The Journal of Open Source Software

aimz: Scalable probabilistic impact modeling

Eunseop Kim [1]

1 Eli Lilly and Company, United States

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Richard Liu

Reviewers:

- @DanWaxman
- @ankurankan

Submitted: 05 October 2025
Published: unpublished

Summary

`aimz` is a Python library for scalable probabilistic impact modeling, enabling assessment of intervention effects on outcomes while providing an intuitive interface for fitting Bayesian models, drawing posterior samples, generating large-scale posterior predictive simulations, and estimating interventional effects with minimal boilerplate. It combines the usability of general machine learning APIs with the flexibility of probabilistic programming through a single high-level object (`ImpactModel`). Built atop JAX (Bradbury et al., 2018) and NumPyro (Phan et al., 2019), it supports (minibatch) stochastic variational inference (SVI) and Markov chain Monte Carlo sampling, just-in-time (JIT)-compiled parallel predictive streaming to chunked Zarr (Miles et al., 2020) stores exposed through Xarray (Hoyer & Hamman, 2017), and first-class intervention handling for effect estimation. Integrated MLflow (Zaharia et al., 2018) support enables experiment tracking and model lineage. These design choices reduce bespoke glue code and enable reproducible, high-throughput analyses on large datasets, while supporting rapid iteration and experimentation.

Statement of need

Standard Bayesian workflows often encounter significant engineering friction when transitioning from model specification to large-scale impact evaluation. While core probabilistic programming frameworks offer mature inference algorithms, they lack native infrastructure for the repetitive engineering tasks essential to production-grade impact analysis: streaming predictive draws without exceeding system memory, managing device-aware sharding for simulation, and coordinating complex interventional scenarios. General machine learning libraries lack the calibrated uncertainty quantification provided by Bayesian sampling, while many causal inference toolkits focus on graph discovery rather than high-throughput predictive workflows. `aimz` addresses these gaps by consolidating model tracing, sharded predictive sampling, and experiment lineage into a single estimator-like object. This reduces the bespoke engineering effort required to connect flexible statistical research with high-throughput data pipelines, supporting reliable and reproducible probabilistic analyses at scale.

State of the field

The landscape of Bayesian software is largely divided between low-level flexibility and high-level rigidity. Core probabilistic programming languages like NumPyro, PyMC (Oriol et al., 2023), and Stan (Carpenter et al., 2017) provide the flexible primitives necessary for custom research but require users to manually handle scaling, parallelization, and other performance considerations. Conversely, domain-specific frameworks like Meridian (Google Meridian Marketing Mix Modeling Team, 2025), Robyn (Zhou et al., 2024), or PyMC-Marketing (PyMC Labs, 2025) can offer robust end-to-end pipelines but are specialized for marketing mix modeling. These tools frequently enforce fixed model architectures (e.g., specific adstock or saturation transformations) and opaque internal logic that are difficult to adapt for broader scientific research, such as evaluating clinical care gaps—where individual-level discrepancies between recommended best practices and actual care require custom hierarchical specifications and structural modifications.

`aimz` is designed to target a middle ground by providing Bayesian infrastructure in an estimator-like form. It does not prescribe a specific model form; instead, it provides a scalable execution layer for user-defined models. While libraries like CausalPy (PyMC Labs, 2026) provide sophisticated, high-level interfaces for a variety of quasi-experimental designs, they are typically optimized for exploratory analysis and causal identification. `aimz` distinguishes itself by focusing on the high-throughput engineering needed to efficiently handle large-scale probabilistic simulations and persist results as structured, reusable artifacts. This makes `aimz` uniquely suited for production-grade Bayesian workflows where reproducibility, experiment lineage, and the parallelized simulation of custom interventions are critical requirements.

Software design

`aimz` is designed to support scalable Bayesian analyses in applied settings, allowing users to iterate quickly across model specifications, inference settings, and intervention scenarios, as well as to handle large datasets and produce reproducible artifacts. The library is built on NumPyro to leverage its modeling flexibility and JAX's accelerator-native ecosystem and composable program transformations (e.g., JIT, vectorization, and sharded execution), which are well suited to impact modeling workflows where posterior predictive simulation and scenario evaluation often dominate overall runtime relative to model fitting.

`aimz` prioritizes performance as a key consideration. Predictive sampling and effect estimation are organized around JIT-accelerated execution, data-parallel sharding, and streaming results to disk-backed, chunked storage, enabling sustained throughput for large simulation workloads without requiring all draws to be retained in memory. This architectural choice comes with trade-offs: only models that are compatible with sharded execution can fully leverage these optimizations. Additional orchestration around execution and I/O is required, which `aimz` manages through a small set of high-level methods, enabling users to focus on analysis while maintaining a consistent interface. At the user interface level, `aimz` adopts an estimator-like API centered on the `ImpactModel` class, which takes a NumPyro model function as its "kernel" (the primary user-specified argument) and exposes familiar estimator methods while still supporting a wide range of model specifications.

`aimz` is also designed for integration: results are materialized as structured artifacts (e.g., Xarray objects backed by Zarr stores) and can optionally be logged via MLflow for experiment tracking and lineage. The combination of a stable, method-based interface and standardized outputs makes `aimz` well suited for AI-enabled workflows, where agentic tools can invoke a small set of operations and reliably consume the resulting artifacts.

Research impact statement

`aimz` enables a class of Bayesian modeling and probabilistic analyses that are otherwise costly to implement and difficult to reproduce at scale: including fitting flexible probabilistic models, generating large posterior predictive simulations, and estimating intervention effects under explicit structural modifications. Its research impact is therefore primarily infrastructural—lowering the engineering barrier to rigorous uncertainty-aware effect estimation—while remaining general enough to support diverse model specifications and inference strategies.

Within Eli Lilly and Company, `aimz` has supported internal analytics workflows that require scalable posterior and posterior predictive sampling, consistent output structures, and experiment traceability. In this setting, the library has reduced duplicated "glue" code for streaming predictive draws, coordinating intervention scenarios, and tracking model lineage, thereby improving iteration speed and reproducibility across analyses. Its stable estimator-like interface and structured, reproducible outputs also make it well suited for integration with AI-enabled workflows, where agentic tools can leverage standardized artifacts for downstream tasks.

Beyond immediate use, `aimz` is designed as reusable research infrastructure: it exposes arbitrary NumPyro model functions through a stable estimator-like interface and standardizes artifacts (samples, predictions, and effect estimates) in formats that integrate cleanly with the broader scientific Python ecosystem (e.g., Xarray/Zarr) and with experiment tracking via MLflow. This makes it easier for researchers and applied scientists to share models and results, compare alternative specifications, and operationalize workflows on larger datasets than would be practical with bespoke scripts. Early external adoption is reflected in package index downloads: since its initial public release in June 2025, `aimz` has been downloaded over 10,000 times via PyPI and conda-forge as of January 2026.

AI usage disclosure

Microsoft Copilot was used solely to assist with drafting code docstrings and adding type hints. All suggestions generated by Copilot were carefully reviewed and edited to ensure accuracy and consistency with the implemented functionality. No other generative AI tools were used in the development of the software, the writing of the manuscript, or the preparation of supporting materials.

Acknowledgements

The author acknowledges support from Eli Lilly and Company.

References

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*, 1–32.

Google Meridian Marketing Mix Modeling Team. (2025). *Meridian: Marketing mix modeling* (Version 1.2.1). https://github.com/google/meridian

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.148

Miles, A., Kirkham, J., Durant, M., Bourbeau, J., Onalan, T., Hamman, J., Patel, Z., shikharsg, Rocklin, M., dussin, raphael, Schut, V., Andrade, E. S. de, Abernathey, R., Noyes, C., sbalmer, bot, pyup.io, Tran, T., Saalfeld, S., Swaney, J., … Banihirwe, A. (2020). *Zarr-developers/zarr-python: v2.4.0* (Version v2.4.0). Zenodo. https://doi.org/10.5281/zenodo.3773450

Oriol, A.-P., Virgile, A., Colin, C., Larry, D., J., F. C., Maxim, K., Ravin, K., Jupeng, L., C., L. C., A., M. O., Michael, O., Ricardo, V., Thomas, W., & Robert, Z. (2023). PyMC: A modern and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, *9*, e1516. https://doi.org/10.7717/peerj-cs.1516

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv Preprint arXiv:1912.11554*.

PyMC Labs. (2025). *Marketing statistical models in PyMC* (Version 0.16.0). https://github.com/pymc-labs/pymc-marketing

PyMC Labs. (2026). *CausalPy: Causal inference for quasi-experiments in python* (Version 0.7.0). https://github.com/pymc-labs/CausalPy

Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, *41*, 39–45. https://api.semanticscholar.org/CorpusID:83459546

Zhou, G., Lares, B., Skokan, I., & Sentana, L. (2024). *Robyn: Semi-automated marketing mix modeling (MMM) from meta marketing science* (Version 3.12.0). https://doi.org/10.32614/cran.package.robyn