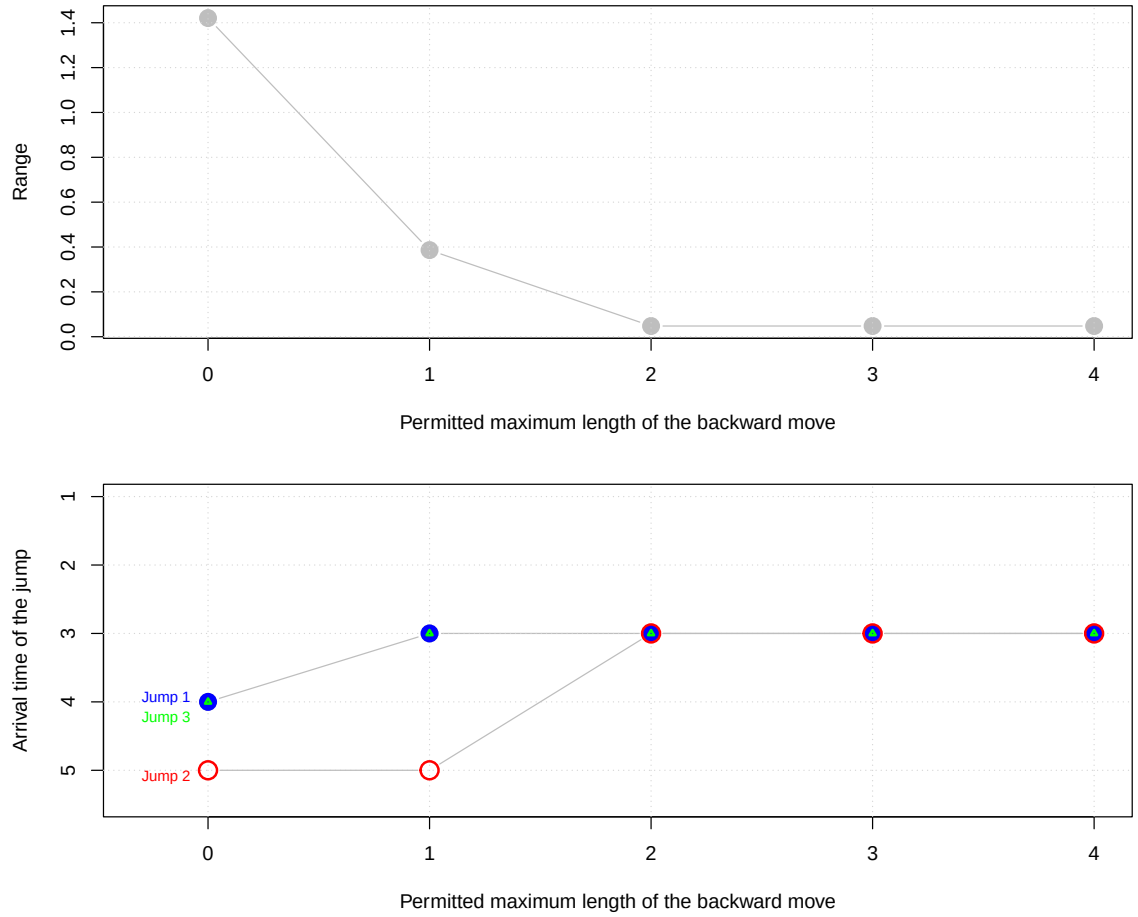
The trace plot in Figure 3 illustrates the minimized criterion function, i.e., the range of the spreads or flatness, and the jump arrival times as a function of the permitted maximum length of the move – a maximum of zero, one, two, three of four minutes – for each jump in the stylistic jump-event matrix. We start from the observed prices: stock A jumps with a delay and gradually in the 4th and 5th periods (Jumps 1 and 2) and stock B jumps with a delay in the 4th period (Jump 3). The negative slope of the line in the top panel of Figure 3 shows that larger permitted backward shifts flattens the return spreads.

The bottom panel shows the rearranged arrival periods for the jumps as a function of the permitted length of backward moves. With no backward moves, jumps 1 and 3 arrive in period 4 and jump 3 arrives in period 5. If we permit one backward move for each jump, the RMILP moves jumps 1 and 3 to period 3 while if we permit 2 backward moves, the RMILP moves all jumps to period 3. The smallest range (0.048) occurs with two backward shifts, aligning all the jumps in the third period in the bottom panel. Optimally rearranging the jumps, produces the following transformation of the jump-event matrix:

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

Figure 4 shows the implied prices after this optimal arrangement of jump returns. The observed (black) prices of stock A and stock B deviate from their efficient (gray) values in the first and second panels. (There are also small deviations in Stock C's observed prices

Figure 3: Trace plot: Range of the spreads and jump arrival times as a function of the permitted maximum length of the move



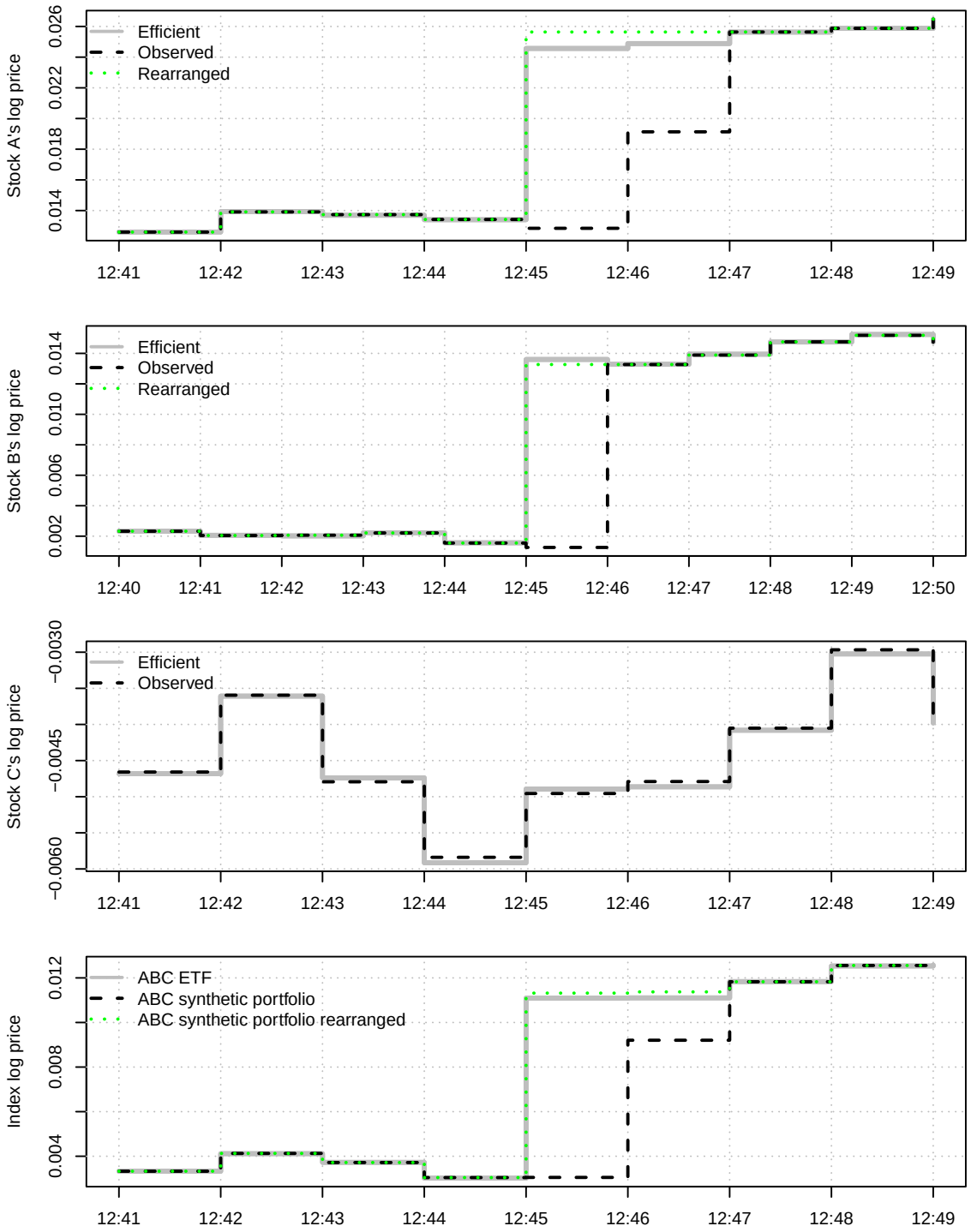
Note: This figure plots the range of the return spreads (the row-sums of the jump-event matrix) and the implied jump arrival times as a function of the length of permitted backward moves of the jumps in each of the columns. The RMILP rearranges the gradual jump of Stock A (Jump 1 in the 4th period and Jump 2 in the 5th period) and the delayed jump of Stock B (Jump 3 in the 4th period). We allow each jump to move backward in time a maximum of zero, one, two, three, four minutes and solve the RMILP for each of these five possible constraints.

due to a (relatively smaller) contamination of the continuous component.) The fourth panel shows that the delays cause the implied (inefficient) price of the basket of stocks (the ABC portfolio) to deviate from the efficient ABC ETF price. The best rearrangement combines two small jumps of stock A to period 3 and shifts the jumps of stock B one period backward to period 3, aligning the jumps in time. The rearranged price paths (green) are now much closer to the efficient price paths (black) in the 1st, 2nd and 4th panel.

3 Empirics

We apply our methods to investigate the reactions of stock prices in event windows around ETF jumps. We illustrate that synchronizing mistimed stock returns increases the Sharpe ratio of a portfolio allocation strategy.

Figure 4: The best rearrangement approximately recovers the efficient stock jumps



Note: This figure shows the efficient (gray), observed (black) and rearranged (green) log prices of the assets. The top panel shows that Stock A jumps gradually and completes the jump 2 minutes after the ETF in bottom panel, while the second panel shows that stock B's jump is 1 minute late and stock C does not jump at all (third panel). The fourth panel shows that the delays cause the implied (inefficient) price of the basket of stocks (the ABC portfolio) to deviate from the efficient ABC ETF price. The best rearrangement approximately recovers the latent, efficient stock price. That is, by shifting the jumps of Stock A and Stock B to earlier periods (top 2 panels), the LP minimizes the distance between the price of the ETF and the synthetic portfolio (lowest panel).

3.1 Data: The Dow and the DIA ETF

The NYSE Trade and Quote (TAQ) database provides equity trade data with millisecond precision timestamps. The basket instrument is the SPDR Dow Jones Industrial Average ETF (DIA). We compare this ETF to the price of a synthetic index of Dow 30 stock prices, as in Bollerslev et al. (2008).[7]

The data span January 3, 2007 through April 2, 2020, which includes several exceptionally turbulent episodes, such as the global housing and credit crisis, the European sovereign debt crisis and the bail-out of Greece, the Russian, Greek, Turkish crisis and the 2020 stock market crash.

We pre-filter the prices as in Barndorff-Nielsen et al. (2009). We also remove banking holidays, half-trading days, any day where there is more than a two-hour gap between consecutive trades and periods of malfunctioning such as the 2010 flash crash.

There are many jumps in the data. We apply Boudt et al. (2011b)'s modified Lee and Mykland (2008)'s univariate jump test that accounts for intraday periodicity, on one-minute returns (at $\alpha = 0.1\%$) to identify jumps. The univariate tests identify 1,710 ETF jumps across 1,163 (jump) days. Some days include gradual or multiple ETF jumps.

We construct 1,529 [-5,+5]-minute jump-event matrices around ETF jumps, as in equation (12). The event window spans from five minutes before the first ETF jump to five minutes after the last jump in the case of multiple jumps. We contrain the RMILP in three ways. 1) We do not rearrange the jumps of stocks that already jump with the index, because we assume those are efficient. 2) No stock jumps may be moved earlier than the highly liquid and carefully watched ETF. 3) No stock jumps may be moved if the ETF jumps within the first 10 minutes or the last 10 minutes of the trading day. After imposing these filters, there remain 380 matrices candidate for rearrangement. The RMILP rearranges stock jumps in 180 cases (or 11.8% of all jump-event matrices). That is, there are many asynchronous jumps.

3.2 News and asynchronous jumps: An empirical illustration

We investigate sluggish cojumps, *i.e.* jumps that occur later than the index jumps in the DIA. Consider the following example: on September 18, 2007 the Federal Reserve announced rate cuts, which the financial press characterized as a bold action.[8]

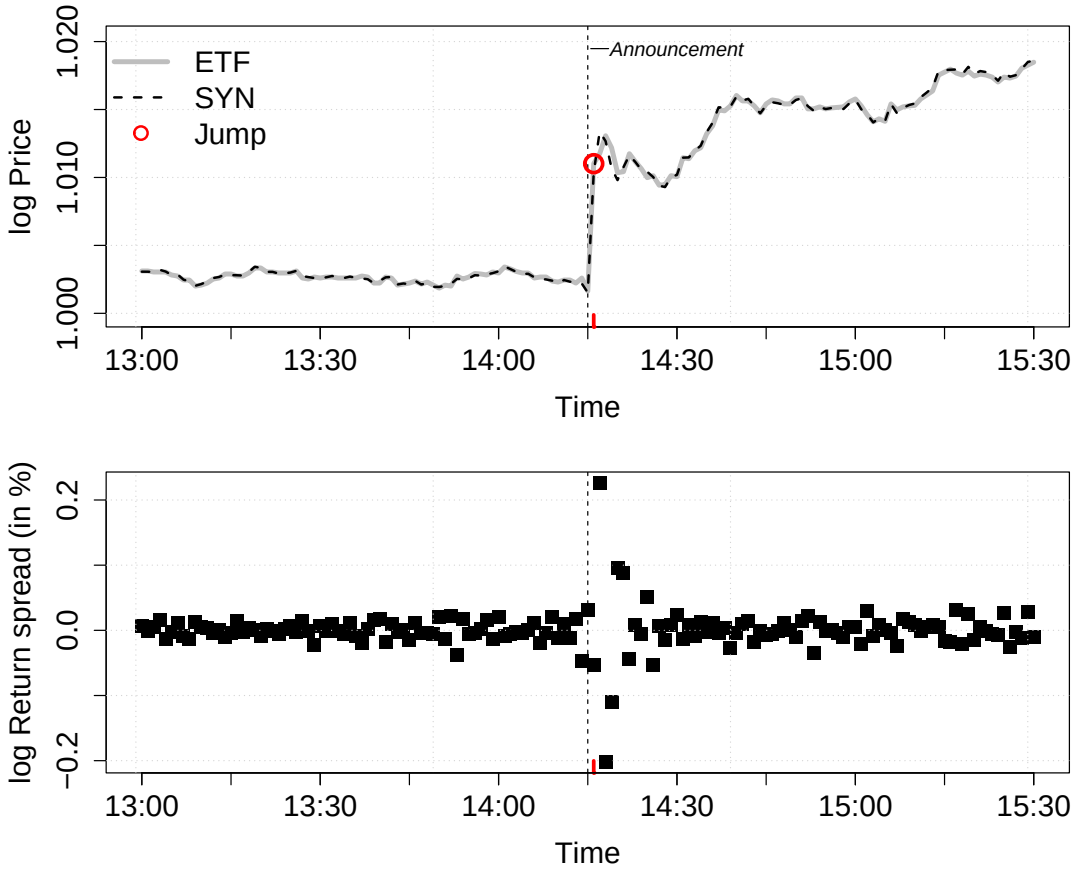
Figure 5 shows the price paths of the DIA ETF and a synthetic Dow 30 index and the spread between the returns of those two assets (6) on the day of the FOMC Statement – September 27, 2008. Lee and Mykland (2008)'s jump test flags a 0.938% ETF price jump at 2:16 pm, one minute after the release of the statement. We mark the ETF jump with a red circle in the upper panel. As noted before, if news reached the entire market instantly, was interpreted homogeneously, and trading were continuous, jumps in a group of stocks should presumably occur simultaneously with the ETF index jump and spreads should be small and

[7]The constituency of the Dow 30 index is dynamic. The 43 unique tickers within our sample period 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.

[8]For more information on this event, we refer to the official Press release and the related FOMC Meeting Statement. Comprehensive coverage and analysis of this decision were provided by major financial news outlets: see e.g., 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

random. The bottom panel shows that the ETF-synthetic index spread temporarily expands, however, following the ETF jump, and then contracts again. That is, markets took time to incorporate the Fed's news into the Dow 30 stock's prices.

Figure 5: When the FOMC statement is released on Sep. 18, 2007, the synthetic index–ETF spread expands and contracts again

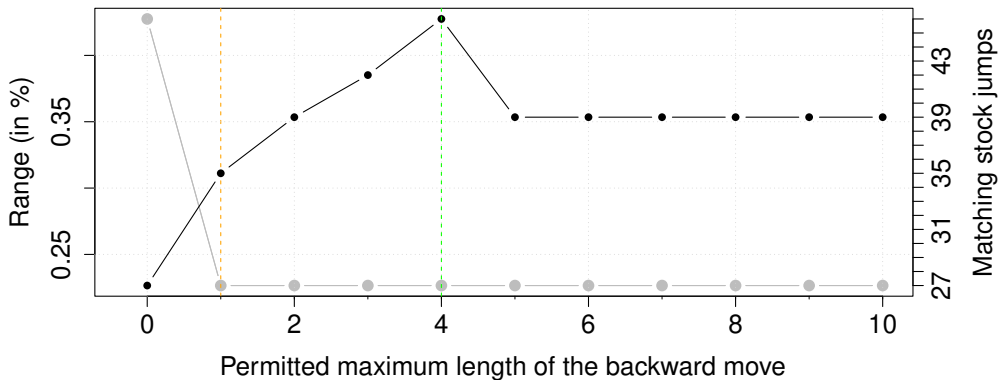


Note: We plot the one-minute log prices of the DIA ETF (in gray) and a synthetically constructed Dow 30 index (in black), alongside the log return spreads, on the day of the FOMC Statement of September 18, 2007. The statement gets issued at 2:15 pm and is marked in the top plot by a dashed line. The subsequent ETF jump at 2:16 pm is marked with a red circle and a red vertical tick on the time axis.

A jump-event matrix (12) characterizes the asset jumps across an event window from five minutes before to five minutes after the index jump. On Sep. 18, 2007, for example, the event window contains 50 stock jumps, including many gradual jumps, from 30 stocks, of which 27 match, *i.e.* occur at the same time as, the ETF jump and 23 lag the ETF jump.

Figure 6 shows the range of the return spreads (left scale, gray) and the number of matched stocks (right scale, black) as a function of the permitted length of the move in time for each individual jump, as in equation (29). Permitting each jump to move one minute backward minimizes the range (orange line); the range drops from 0.427% to 0.227%. Permitting a maximum backward move of 4 periods or 4 minutes (green line) produces the same minimal range for a larger number of matching stocks. This rearrangement matches 19 out of 23 scattered stock jumps with the ETF jump, approximately recovering the common jump in the stocks.

Figure 6: Range of the spreads and matching stocks as a function of the permitted maximum length of the move



Note: This figure plots the range of the return spreads (left scale, gray) and the number of matching stocks (right scale, black) as a function of the number of the permitted length of the move during the Sep. 18, 2007 FOMC statement. The RMILP rearranges scattered jumps of the Dow 30 stocks. We allow each jump to move at most 10 minutes backward in time and solve the RMILP for each of these possible constraints. We mark the rearrangement with minimal range with an orange line and the rearrangement with the maximum number of matching stock jumps (for the same minimal range) with a green line.

Recovering the common jump is likely to improve estimates of the daily realized covariance matrix. The realized covariance matrix is defined as (Barndorff-Nielsen and Shephard, 2004):

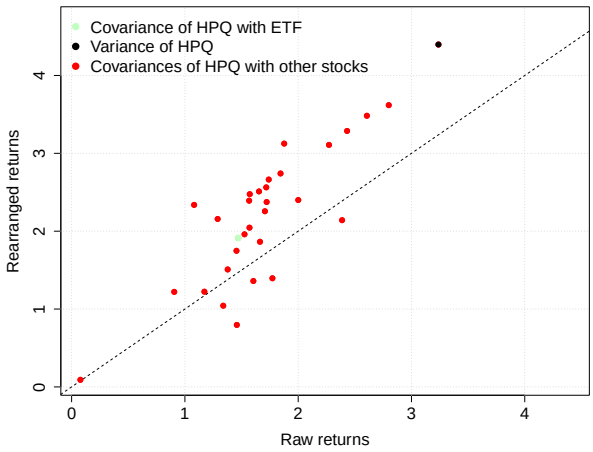
$$C_d = \sum_{i=1}^{\lfloor T/\Delta_n \rfloor} (\Delta_i^n Y)(\Delta_i^n Y)^\top \tag{30}$$

in which, $Y_{i\Delta_n} = (Y_{1,\Delta_n}, ..., Y_{p,\Delta_n})^\top$ is the observed log price process (2) of the p stocks sampled on a regular time grid $\{i\Delta_n : 0 \leq i \leq \lfloor T/\Delta_n \rfloor\}$ over one day $T = 1$, the ith return of $Y_{i\Delta_n}$ is $\Delta_i^n Y = Y_{i\Delta_n} - Y_{(i-1)\Delta_n}$. The standard realized covariance matrix (30) plugs in raw returns. Other estimators, like the multivariate realized kernel in Barndorff-Nielsen et al. (2011) or a Cholesky factorization in Boudt et al. (2017), protect against mild market microstructure noise and the Epps (1979) effect, that is, the downward bias in covariance estimates due to asynchronous trading. Using rearranged returns in (30) also protects against asynchronous jumps and the underestimation of jump dependence.

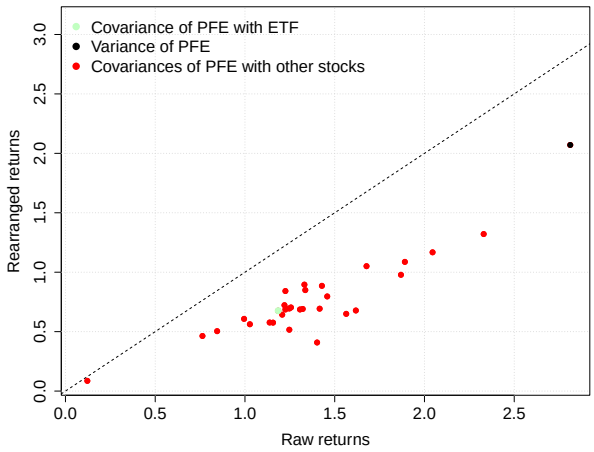
Figure 7 shows a scatterplot of the estimated covariances of two stocks, HPQ and PFE, versus the ETF and the 29 other Dow 30 stocks, when the covariances are constructed with raw returns and with rearranged returns. The scattering of red dots away from (1,1) illustrates that optimally changing the time labels of one of two observations (out of the 390 one-minute returns required to estimate the realized covariance) changes the estimated covariance structure between the stocks and the ETF returns on the day of the FOMC statement. For example, the fact that the statistics in the left panel generally lay above the 45-degree line shows that rearranging the jumps of HPQ, which jumps gradually, increases its variance and most covariances with other stocks. The right panel shows that rearranging the jumps of PFE, which jumps with an overreaction, reduces its own variance and all the covariances with PFE returns.

Figure 7: The realized covariance is different in the rearranged returns on the day of the FOMC statement

(a) Realized covariances of HPQ: Sep. 18, 2007



(b) Realized covariances of PFE: Sep. 18, 2007



Note: This figure shows two scatter plots consisting of the elements in the realized covariance matrix for two stocks (HPQ and PFE), relative to DIA and the other Dow 30 stocks, on the day of the FOMC statement. The x-axis shows the realized covariances with raw returns and the y-axis the realized covariances with rearranged returns.

3.3 Minimum-variance portfolios

We show the usefulness of using rearranged returns, compared to using raw returns, in the context of an exercise to rebalance a portfolio to minimize daily variance.

Stock returns determine the weights that minimize the portfolio variance. The optimal weights for a particular day minimize the portfolio variance, subject to the constraints that they deliver a given expected return and that they sum to one (Markowitz, 1952):

$$\underset{w_d}{\text{Minimize}} \ \sigma^2_{p,d} = w_d' C_d w_d, \ \text{ s.t. } \ w_d' \mu = \mu_{p,d} \text{ and } w_d' 1, \tag{31}$$

in which $\sigma^2_{p,d}$ is the daily variance of the portfolio return, $w_d = (w_{1,d}, ..., w_{p,d})^\top$ is the daily p-dimensional weight vector, C_d is the daily realized covariance matrix and $\mu_{p,d}$ is the target portfolio return. We make a grid of 100 target returns that range from the lowest average DJIA stock return to the highest one. To test our synchronization procedure, we compare a simulated portfolio's performance using raw vs. rearranged returns to compute the realized covariance matrix.

We optimize weights (31) on rearrangement days, *i.e.* the 184 ETF jump days for which we rearrange jumps and use those weights to rebalance the portfolio the next day. We keep those weights until the next rearrangement day. In those cases when the Dow constituency changes in between rearrangement days, we reset to an equally weighted portfolio until the next rearrangement day.

Table 1 reports the portfolios' closing value, standard deviation and modified Sharpe ratio at the 5% level and the p-value of their difference (Ardia and Boudt, 2015) for the full sample and for each year. For 12 of 14 years and over the whole sample, the rearranged-return portfolio statistics are superior to the raw-returns portfolio statistics in terms of the modified Sharpe ratio. Over the full sample, the modified Sharpe ratios are significantly different and the rearranged-return portfolio delivers an additional 5% performance.

Table 1: Performance of minimum-variance portfolios

	Days	#CJ	#CJ-R	Closing value		SD		mSharpe		
				Raw	Rearr.	Raw	Rearr.	Raw	Rearr.	p-value
Full period	3,264	1,163	184	1.8500	1.9000	0.7410	0.7410	0.0159	0.0178	0.0879
Year-by-year										
2007	246	57	13	1.0191	1.0294	0.5592	0.5652	0.0357	0.0430	0.2520
2008	247	55	6	0.9900	0.9880	1.5156	1.5043	0.0377	0.0378	0.9970
2009	243	68	11	1.0607	1.0684	0.9220	0.9254	-0.0127	-0.0110	0.2700
2010	247	65	5	1.0213	1.0384	0.6292	0.6324	-0.0373	-0.0295	0.0919
2011	247	72	6	1.1505	1.1537	0.6268	0.6219	0.0628	0.0644	0.5500
2012	245	107	8	0.9439	0.9493	0.5045	0.5056	-0.0634	-0.0609	0.2950
2013	245	90	13	1.1049	1.0965	0.4865	0.4864	-0.0034	-0.0101	0.1130
2014	247	78	15	1.2607	1.2510	0.4790	0.4730	0.2310	0.2260	0.5180
2015	247	72	9	1.0135	1.0211	0.6918	0.6843	-0.0061	-0.0006	0.4530
2016	249	101	14	1.1480	1.1482	0.5113	0.5109	0.0118	0.0119	0.9860
2017	247	133	23	1.1220	1.1234	0.2977	0.2986	0.0692	0.0709	0.6880
2018	245	93	14	0.9231	0.9310	0.8102	0.8278	0.0254	0.0302	0.4400
2019	246	140	42	1.0817	1.0921	0.5076	0.5123	-0.0028	0.0040	0.2380
2020	63	32	5	1.0135	1.0140	1.5388	1.5407	-0.0021	-0.0017	0.8820

Note: This table reports the performance of the daily returns of a minimum-variance portfolio optimization with raw *vs.* rearranged returns. From left to right, the columns report the number of days (Days), the number of ETF jumps or cojump days (#CJ) and the number of cojump days on which the rearrangement moves stock jumps (#CJ-R). "Closing value" is the end-of-sample portfolio value if the initial value was \$1 using cumulative daily returns. "SD" and "mSharpe" are the standard deviations and modified Sharpe ratios and the p-value of their difference (Ardia and Boudt, 2015) of the portfolio over the sample. Gray cells denote the higher of the two performance statistics. Dark gray cells denote statistically significant superiority at the 10% level. To calculate the modified Sharpe ratio, we use excess returns *vs.* the DIA ETF. We rely on the studentized circular bootstrap approach, with 1000 bootstrap replications and a block length of 5, to infer the statistical significance of differences in Sharpe ratios.

We recommend using rearranged returns rather than raw returns. These synchronized returns are more precise in a high-frequency analysis of market reactions around jumps.

4 Concluding remarks

In a panel of intraday stock returns, sampled on a fine sampling grid, jumps occur at close but distinct points in time. We introduce tools to synchronize the scattered jumps and better approximate the common jump. The rearrangement is currently very practical, requiring only a few seconds for a rearrangement problem with 30 stocks.

Estimating realized covariance matrices with synchronized stock returns, as opposed to raw returns, improves out-of-sample portfolio performance. Recovering the common jump on a fine sampling grid is likely to improve other asset allocation and risk management decisions, like estimating the jump size distribution (see *e.g.*, Boudt et al., 2011a), estimating jump dependence (see *e.g.*, Li, Todorov, Tauchen, and Chen, 2017; Li, Todorov, and Tauchen, 2017; Li, Todorov, Tauchen, and Lin, 2019) or forecasting realized measures (see *e.g.*, Andersen et al., 2007; Bollerslev et al. 2020; Bollerslev et al., 2022). A more thorough analysis must, however, await future work.

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., T. Bollerslev, and F. X. Diebold (2007). Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility. *Review of Economics and Statistics 89*(4), 701–720.

Ardia, D. and K. Boudt (2015). Testing equality of modified sharpe ratios. *Finance Research Letters 13*, 97–104.

Bajgrowicz, P., O. Scaillet, and A. Treccani (2016). Jumps in high-frequency data: Spurious detections, dynamics, and news. *Management Science 62*(8), 2198–2217.

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.

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2011). Multivariate realised kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. *Journal of Econometrics 162*(2), 149–169.

Barndorff-Nielsen, O. E. and N. Shephard (2004). Econometric analysis of realized covariation: High frequency based covariance, regression, and correlation in financial economics. *Econometrica 72*(3), 885–925.

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., J. Li, A. J. Patton, and R. Quaedvlieg (2020). Realized semicovariances. *Econometrica 88*(4), 1515–1551.

Bollerslev, T., A. J. Patton, and R. Quaedvlieg (2022). Realized semibetas: Disentangling "good" and "bad" downside risks. *Journal of Financial Economics 144*(1), 227–246.

Bondarenko, O. and C. Bernard (2023). Option-implied dependence and correlation risk premium. *Journal of Financial and Quantitative Analysis*, 1–51.

Boudt, K., C. Croux, and S. Laurent (2011a). Outlyingness weighted covariation. *Journal of Financial Econometrics 9*(4), 657–684.

Boudt, K., C. Croux, and S. Laurent (2011b). Robust estimation of intraweek periodicity in volatility and jump detection. *Journal of Empirical Finance 18*(2), 353–367.

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., S. Laurent, A. Lunde, R. Quaedvlieg, and O. Sauri (2017). Positive semidefinite integrated covariance estimation, factorizations and asynchronicity. *Journal of Econometrics 196*(2), 347–367.

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.

Christensen, K., A. Timmermann, and B. Veliyev (2023). Warp speed price moves: Jumps after earnings announcements. *Available at SSRN 4422376*.

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.

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, and G. Tauchen (2017). Jump regressions. *Econometrica 85*(1), 173–195.

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.

Li, Z. M. and O. Linton (2022). A ReMeDI for microstructure noise. *Econometrica 90*(1), 367–389.

Markowitz, H. (1952). Portfolio selection*. *Journal of Finance 7*(1), 77–91.

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.

A Simulating sluggish news reactions

We show how to generate a sample path from the new DGP, in which the discontinuous component is spread across several time intervals. We first simulate second-by-second ($\Delta_n = 1/23,401$) efficient log prices (1) for 1 stock ($p = 1$), $X_{i\Delta_n}$, across one trading day ($T = 1$) from a jump-diffusion process.

A.1 The continuous component

The continuous component, $X^c_{i\Delta_n}$, has a zero drift and constant variance $\sigma^2_t = 0.039$, corresponding to an annualized return volatility of 20%. 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 $\gamma = \left(n\omega^2/\sigma^2_t\right)^{1/2} = 0.5$, to contaminate the continuous part of the efficient price (as in Christensen et al., 2014).

A.2 The discontinuous component

Econometricians typically assume a compound Poisson for the efficient jump process:

$$X^d_{i\Delta_n} := \sum_{j=1}^{N^J} I_{\{U_j \cdot T \leq i\Delta_n\}} \Delta X_j, \tag{32}$$

in which $I(\cdot)$ is an indicator function, N^J is the number of jumps that occur during 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 normally distributed jump sizes.

The news that 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, $Y^d_{i\Delta_n}$, which includes a step function that spreads each individual efficient jump size, ΔX_j, within the compound Poisson process (32), across several time intervals:

$$Y^d_{i\Delta_n} := \sum_{j=1}^{N^J} I_{\{U_j \cdot T \leq i\Delta_n\}} \underbrace{\sum_{d=0}^{N^D_j} I_{\{W_{j,d} \cdot T \leq i\Delta_n\}} \Delta L_{j,d}}_{\text{Progress to efficiency}} \Delta X_j, \tag{33}$$

in which N^D_j governs the total number of steps within each jump's delay process, $W_{j,0}, ..., W_{j,N^D_j}$ are the step arrival times and $\Delta L_{j,0}, ..., \Delta L_{j,N^D_j}$ are increments in the step sizes, which add chunks of the efficient jump size, ΔX_j, to the observed jump process, Y^d. The step function captures the progress to effiency; it rises from 0 to 1 as information about the efficient jump is fully incorporated in the stock's observed price.

A.2.1 Step widths

Suppose an efficient price jumps at the random arrival time, U_j, with a jump size, ΔX_j. To simulate the step function within for this particular jump, we first draw the number of steps, N^D_j, from a binomial distribution:

$$N^D_j \sim \text{Bin}(\text{Number of trials} = 5,\ \text{Success probability} = 0.4),\ \text{for}\ j\ \text{fixed}.$$

Each of the steps has a random width. An exponential distribution governs the waiting times between the steps:

$$w_{j,d} \sim \lceil \text{Exp}[\text{Rate} = 1/(15\, N_j^D)] \rceil, \text{ for } j \text{ fixed and } d = 1, ..., N_j^D,$$

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:

$$\begin{aligned} W_{j,0} \cdot T &:= U_j \cdot T \quad , \\ W_{j,1} \cdot T &:= U_j \cdot T \quad + w_{j,1}, \\ W_{j,1} \cdot T &:= U_j \cdot T \quad + w_{j,1} + w_{j,2}, \\ &\vdots \\ W_{j,N_j^D} \cdot T &:= U_j \cdot T \quad + D_j. \end{aligned}$$

The starting point of the step function, $W_{j,0}$, is the arrival time of the efficient jump. The end point, W_{j,N_j^D}, or the moment when the information has been fully impounded, is the starting point plus the sum of the waiting times. 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 efficient jump, the total delay will be longer.

A.2.2 Step sizes

The steps also have random sizes 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

The Brownian bridge models the latent news impoundment process. When news is released, the market processes and accumulates information, including under- and overreactions. Eventually, the price fully and correctly impounds the information.

The Brownian bridge, $\Lambda_{j,t}$, is a continuous-time, stochastic process defined as:

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

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 interval, at $t = U_j \cdot T$ and $t = U_j \cdot T + D_j$. 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 $\Lambda_{j,U_j \cdot T + D_j} = 1$.

Sampling step sizes from the Brownian bridge

The variable governed by the Brownian bridge gradually moves from 0 to 1 – non-monotonically – but we observe its values at discrete intervals, because investors impound chunks of new information in the observed price at each interval. We sample the step sizes from the Brownian bridge process $\Lambda_{j,t}$ at discrete points, $W_{j,0}, W_{j,1}, ..., W_{j,N_j^D}$.

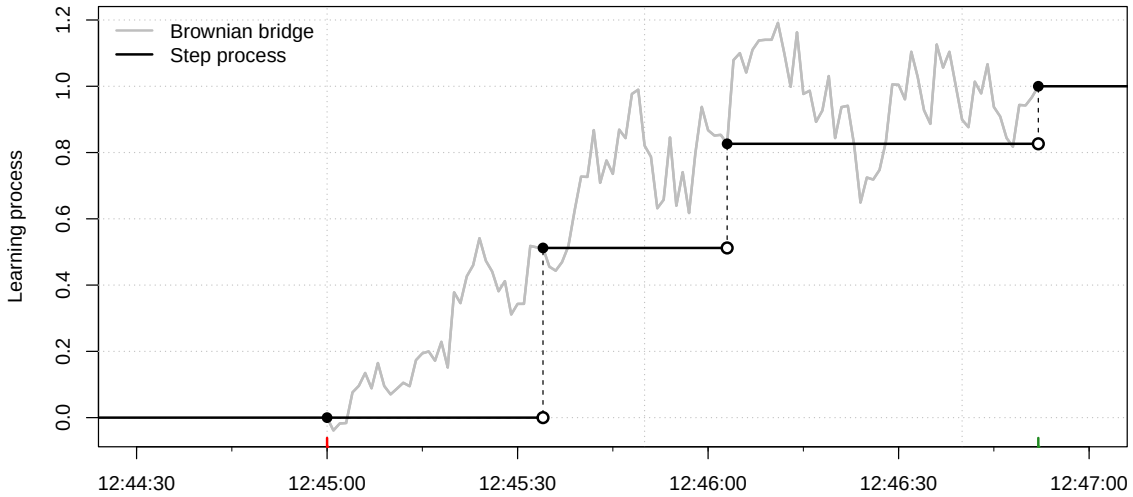
The Brownian bridge and the waiting times exist in continuously observed prices and are functions of the data generating features that delay jumps. They have nothing to do with the data frequency used by the econometrician.

A.2.3 Example of a step function

Figure 8 plots an example of such a step function for one delayed jump. The efficient stock price jumps at $i\Delta_n = 11,701$ or 12:45:00. The waiting times between each of three steps are equal to $w_{1,1} = 34, w_{1,2} = 29, w_{1,3} = 49$ 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 sampled sizes are equal to 0.000, 0.512, 0.826, and 1.000 and the 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. This step process manifests as a gradual jump in the observed price as we saw in the methodology section.

Figure 8: A step function captures the speed with which observed prices incorporate new information



Note: This figure plots a sample path of the step function we use to contaminate an efficient jump. The efficient stock price jumps at $i\Delta_n = 11,701$ or 12:45:00. The waiting times between each of three steps are equal to $w_{1,1} = 34, w_{1,2} = 29, w_{1,3} = 49$ seconds. The sampled sizes are equal to 0.000, 0.512, 0.826, and 1.000 and the increments are equal to $\Delta L_{1,0} = 0.000$, $\Delta L_{1,1} = 0.512$, $\Delta L_{1,2} = 0.314$, and $\Delta L_{1,3} = 0.174$.

B The vanilla rearrangement algorithm

Suppose that we want to rearrange the jump-event matrix of size $h \times q$. The rearrangement algorithm of Puccetti and Rüschendorf (2012) and Embrechts et al. (2013) loops over each column in a matrix to order it oppositely to the sum of the other columns. If the matrix has a fixed target in the last column (see *e.g.*, Bernard et al., 2018, 2017), the algorithm is as follows:

1. Randomly shuffle the elements (in each of the first $q-1$ columns) to obtain the starting matrix of the algorithm:

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

2. Iteratively rearrange the l-th column in 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:

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

3. Repeat Step 2 until no further changes occur. That is, until a matrix Γ^π is found with each column oppositely ordered to the sum of the other columns.