Ultra-High Frequency Verifiable Public Randomness from Tick-by-Tick Data

Silvia Onofri[1] Andrey Shternshis[2] Stefano Marmi[1]

[1] Scuola Normale Superiore, Pisa (Italy)
[2] Uppsala University, Uppsala (Sweden)

DeFi&Crypto Workshop - 27/01/2026

Random Beacons

What is a Random Beacon?

A Random Beacon is a protocol or service that regularly publishes random values (seeds) that are unpredictable before publication and publicly verifiable afterwards.
A secure beacon must satisfy three core properties:

1. Unpredictability: No party (including the beacon provider) can predict the value of a future beacon output.
2. Verifiability: Anyone can verify that the beacon output was generated correctly according to the public protocol.
3. Availability: The beacon produces new values at guaranteed, regular intervals.

Random beacons are useful in scenarios such as lotteries, public elections, audit challenges, cryptographic protocols, etc.

Sources of entropy to build random beacons

- Physical methods: dice, coin flips, lottery ball machines (easy to understand, hard to scale remotely)
- Pre-published randomness: random tables, printed books of values (verifiable but limited entropy and risk of re-use)
- Network / harvested data: news headlines, weather reports, blockchains
- **Financial/market data**

Efficient markets and random number generators

- Modern portfolio theory (Markowitz, Sharpe) and the statistical evidence that stock prices follow random walks (Mandelbrot, Fama), has led to the Efficient Market Hypothesis, see Samuelson (1965), Fama (1970).

Efficient Market Hypothesis

Asset prices fully and fairly reflect all available information.

Weak EMH

The price of a financial asset in an efficient market is a martingale, it behaves like a random walk.

- Therefore, an efficient market (EM) is a candidate for being a random number generator (RNG): If we code only the direction of market returns, then an efficient market should generate a random stream of 0s and 1s representing negative and positive returns, respectively.
- Whether a market is efficient can then be tested in much the same way that a hardware RNG might be tested.

Clark's proposal

In [Clark and Hengartner, 2010], Clark and Hengartner propose a model of random beacon that uses a stock market closing prices to build a random beacon.

- Fit a geometric Brownian motion (the Black–Scholes stochastic model) to historical closing prices;
- Use a Monte Carlo simulation to predict many next-day outcomes to bound conservative min-entropy ($6 - 9$ bits of entropy per day);
- Use a portfolio of stocks to obtain more entropy (accounting for correlations);
- Use a randomness extractor to produce the beacon output.

Tick data and methodology

Random beacon based on tick data

Our idea is to build a random beacon based on ultra-high frequency tick data:

- Longer strings $\Rightarrow$ more entropy
- Model-free: no need of any model and does not require to run any simulation.
- Online approach: one can compute the strings while data becomes available.

In [Onofri et al., 2025], we develop a methodology to extract random strings from tick data.

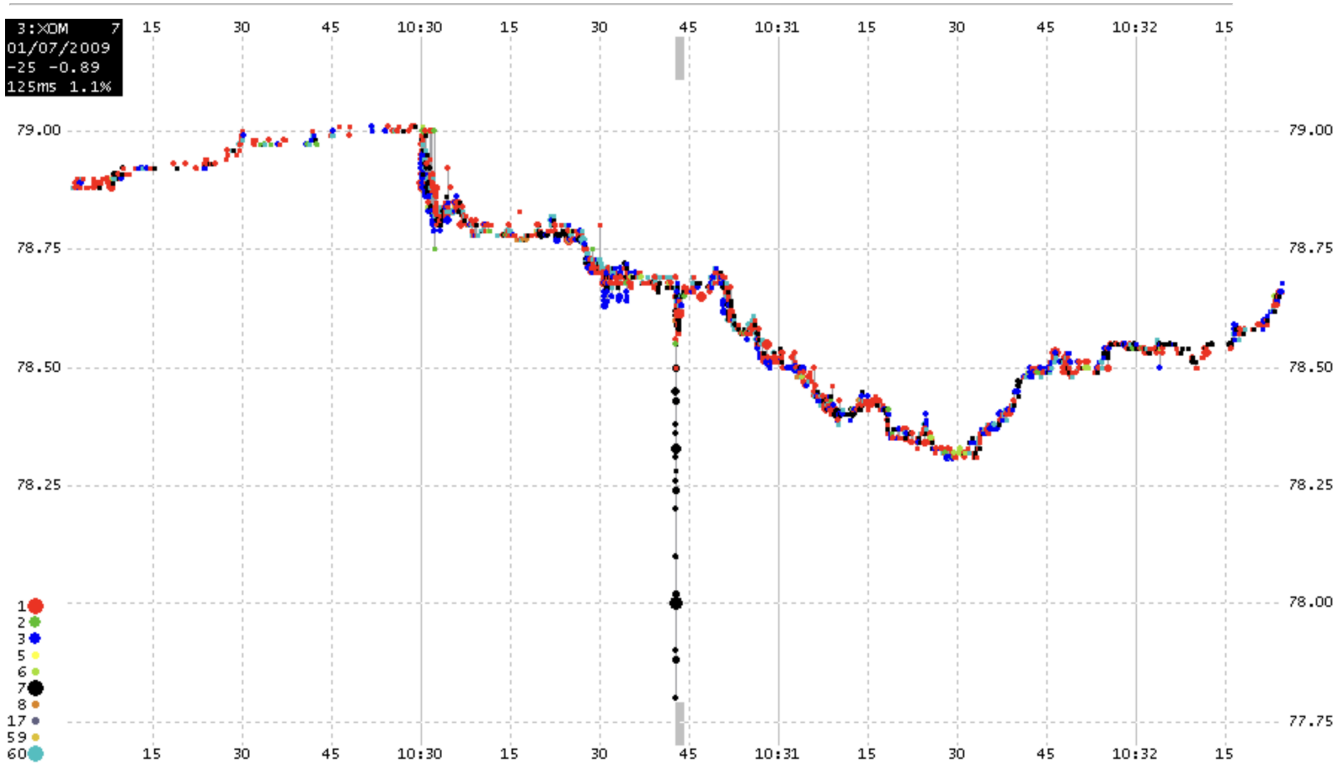
The journey from ticks to randomness

- *What is Tick Data?* Every single trade event for a stock, e.g. Apple (AAPL).
- Every time shares are bought or sold a *tick* is created, recording at what price, and how many stocks were traded.
- The time frequency of this data varies a lot: You might get 50 events in one millisecond during a market open and then no events for two full seconds in the middle of the day.
- At this tick-by-tick level, the price series is not random at all. It has several predictable, short-term patterns (the so-called *microstructure noise*)
- For example, a large order being filled can create a temporary, predictable price impact as it "eats through" the available liquidity.
- *Algorithmic Footprints*: High-frequency trading algorithms might have their own patterns (e.g., market-making strategies). Around two-thirds of all the volume traded on stockmarkets is generated by algorithms.

Why do we need aggregation?

- We aggregate trades $\{s_t\}$ by looking at (sub)sequences $\{s_{j+i\ell}\}$ of ticks of step $\ell \geq 1$.
- We are essentially sampling the tick data at regular intervals in "trade time".
- *Why Does Aggregation Increase Randomness?* This is the key. As you aggregate over longer and longer periods, the (non-random) microstructure noise is "averaged out".

Why do we need aggregation?



Inquiries: pr@nanex.net

Publication Date: November 29, 2010
http://www.nanex.net

Figure 1: From Nanex "Flash Crash Analysis" study

Methodology: from prices to binary strings

- We consider only executed orders prices $\{s_1, s_2, \ldots, s_N\}$ and compare adjacent times to obtain binary strings $\overline{b} \in \{0,1\}^*$.

$$r = \frac{s_i}{s_{i-1}} \begin{cases} < 1 & \text{then } \overline{b} \to \overline{b}0 \\ = 1 & \text{then } \overline{b} \to \overline{b} \\ > 1 & \text{then } \overline{b} \to \overline{b}1 \end{cases}$$

- Then we construct other sequences by aggregating data by an aggregation level $\ell = 1, \ldots, 100$. We consider ℓ samplings to build binary strings $\overline{b_j}$, $j = 1, \ldots, \ell$.

$$r = \frac{s_{j+i\ell}}{s_{j+(i-1)\ell}} \begin{cases} < 1 & \text{then } \overline{b_j} \to \overline{b_j}0 \\ = 1 & \text{then } \overline{b_j} \to \overline{b_j} \\ > 1 & \text{then } \overline{b_j} \to \overline{b_j}1 \end{cases}$$

- From each initial sequence $\{s_t\}$ we get $\sum_{k=1}^{100} k = 5050$ different binary sequences. We test all these sequences with standard batteries of randomness tests.

Dataset

Asset	Ticker	Mean price	Standard deviation of price	Daily trading volume	Daily number of transactions	Average time between transactions
Apple Inc.	AAPL	153.47	0.93	12,184,032	136,136	0.165
Microsoft Corporation	MSFT	251.78	1.37	4,529,093	84,342	0.269
Tesla Inc.	TSLA	388.02	3.81	8,686,354	178,704	0.127
Intel Corporation	INTC	30.15	0.20	7,055,642	38,255	0.595
Eli Lilly and Company	LLY	327.33	1.73	370,050	11,404	2.086
Snap Inc.	SNAP	10.67	0.14	4,967,779	18,521	1.358
Ford Motor Company	F	13.93	0.10	4,468,175	12,954	1.815
Carnival Corporation & plc	CCL	9.24	0.12	5,874,376	15,372	1.518
SPDR S&P 500 ETF	SPY	390.52	1.56	9,136,137	95,181	0.246

Mean price, its standard deviation, trading volume, number of transactions, and average time between transactions are calculated for each day and then are averaged over 80 days. Trading volume is summed up for each day. Average time is given in seconds.

Our study covers 80 trading days between 01-08-2022 and 21-11-2022.

Previous works

We continue the analysis in [Shternshis and Marmi, 2025].

- The predictability of ultra-high frequency time series is assessed by two entropy-based randomness tests.
- The authors transform the tick-by-tick time series into binary sequences. Their experiments prove that the degree of predictability decreases when we aggregate data by a number of transactions.

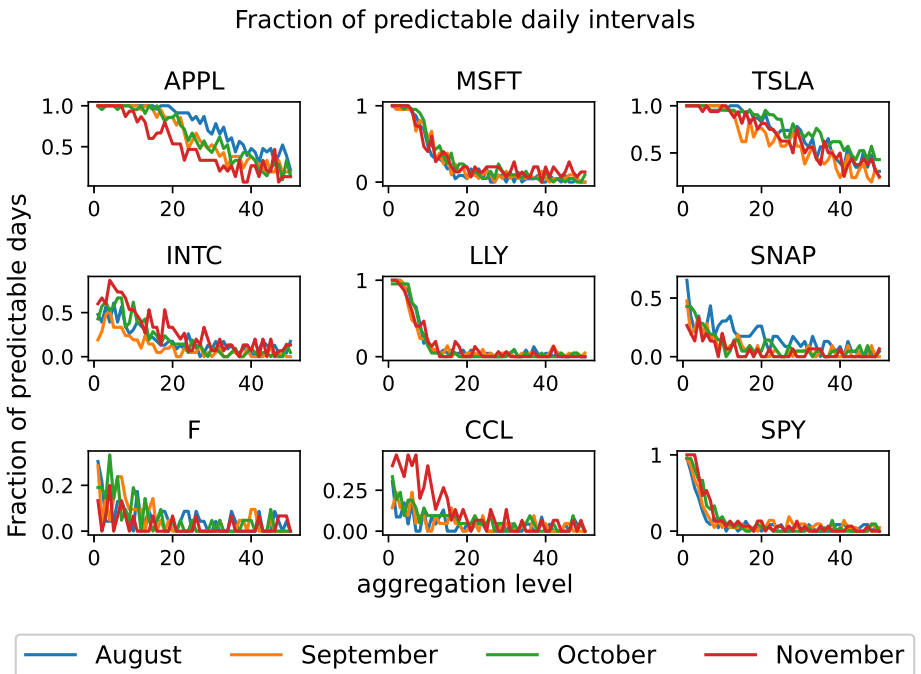


Figure 2: Fraction of predictable days for different months

Batteries of tests

We use two batteries to analyze our data:

1. NIST Statistical Test Suite
2. TestU01

\+ Two entropy-based randomness tests from [Shternshis and Marmi, 2025]

Classification of tests

- **Frequency tests**: they evaluate whether the proportion of 0s and 1s is coherent with the one of a uniform distribution.
- **Pattern tests**: they detect specific local structures, repeated patterns, and correlations between bits.
- **Entropy and complexity tests**: they assess how difficult a sequence is to compress or predict, using measures such as entropy estimation or linear complexity.
- **Spectral tests**: they apply discrete Fourier transforms to detect periodic structures or unexpected frequency spikes that would not be present in truly random data.
- **Random Walks tests**: they analyze the cumulative behavior of sequences interpreted as random walks, checking for imbalances, excursions from the origin, and path regularities.

Taxonomy of tests

Category	**NIST STS**	**Rabbit**	**Alphabit**
Frequency tests	Frequency (Monobit), Frequency Test within a Block	MultinomialBitsOverlapping, HammingWeight	MultinomialBitsOverlapping (x4)
Pattern tests	Runs Test, Longest Run of Ones in a Block, Non-overlapping Template Matching, Overlapping Template Matching, Serial Test	ClosePairsBitMatch(x2), LongestHeadRun, PeriodsInStrings, HammingCorrelation(x3), HammingIndependence(x3), AutoCorrelation(x2), Runs Test	HammingCorrelation, HammingIndependence (x2)
Entropy and complexity tests	Binary Matrix Rank Test, Maurer's Universal Statistical Test, Linear Complexity Test, Approximate Entropy Test	AppearanceSpacings, LinearComp, LempelZiv, MatrixRank(x3)	–
Spectral tests	Discrete Fourier Transform	Fourier1, Fourier3	–
Random walks tests	Cumulative Sums Test, Random Excursions Test, Random Excursions Variant Test	RandomWalk1(x3), RandomExcursions, RandomExcursionsVariant	RandomWalk1(x2)

Shannon entropy

Definition ([Shannon, 1948])

Let $X = \{X_1, X_2, \dots\}$ be a stationary random process with finite alphabet A and measure p. The k-th order entropy of X is

$$H_k(X) = -\sum_{x^k \in A^k} p(x^k) \log p(x^k)$$

The **process entropy** (aka **entropy rate**) of X is $h(X) = \lim_{k\to\infty} \frac{H_k(X)}{k}$.

If $X = \{X_1, X_2, \dots.\}$ is a sequence of i.i.d. binary Bernoulli variables with $p = (\frac{1}{2}, \frac{1}{2})$, then the process entropy is 1 bit.

Shannon Entropy test

- Divide the sequence into $n_b = \lfloor \frac{n}{k} \rfloor$ non-overlapping blocks of length k: $\hat{x}_t = \{x_{(t-1)k+1}, x_{(t-1)k+2}, \ldots, x_{tk}\}, t \in [1, n_b]$
- Compute empirical frequencies $\hat{f}_j$ of all blocks of length k: $\hat{f}_j = \sum_{t=1}^{n_b} \mathcal{I}(\hat{x}_t = a_j),\ a_j \in \{0,1\}^k, j \in [1, 2^k]$
- Estimate the Shannon entropy: $\hat{H} = -\sum_j \frac{\hat{f}_j}{n_b} \ln \frac{\hat{f}_j}{n_b}$
- Test if the estimation is close to the possible maximum of entropy: $Y_1 = 2n_b(k \ln 2 - \hat{H}), \quad H_0 : Y_1 \sim \chi^2(2^k - 1)$

KL distance and Mutual information

Definition

The **Kullback–Leibler divergence** (or **relative entropy**) between two probability distributions P and Q is

$$D(P \parallel Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}.$$

Definition

Let (X, Y) be a pair of random variables with values over the space $X \times Y$. The **mutual information** is defined as

$$I(X; Y) = D_{KL}(P_{XY} || P_X \otimes P_Y).$$

Kullback-Leibler (KL) divergence test

- Define $n_o = n - k + 1$ overlapping blocks:
  $\bar{x}_t = \{x_t, x_{t+1}, \ldots, x_{t+k-2}\},\ t \in [1, n_o]$
- Compute empirical frequencies of blocks of length k: $f_{ij} = \sum_{t=1}^{n_o} \mathcal{I}\left(\bar{x}_t = a_i\right) \mathcal{I}\left(x_{t+k-1} = a_j\right),\ a_i \in \{0,1\}^{k-1}, a_j \in \{0,1\}$
- Evaluate the test statistic and assess whether it follows a χ^2-distribution:
  $Y_2 = 2 \sum_{ij} f_{ij} \ln \frac{n_o f_{ij}}{f_{\cdot j} f_{i\cdot}}, \quad H_0 : Y_2 \sim \chi^2\left((2^{k-1} - 1)\right),$
  where $f_{\cdot j} = \sum_i f_{ij}$ and $f_{i\cdot} = \sum_j f_{ij}$

Sanity check using RNGs

Since random number generators (RNGs) can be very different depending on the nature of the "natural source" of randomness they use, we choose three RNGs with different sources to perform a sanity check on the tests:

- Quantis USB (ID Quantique) - quantum physics
- *urandom* from Linux - environmental noise
- Möbius function and the Riemann Hypothesis - pure mathematics

We generate strings of approximately the same length of our tick data strings, use the same methodology on them, and run a sanity check by fixing a threshold of 2% of accepted errors: if the number of failed tests is higher than the threshold for all the three RNGs, we consider the sanity check failed for that string-length and do not use the test for the corresponding financial data.

Sanity check results

NIST STS					
N.	Test	50K	100K	500K	1M
1	Frequency	✓			
2	BlockFrequency	✓	✓		
3	CumulativeSums	✓	✓		
4	Runs	✓	✓	✓	
5	LongestRun	✓	✓	✓	
6	Approx. Entropy	✓	✓	✓	✓
7	Serial	✓	✓	✓	✓
8	FFT	✓	✓		
9	NonOverlappingTemplate	✓	✓		
ALPHABIT BATTERY					
1	MultinomialBitsOverlapping, $L = 2$	✓	✓	✓	✓
2	MultinomialBitsOverlapping, $L = 4$	✓	✓	✓	✓
3	MultinomialBitsOverlapping, $L = 8$	✓	✓	✓	✓
4	MultinomialBitsOverlapping, $L = 16$	✓	✓	✓	✓
5	HammingIndependence, $L = 16$	✓	✓	✓	✓
6	HammingIndependence, $L = 32$	✓	✓	✓	✓
7	HammingCorrelation, $L = 32$	✓	✓	✓	✓
8	RandomWalk1, $L = 64$	✓	✓		
9	RandomWalk1, $L = 320$	✓	✓	✓	

RABBIT BATTERY					
N.	Test	50K	100K	500K	1M
1	MultinomialBitsOverlapping				
2	ClosePairsBitMatch, $t = 2$	✓	✓	✓	✓
3	ClosePairsBitMatch, $t = 4$		✓		
4	AppearanceSpacings			✓	✓
5	LinearComp	✓	✓	✓	✓
6	LempelZiv	✓	✓	✓	✓
7	Fourier1	✓	✓	✓	✓
8	Fourier3	✓	✓	✓	✓
9	LongestHeadRun	✓	✓	✓	✓
10	PeriodsInStrings	✓	✓	✓	✓
11	HammingWeight, $L = 32$	✓	✓	✓	✓
12	HammingCorrelation, $L = 32$	✓	✓	✓	✓
13	HammingCorrelation, $L = 64$	✓	✓	✓	✓
14	HammingCorrelation, $L = 128$		✓	✓	✓
15	HammingIndependence, $L = 16$	✓	✓	✓	✓
16	HammingIndependence, $L = 32$	✓	✓	✓	✓
17	HammingIndependence, $L = 64$	✓	✓	✓	✓
18	AutoCorrelation, $d = 1$	✓	✓	✓	✓
19	AutoCorrelation, $d = 2$	✓	✓	✓	✓
20	Run			✓	✓
21	MatrixRank, 32×32	✓	✓	✓	✓
22	MatrixRank, 320×320	✓	✓	✓	✓
23	MatrixRank, 1024×1024	✓	✓	✓	✓
24	RandomWalk1, $L = 128$	✓			
25	RandomWalk1, $L = 1024$	✓	✓	✓	✓
26	RandomWalk1, $L = 10016$	✓	✓	✓	✓

Results

When running the successful tests on the strings obtained from financial data, we get the following results:

1. Our study aligns with the findings in [Shternshis and Marmi, 2025], reaffirming that, generally, randomness tends to increase as the aggregation level grows up.

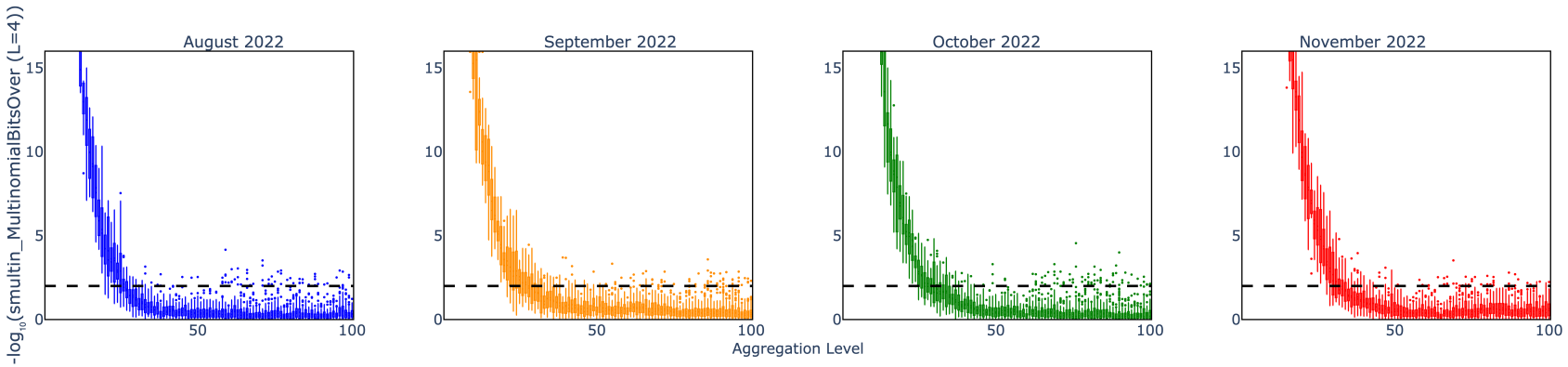


Figure 3: Results of smultin_MultinomialBitsOver ($L = 4$) test from Alphabit applied to CCL data.

Randomness tests results

2. However, novel methods and tests sometimes reveal exceptions; for instance, there are cases where even at high aggregation levels the result of tests reveals predictability. *We attribute this persistent predictability to their high trading activity: sometimes it happens that aggregating until level 100 is not enough to see randomness emerge from the strings.*

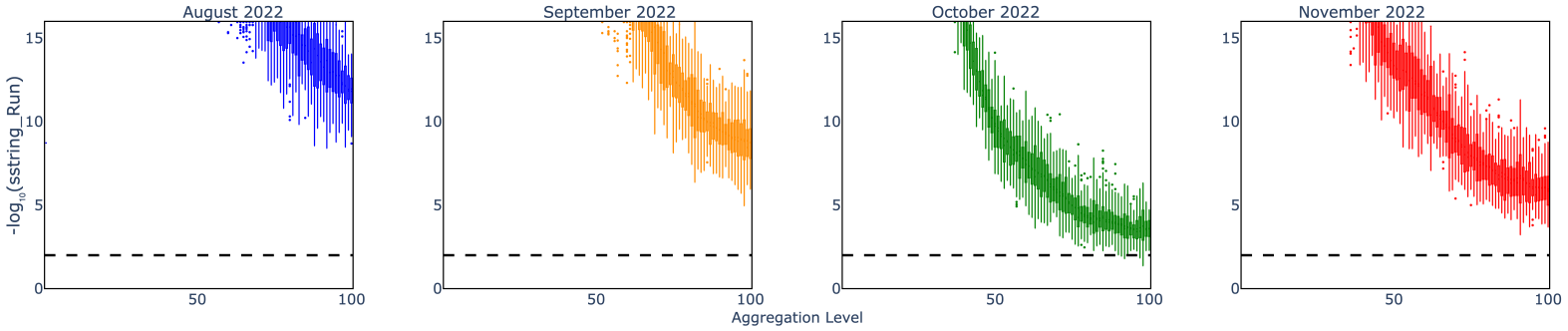
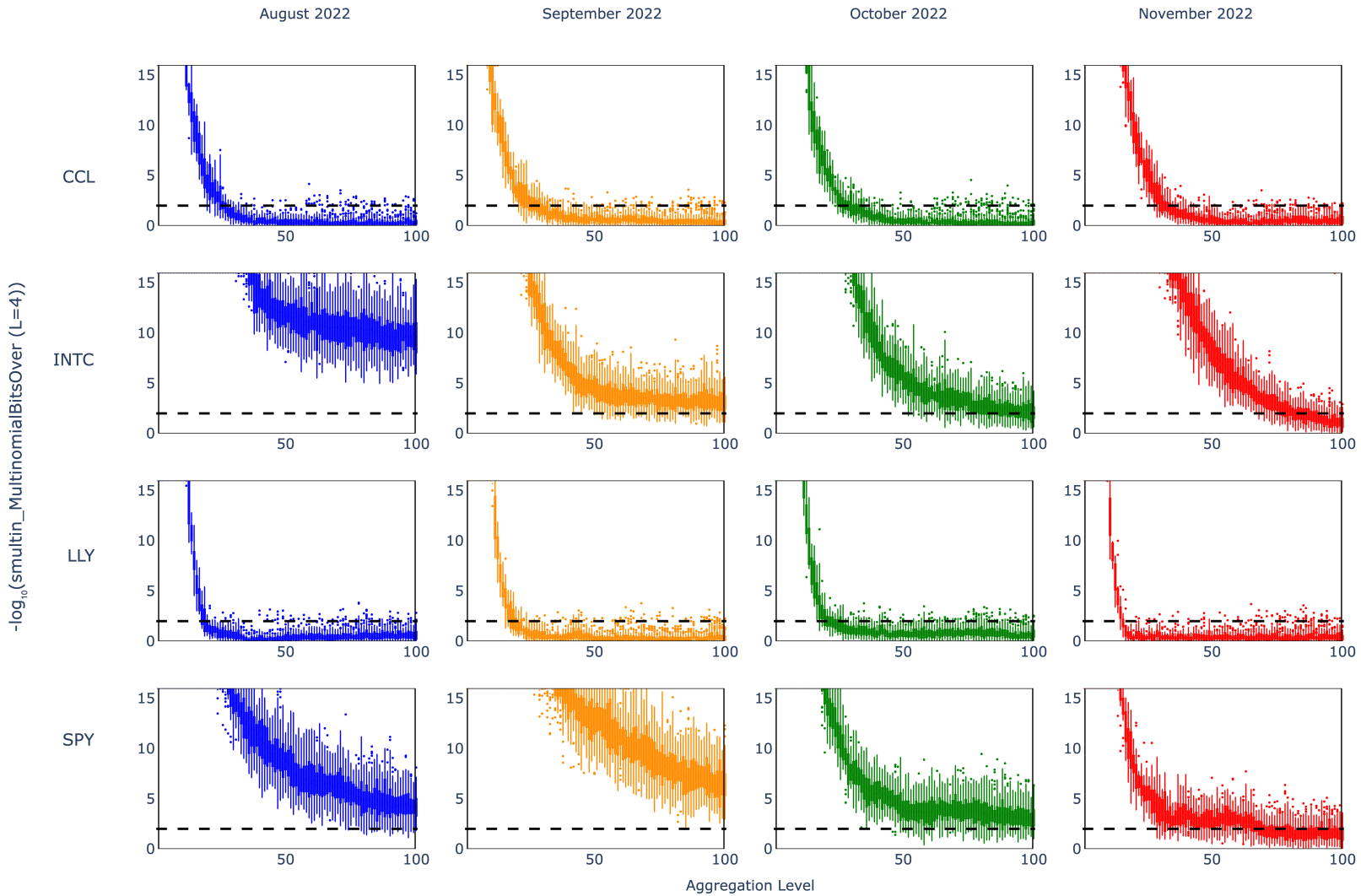
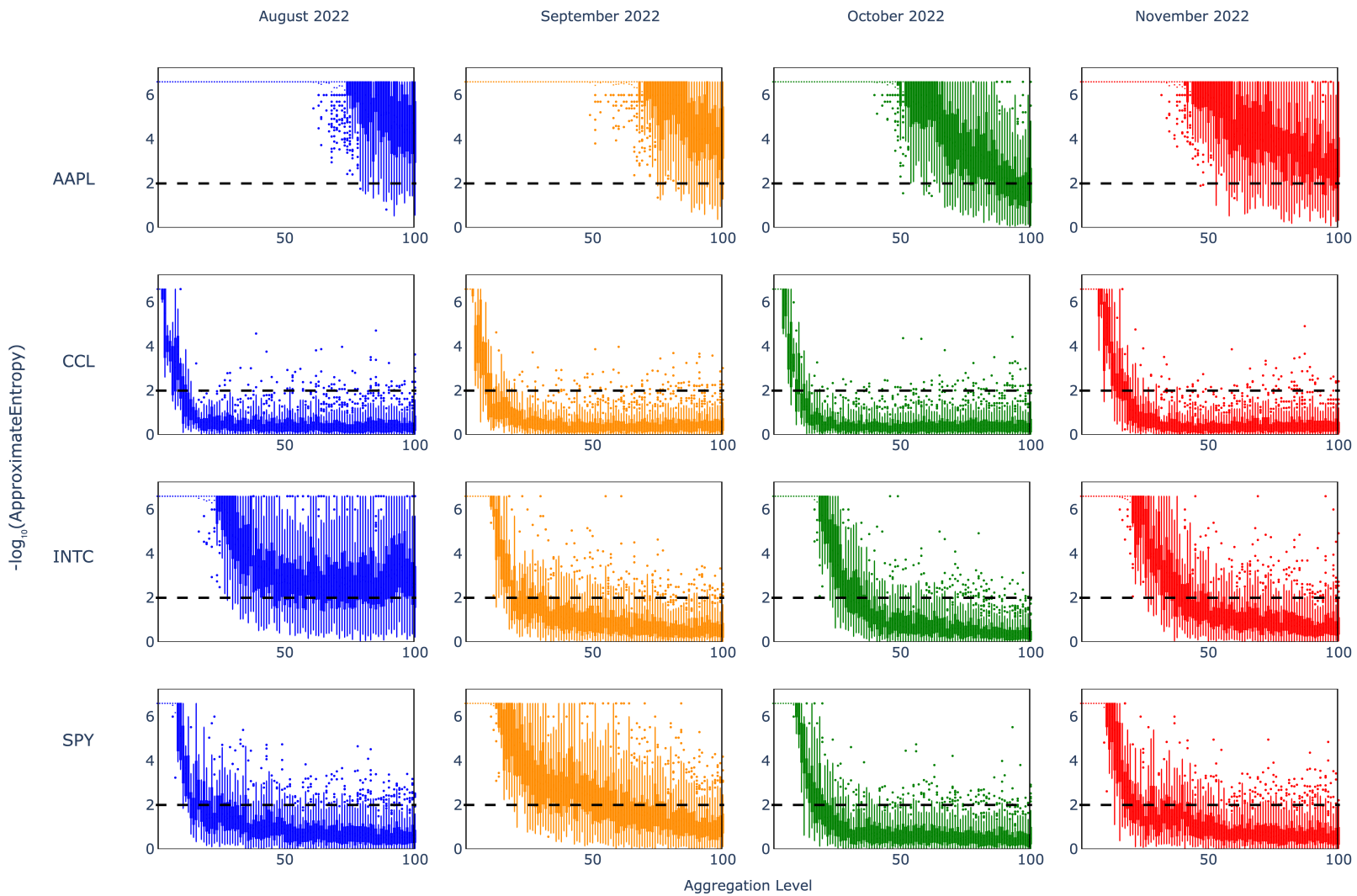


Figure 4: Results of Run test from Rabbit applied to AAPL data.

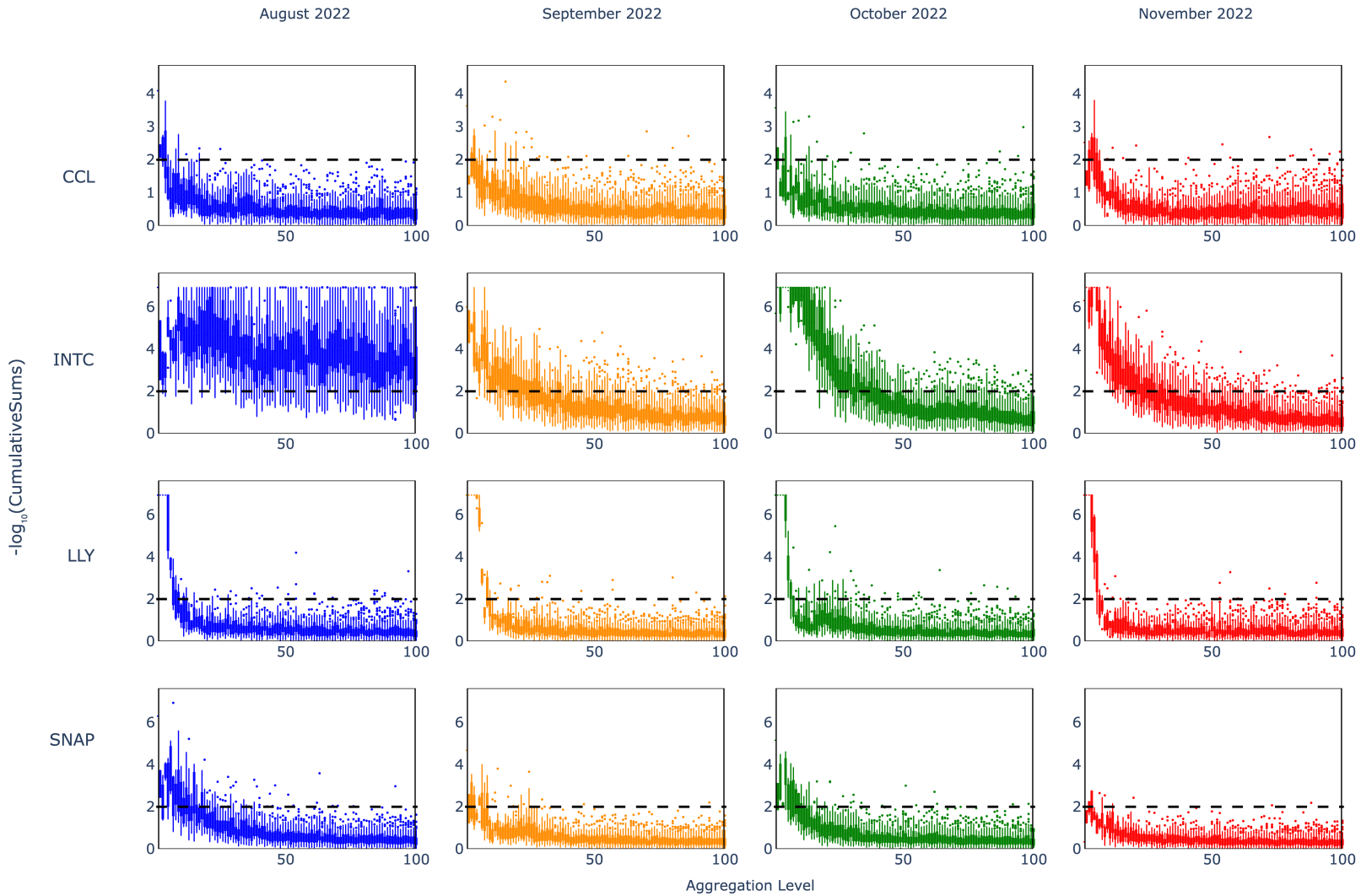
MultinomialBitsOver ($L = 4$) test



ApproximateEntropy



Cumulative Sums



Application to random beacons

The methodology we introduce in [Onofri et al., 2025] gives us the possibility to let non-random binary strings obtained by UHF data be whitened out by the process of aggregation for most stocks. Using this randomizing process to build a random beacon brings several advantages:

- Online process: it does not require future information on tick data, so we can update the strings as new data becomes available. Strings can be computed on-the-fly.
- Model-free: it does not assume any hypothesis, any model and does not require to run any simulation.
- More entropy: up to 500 entropy bits per day - if we consider stocks of nearly 1 000 000 of bits per month aggregated at level 100.
- Strings can be used as a random seed as they are, or as high-quality input for a randomness extractor.

Randomizing cryptocurrency data

Randomizing cryptocurrency data

There are several reasons to apply our methodology to cryptocurrency data.

- The major cryptocurrencies exhibit much higher transaction volumes (for example, the trading of Bitcoin on a major CEX recorded more than 169 million transactions in November 2025).
  $\Rightarrow$ extraction of more entropy per unit time
- Stock markets operate only during limited trading hours, while cryptocurrency markets are open 24/7
  $\Rightarrow$ *real* online approach.

By applying our aggregation-based whitening framework to crypto transaction data, we aim to quantify whether their characteristics lead to higher-quality randomness and to assess their suitability for decentralized randomness generation and beacon applications.

Results on cryptocurrency data

Since tick data show a huge amount of correlation, we decided to start aggregating from 1s data.

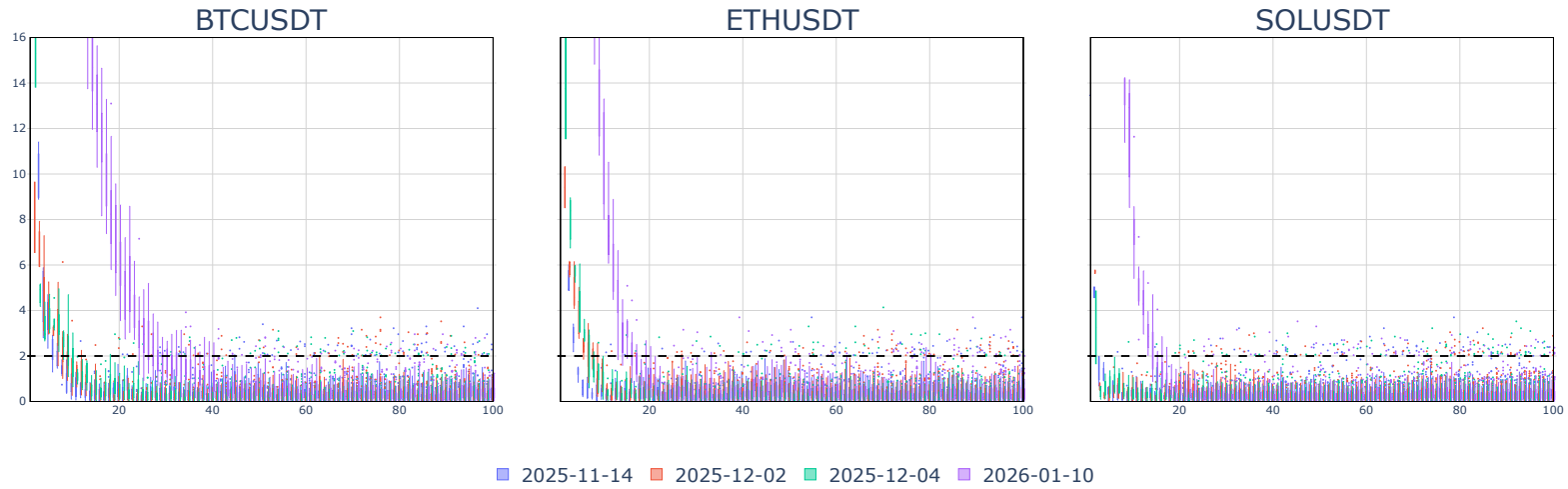


Figure 8: Results of Fourier3 test from Rabbit applied to BTCUSDT, ETHUSDT and SOLUSDT data.

Results on cryptocurrency data

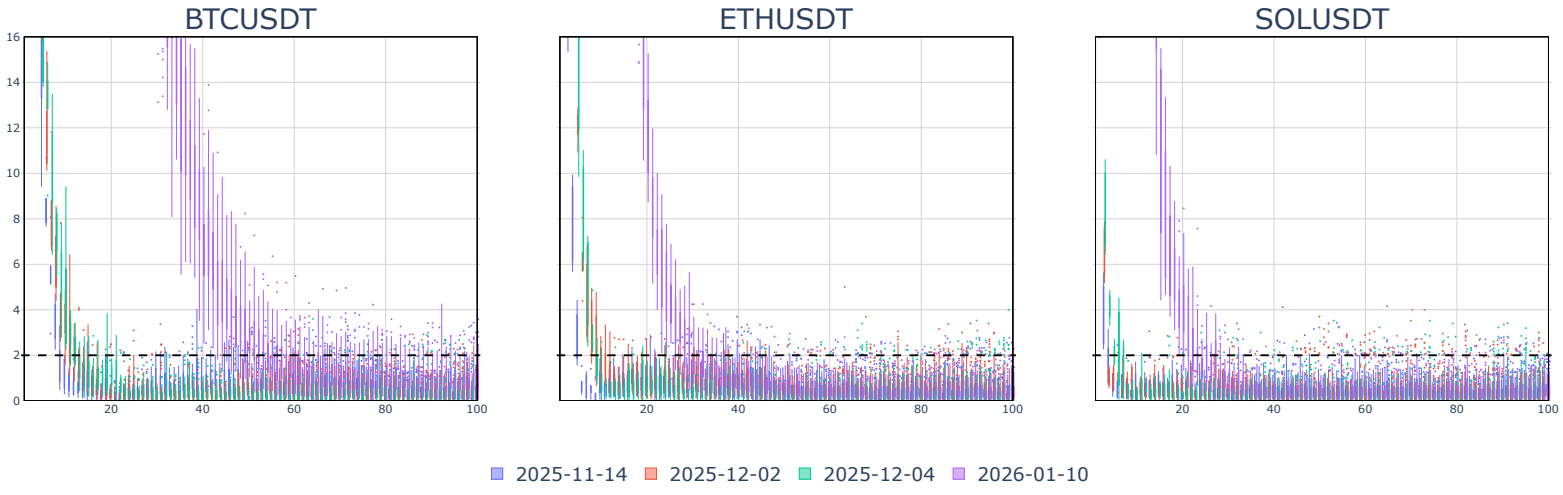
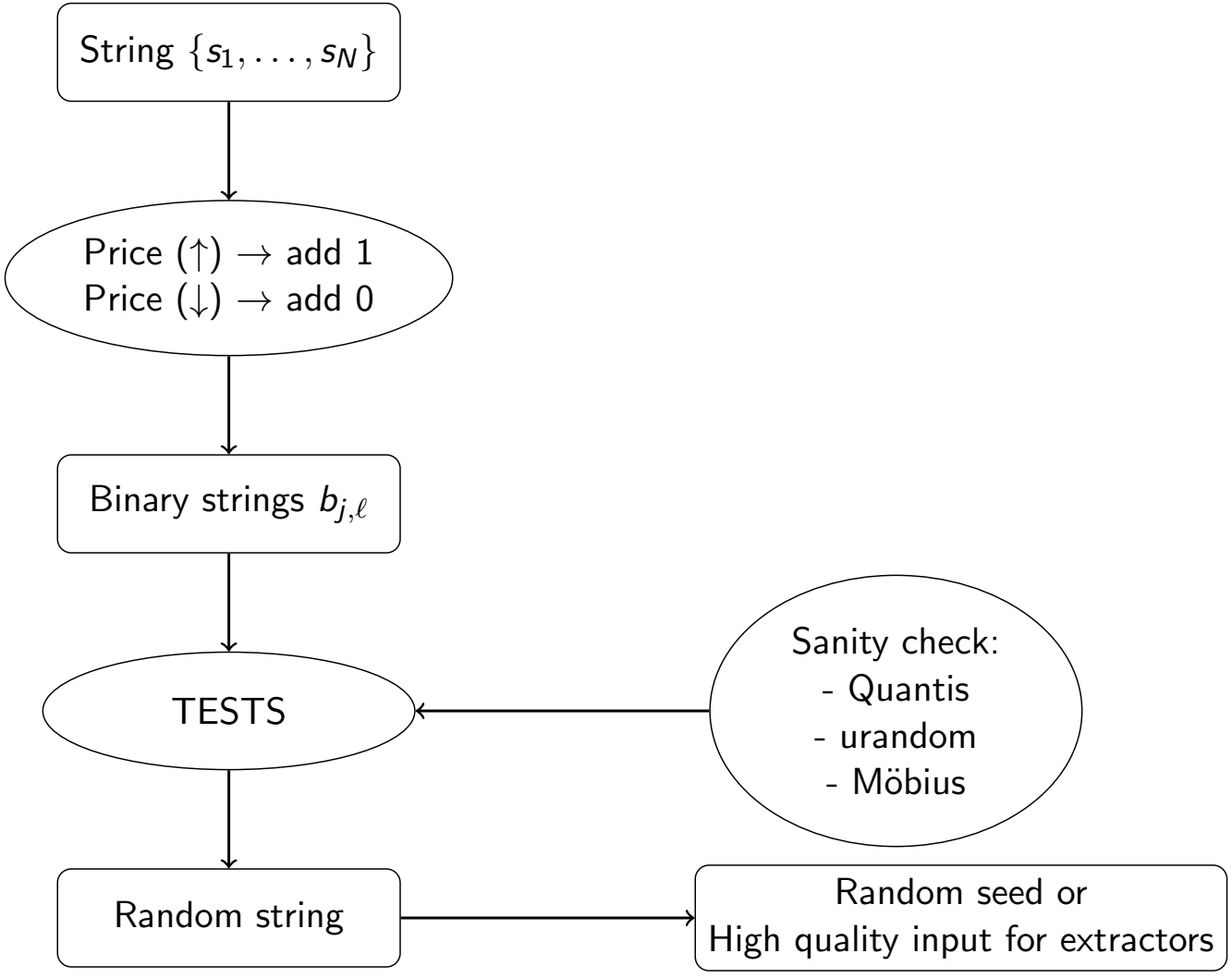


Figure 9: Results of MultinomialBitsOverlapping with $L = 4$ test from Alphabit applied to BTCUSDT, ETHUSDT and SOLUSDT data.

Conclusions and bibliography

Methodology



Conclusions

- We introduce a new methodology to get random strings from tick data. We apply standard RNG test batteries that allow for a *multidimensional* analysis (Frequency, Patterns, Spectral, etc.).
- We confirm that aggregation increases randomness, but monotonicity and speed of convergence may depend on the asset and on the test employed.
- Our methodology can be seen as an online model-free whitening process, taking correlated strings to random ones, then a good candidate for the construction of a random beacon.

THANK YOU!

Randomness is in the eye of the beholder, or more precisely, in its computational capabilities. [...] This viewpoint, developed by M. Blum, S. Goldwasser, S. Micali, and A. Yao in the early 1980s, marks a significant departure from older views and has led to major breakthroughs in computer science of which the field of cryptography is only one (Wigderson, 2009).

Bibliography I

- Clark, J. and Hengartner, U. (2010).
  On the use of financial data as a random beacon.
  In *2010 Electronic Voting Technology Workshop/ Workshop on Trustworthy Elections (EVT/WOTE 10)*, Washington, DC. USENIX Association.
- Onofri, S., Shternshis, A., and Marmi, S. (2025).
  Emergence of randomness in temporally aggregated financial tick sequences.
- Shannon, C. E. (1948).
  A mathematical theory of communication.
  The Bell System Technical Journal, 27(3):379–423.
- Shternshis, A. and Marmi, S. (2025).
  Price predictability at ultra-high frequency: Entropy-based randomness test.
  Communications in Nonlinear Science and Numerical Simulation, 141:108469.