IMPERMANENT: A LIVE BENCHMARK FOR TEMPORAL GENERALIZATION IN TIME SERIES FORECASTING

Azul Garza **Renée Rosillo**
TimeCopilot
{azul,renee}@timecopilot.dev

Rodrigo Mendoza-Smith
Independent Researcher

David Salinas
ELLIS Institute Tübingen

Andrew Robert Williams
Mila – Quebec AI Institute
Université de Montréal

Arjun Ashok
Mila – Quebec AI Institute
Université de Montréal

Mononito Goswami
Amazon Web Services[†]

José Martín Juárez
TimeCopilot

ABSTRACT

Recent advances in time-series forecasting increasingly rely on pre-trained foundation-style models. While these models often claim broad generalization, existing evaluation protocols provide limited evidence. Indeed, most current benchmarks use static train-test splits that can easily lead to contamination as foundation models can inadvertently train on test data or perform model selection using test scores, which can inflate performance. We introduce `Impermanent`, a live benchmark that evaluates forecasting models under open-world temporal change by scoring forecasts sequentially over time on continuously updated data streams, enabling the study of temporal robustness, distributional shift, and performance stability rather than one-off accuracy on a frozen test set. `Impermanent` is instantiated on GitHub open-source activity, providing a naturally live and highly non-stationary dataset shaped by releases, shifting contributor behavior, platform/tooling changes, and external events. We focus on the top 400 repositories by star count and construct time series from issues opened, pull requests opened, push events, and new stargazers, evaluated over a rolling window with daily updates, alongside standardized protocols and leaderboards for reproducible, ongoing comparison. By shifting evaluation from static accuracy to sustained performance, `Impermanent` takes a concrete step toward assessing when—and whether—foundation-level generalization in time-series forecasting can be meaningfully claimed. Code and a live dashboard are available at `https://github.com/TimeCopilot/impermanent` and `https://impermanent.timecopilot.dev`.

Track: Research

1 INTRODUCTION

Forecasting seeks to reduce uncertainty about the future. In practice, forecasters rely on a diverse set of models (statistical methods, machine learning approaches, and increasingly, foundation models), selecting among them based on properties of the data such as trend, seasonality, sparsity, scale, and domain structure, and validating choices through backtesting on historical observations (1). In recent years, *time-series foundation models* (TSFMs) have emerged with the promise of broad generalization across domains (2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16). Trained on large and heterogeneous collections of temporal data, these models aim to operate as generalists, producing accurate forecasts on previously unseen series with minimal adaptation. The central claim

[†] The work does not relate to the author's position at Amazon.

underlying this paradigm is *temporal generalization*, i.e. that learned representations transfer across datasets, frequencies, and domains.

However, despite this promise, most empirical evaluations of TSFMs rely on *static* benchmarks. Datasets are fixed in time, and evaluation is conducted on held-out splits drawn from the same underlying distribution as the training data, as in widely used benchmarks such as GIFT-Eval(17), FEV (18), and the Monash Forecasting Repository(19). While this protocol measures cross-sectional generalization, it does not fully test whether performance persists *across time* in evolving, non-stationary environments. Real-world forecasting is inherently dynamic: data distributions shift, new series appear, and structural breaks occur. Moreover, public datasets used for pretraining are often reused in downstream benchmarks, raising concerns about memorization, data leakage, and test-set contamination (20).

The importance of this sequential-evaluation requirement is well recognized in adjacent literatures. For example, the *prequential* viewpoint holds that probabilistic forecasts should be evaluated in the order they are made (21). More broadly, the concept-drift literature examines how predictive systems degrade as data distributions evolve over time (22). In forecasting practice, rolling-origin evaluation is similarly recommended because it better reflects real-world deployment (23; 24). Related concerns also arise for language foundation models, where live benchmarks aim to reduce contamination and ensure that reported results are not affected by prior exposure to test examples (25; 26). Closest to our setting, (27) considers sequential evaluation for LLM prediction of single future events, but not for time-series forecasting.

Contributions. To address these limitations, we introduce `Impermanent`, which, to our knowledge, is the first *live* benchmark designed specifically to evaluate *temporal generalization* in time-series forecasting. `Impermanent` makes temporal generalization measurable through a live, leakproof evaluation protocol in which forecasts are generated and scored sequentially over time. This setup enables analyses that are difficult in static benchmarks, including sustained accuracy, robustness to distribution shift and shocks, and the stability of model rankings under ongoing change. This first iteration of `Impermanent` is built on GitHub software development activity, a naturally live and highly non-stationary environment shaped by releases, shifting contributor behavior, workflow and tooling changes, and external events (28; 29; 30). The benchmark is designed around a prequential, deployment-faithful evaluation loop: at each cutoff time, models must produce forecasts *before* the corresponding ground truth exists.

2 THE `Impermanent` BENCHMARK

We instantiate `Impermanent` on GitHub activity using GH Archive event streams (29) (Figure 3). We track four event types (issues opened, pull requests opened, push events, and new stargazers) for 400 repositories (selected by star count), construct time series at four forecast frequencies (daily, $h = 7$; weekly, $h = 4$; and monthly, $h = 1$), and stratify repositories by activity level.In the current release, each repository–event-type pair defines a *univariate* series (we forecast one target count at a time); capturing cross-repository co-movement and incorporating additional covariates are important next steps. Figures 1 and 2 provide exploratory views on a random sample of 25 repositories, visualizing non-stationarity in raw weekly counts and in compact dynamics descriptors, including the *spectral centroid* and *spectral entropy*. Formally, let $y_0, \ldots, y_{n-1}$ denote a series, $X_k = \sum_{t=0}^{n-1} y_t e^{-i2\pi kt/n}$ its real-FFT coefficients for $k = 0, \ldots, K-1$ with $K = \lfloor n/2 \rfloor + 1$, and $f_k = k/n$ (cycles per time step); defining $P_k = |X_k|^2$ and $p_k = P_k / \sum_j P_j$, we compute the spectral centroid C and normalized spectral entropy H as:

$$C = \frac{\sum_{k=0}^{K-1} f_k \, |X_k|}{\sum_{k=0}^{K-1} |X_k|}, \qquad H = -\frac{1}{\log K} \sum_{k=0}^{K-1} p_k \log p_k. \tag{1}$$

Qualitatively, larger C indicates relatively faster dynamics (more power at higher frequencies), while larger H indicates a more diffuse, less structured spectrum (power spread across many frequencies). In Figure 2, we see that the GitHub streams occupy a wide range of (C, H) values and that different event types populate different regions of this space: some series concentrate power at lower frequencies (lower C) with more structured spectra (lower H), while others show higher-frequency, more broadband behavior (higher C and/or higher H). For `Impermanent`, the key takeaway is simple:

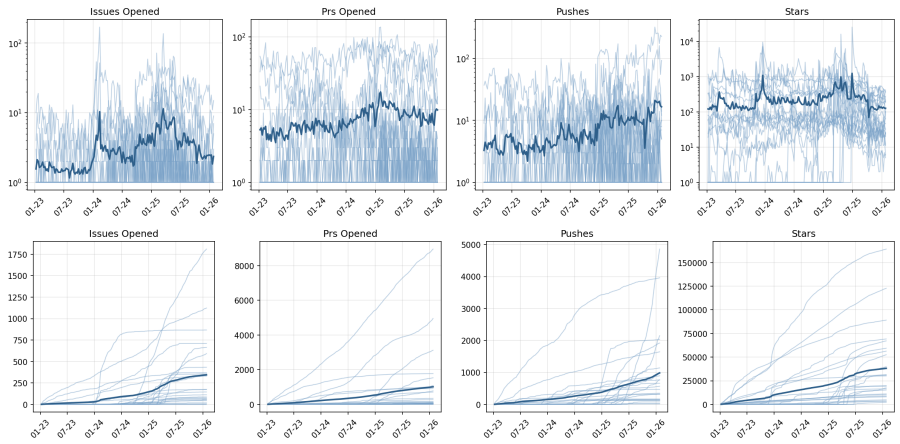


Figure 1: Weekly GitHub activity for a random sample of 25 repositories across four event types (issues opened, pull requests opened, push events, new stargazers). Top: weekly event counts. Bottom: cumulative counts over time.

the dataset mixes smooth, trend-like behavior with spiky, volatile behavior, so a good forecaster must handle both slow changes and sudden bursts rather than fitting one fixed pattern.

3 Evaluation Protocol

At each cutoff date, every model receives a context window of historical observations and must produce point and probabilistic forecasts for the next h periods before any ground truth is available. Cutoffs are spaced exactly one horizon apart, and the most recent cutoff is always excluded because observations may still be incomplete. Table 1 summarizes the evalution protocol.

	Daily	**Weekly**	**Monthly**
Horizon h	7	1	1
Max context	512	114	24
Cutoff step	7 d	1 w	1 mo
First cutoff	2026-01-04	2026-01-04	2025-10-01

Table 1: Evaluation protocol parameters. All frequencies use MASE and scaled CRPS.

We evaluate forecasts with two complementary metrics: MASE (Mean Absolute Scaled Error) for point accuracy (31), and scaled CRPS for the predictive distribution (32), estimated from nine quantile levels $\mathcal{Q} = \{0.1, 0.2, \ldots, 0.9\}$. For a series with training window $y_{1:T}$, seasonal period m, and horizon h, we use

$$\text{MASE} = \frac{\frac{1}{h}\sum_{i=1}^{h} |y_{T+i} - \hat{y}_{T+i}|}{\frac{1}{T-m}\sum_{t=m+1}^{T} |y_t - y_{t-m}|}, \qquad \text{CRPS} \approx \frac{1}{h}\sum_{i=1}^{h} \frac{2}{|\mathcal{Q}|} \sum_{\tau \in \mathcal{Q}} \rho_\tau(y_{T+i} - \hat{q}_{T+i}(\tau)), \quad (2)$$

where $\rho_\tau(u) = u\big(\tau - \mathbf{1}\{u < 0\}\big)$ is the pinball loss. We compute both metrics per series and aggregate per subdataset using the median. To make scores comparable across subdatasets, we scale each metric by the *zero model*, which always predicts zero: for model value v and zero-model score b (the metric value from all-zero forecasts), we report $v/\max(b, \tau_0)$, where τ_0 is the 10th percentile of strictly positive zero-model scores for that metric. This floor prevents unstable ratios when b is very small. We evaluate eleven models in three groups. Baselines are *ZeroModel* (which always predicts zero and provides the scaling denominator), *HistoricAverage*, and *SeasonalNaive*. Statistical models are AutoARIMA (33), AutoETS (34), AutoCES (35), Dynamic Optimized Theta (36), and Prophet (37), and all run on CPU. Foundation models are Chronos-2 (38), Moirai 2.0-R-Small (4),

TimesFM 2.5 (9), and TiRex (39), and each runs on an A10G GPU with batch size 64. Each model outputs nine quantile forecasts ($\tau \in \mathcal{Q}$), enabling direct comparison on MASE and scaled CRPS. We include only open source TSFMs with released weights and reproducible inference code, and we run all models through TimeCopilot (40). Section 5 describes the pipeline.

4 RESULTS

Table 2 illustrates the kind of analysis `Impermanent` enables. In this snapshot, pre-trained foundation models occupy the top four positions, with TiRex leading on three of four columns and Chronos achieving the best median CRPS. However, the picture is nuanced: SeasonalNaïve achieves a lower median MASE (0.49) than every statistical method, yet shows the poorest probabilistic calibration (CRPS rank 28.94), and AutoETS attains a CRPS rank (20.71) comparable to DynOptTheta (21.93) despite weaker point accuracy. Because `Impermanent` scores models sequentially over an evolving stream, these rankings will shift as new cutoffs accumulate, making it possible to track *whether* early advantages persist under continued distributional shift rather than taking any one leaderboard snapshot as definitive.

	Median value ↓		Mean rank ↓	
Model	MASE	CRPS	MASE	CRPS
HistoricAverage	2.095	4.376	30.857	27.417
SeasonalNaive	0.485	5.670	19.044	28.937
Prophet (37)	1.264	3.640	27.560	24.326
AutoARIMA (33)	0.754	3.450	21.677	23.289
AutoETS (34)	0.630	3.209	20.058	20.710
AutoCES (35)	0.618	3.248	19.356	23.346
DynOptTheta (36)	0.555	3.297	17.551	21.926
Moirai (4)	0.390	1.143	14.280	12.762
TimesFM (9)	0.373	1.053	12.070	11.376
Chronos (38)	*0.336*	**1.018**	*11.987*	*10.941*
TiRex (39)	**0.322**	*1.024*	**11.350**	**10.836**

Table 2: Overall leaderboard results on `Impermanent`, aggregated across all subdatasets, frequencies, and cutoff dates. *Median value* reports the median of the baseline-scaled metric across all evaluation instances; *mean rank* is the average rank computed hierarchically (within each group, then across subdatasets and frequencies). **Bold**: best per column; *italic*: second best. Models are sorted by MASE inside each group (worst first); horizontal rules separate naive baselines, statistical methods, and foundation models.

5 CONCLUSION AND FUTURE WORK

We introduced `Impermanent`, which, to our knowledge, is the first live benchmark designed to measure temporal generalization in time-series forecasting. By evaluating models sequentially on a continuously evolving data stream, with forecasts issued before outcomes are observed and scored only after those outcomes arrive, `Impermanent` enables analyses that static benchmarks cannot support. These include measuring sustained accuracy over time, assessing robustness under distributional change, and tracking the stability of model rankings as evaluation unfolds. All data pipelines, evaluation code, and leaderboard infrastructure are open and fully automated, enabling reproducibility and extension without reprocessing historical data.

This first iteration of `Impermanent` is built on GitHub software development activity, but the framework is designed to support broader future development. Natural next steps include expanding to additional live data streams, enriching forecasting tasks with auxiliary contextual information, and using longer evaluation horizons to better understand performance stability and ranking dynamics over time. More broadly, we hope `Impermanent` can serve as a shared resource for studying whether benchmark performance in static settings translates into reliable performance after deployment.

REFERENCES

[1] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3 edition, 2021. URL https://otexts.com/fpp3/. Accessed 2026-02-06.

[2] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series, 2024. URL https://arxiv.org/abs/2401.03955.

[3] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models, 2024. URL https://arxiv.org/abs/2402.03885.

[4] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

[5] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting, 2025. URL https://arxiv.org/abs/2410.04803.

[6] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts, 2025. URL https://arxiv.org/abs/2409.16040.

[7] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability, 2024. URL https://arxiv.org/abs/2407.07874.

[8] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models, 2024. URL https://arxiv.org/abs/2402.02368.

[9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023. URL https://arxiv.org/abs/2310.10688.

[10] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023. URL https://arxiv.org/abs/2310.08278.

[11] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

[12] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Autotimes: Autoregressive time series forecasters via large language models, 2024. URL https://arxiv.org/abs/2402.02370.

[13] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

[14] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023. URL https://arxiv.org/abs/2310.01728.

[15] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines, 2021. URL https://arxiv.org/abs/2103.05247.

[16] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. 2023. URL https://arxiv.org/abs/2310.03589.

[17] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024. URL https://arxiv.org/abs/2410.10393.

[18] Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting. *arXiv preprint arXiv:2509.26468*, 2025. URL https://arxiv.org/abs/2509.26468.

[19] Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *NeurIPS 2021 Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/eddea82ad2755b24c4e168c5fc2ebd40-Paper-round2.pdf.

[20] Marcel Meyer, Sascha Kaltenpoth, Kevin Zalipski, and Oliver Müller. Time series foundation models: Benchmarking challenges and requirements. *arXiv preprint arXiv:2510.13654*, 2025.

[21] A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–292, 1984. doi: 10.2307/2981683.

[22] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014. doi: 10.1145/2523813.

[23] J S Armstrong, editor. *Principles of forecasting: a handbook for researchers and practitioners*. Kluwer Academic Publishers, 2001. ISBN 0792379306. URL http://amazon.com/dp/0792379306.

[24] Rob J. Hyndman, George Athanasopoulos, Azul Garza, Cristian Challu, Max Mergenthaler, and Kevin G. Olivares. *Forecasting: Principles and Practice, the Pythonic Way*. OTexts, Melbourne, Australia, 2024. URL https://otexts.com/fpppy. Accessed on 22 August 2025.

[25] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 4, 2024.

[26] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference, March 2024.

[27] Ezra Karger, Houtan Bastani, Chen Yueh-Han, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip E. Tetlock. ForecastBench: A Dynamic Benchmark of AI Forecasting Capabilities, February 2025.

[28] Georgios Gousios and Diomidis Spinellis. Ghtorrent: Github's data from a firehose. In *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR)*, pages 12–21, 2012. URL https://cs.uwaterloo.ca/~m2nagapp/courses/CS846/1171/papers/gousios_msr12.pdf.

[29] Ilya Grigorik. Gh archive: Public github event data. https://www.gharchive.org/, 2011. Accessed 2026-02-06.

[30] Alexandre Decan, Eleni Constantinou, Tom Mens, and Henrique Rocha. Gap: Forecasting commit activity in git projects. *Journal of Systems and Software*, 165:110573, 2020. doi: 10.1016/j.jss.2020.110573.

[31] R J Hyndman and A B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. doi: 10.1016/j.ijforecast.2006.03.001.

[32] T Gneiting and M Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and its Application*, 1(1):125–151, 2014. doi: 10.1146/annurev-statistics-062713-085831.

[33] R J Hyndman and Y Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 27(1):1–22, 2008. doi: 10.18637/jss.v027.i03. URL http://www.jstatsoft.org/article/view/v027i03.

[34] R J Hyndman, A B Koehler, R D Snyder, and S Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18 (3):439–454, 2002. doi: 10.1016/S0169-2070%2801%2900110-8.

[35] Ivan Svetunkov, Nikolaos Kourentzes, and John Keith Ord. Complex exponential smoothing. *Naval Research Logistics (NRL)*, 69(8):1108–1123, August 2022. doi: 10.1002/nav.22074. URL http://dx.doi.org/10.1002/nav.22074.

[36] Jose A. Fiorucci, Tiago R. Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B. Koehler. Models for optimising the theta method and their relationship to state space models. *International Journal of Forecasting*, 32(4):1151–1161, 2016. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2016.02.005. URL https://www.sciencedirect.com/science/article/pii/S0169207016300243.

[37] S J Taylor and B Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. doi: 10.1080/00031305.2017.1380080.

[38] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

[39] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning, 2025. URL https://arxiv.org/abs/2505.23719.

[40] Azul Garza and Renée Rosillo. Timecopilot, 2025. URL https://arxiv.org/abs/2509.00616.

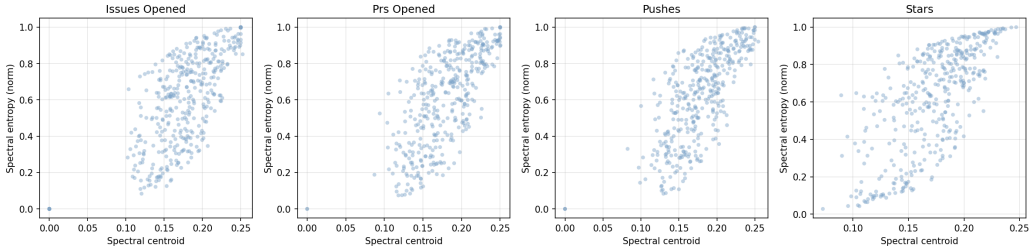
APPENDIX

EXPLORATORY STATISTICS

This section provides additional descriptive views of the GitHub activity streams used in `Impermanent`, complementing the main-text plots in Figures 1 and 2. The weekly trajectories reveal pronounced intermittency and burstiness: many repository–event series spend long stretches near low counts and then undergo short-lived spikes, especially for issues and pushes, while stars operate at substantially larger scales with occasional surges. In cumulative form, these same bursts appear as slope changes and step-like jumps, indicating that activity regimes are not stable over time even within a single repository–event pair.

Figure 2 provides a compact summary through spectral descriptors. Across all four event types, spectral entropy generally increases with spectral centroid, producing an upward wedge-shaped cloud: faster series tend to exhibit broader-band spectra. At the same time, dispersion differs by event type: pull-request and push series are concentrated in mid-to-high centroid regions with moderate-to-high entropy, whereas stargazer series span a wider range from low-centroid/low-entropy to high-centroid/high-entropy regimes. This heterogeneity reinforces the motivation for `Impermanent`: evaluation should track performance sequentially over time rather than rely on a single static slice.

Figure 2: Weekly-series summary statistics for a random sample of 25 repositories, grouped by event type (columns). Each panel shows a relationship between two descriptors: log low/high bandpower ratio (slow vs fast variation) vs permutation entropy (irregularity), and spectral centroid (typical frequency) vs spectral entropy (spectral spread).



INFRASTRUCTURE

`Impermanent` runs as a set of serverless pipelines on Modal, with artefacts stored on Amazon S3. The pipeline has three stages. *Data ingestion* downloads hourly JSON archives from GH Archive, filters for the four target event types, aggregates per-repository counts with DuckDB, and rolls them up to the daily, weekly, and monthly granularities (subject to completeness thresholds of 90%, 95%, and 99% of constituent hours, respectively). *Forecasting* dispatches one job per `(model, cutoff)` pair: CPU models run on 32-core instances, foundation models on NVIDIA A10G GPUs, with up to 125 containers in parallel. *Evaluation* scores stored forecasts once ground truth arrives and rebuilds the leaderboard by reading all per-cutoff metric files, scaling by the zero-model baseline, and writing a single result Parquet. The full cycle is triggered weekly; every stage is idempotent, so re-runs skip completed work and new models can be added without reprocessing history.

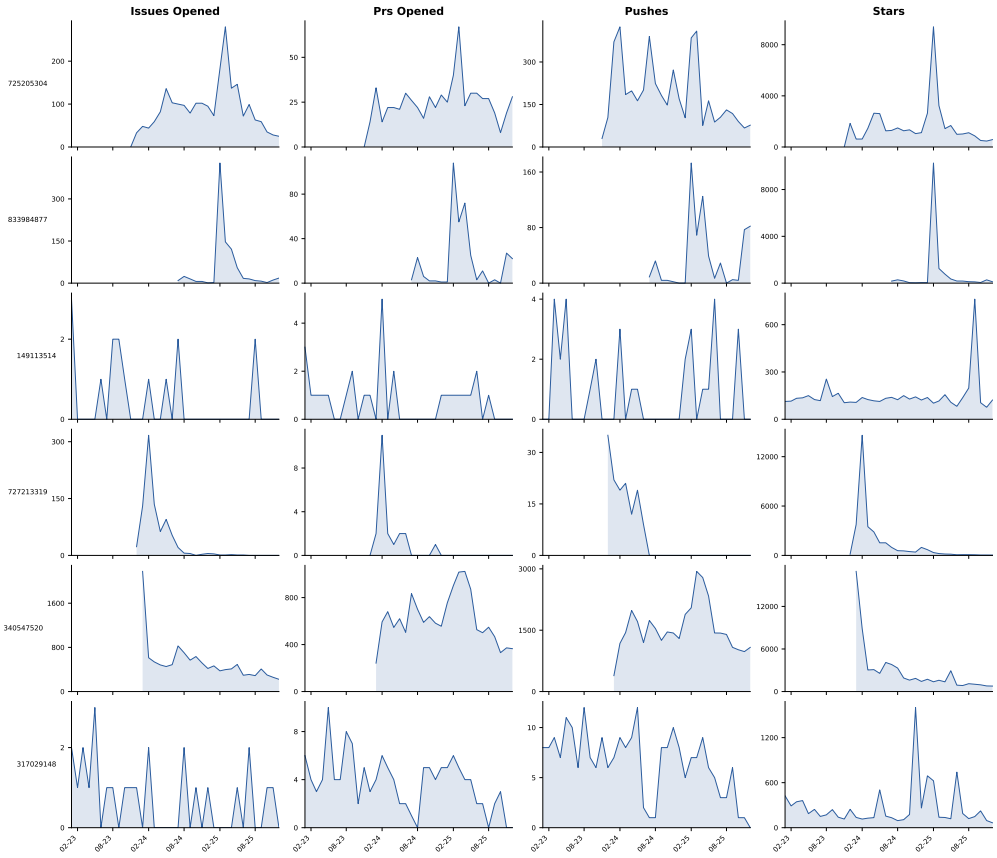


Figure 3: Monthly event counts for a random sample of six repositories across four activity signals. The series exhibit hallmark features of real-world temporal data: intermittency (long zero-count stretches), burstiness (isolated spikes), level shifts, and heterogeneous scales across repositories and event types.