The Journal of Open Source Software

conjugate-models: Conjugate Models in Python

William Dean [1]

1 Independent Researcher

DOI: 10.21105/joss.09890

Software

- Review
- Repository
- Archive

Editor: Matt Graham

Reviewers:

- @aneeshnaik
- @GMarupilla

Submitted: 29 December 2025
Published: 29 April 2026

Summary

`conjugate-models` is a modern Python package for Bayesian conjugate inference that prioritizes a clean, idiomatic application programming interface (API) and seamless integration with widely used Python data analysis libraries. It implements the conjugate likelihood-prior pairs cataloged in Fink's compendium and Wikipedia's conjugate prior table, making rigorous Bayesian updating, exploration, and visualization accessible for practitioners, educators, and researchers. Comprehensive documentation and interactive examples support flexible application and learning.

Statement of Need

Bayesian inference with conjugate priors offers a tractable and interpretable approach to updating distributions in light of new evidence. While general-purpose probabilistic programming frameworks exist, they can introduce significant cognitive and computational overhead for common conjugate models. The `conjugate-models` package is designed to provide efficient, intuitive, and didactic support for Bayesian conjugate workflows across statistics, data science, and education, allowing direct use with array-like objects from libraries such as NumPy (Harris et al., 2020), pandas (The pandas development team, 2026), and Polars (Polars Contributors, 2024).

A prior distribution is conjugate to a likelihood when the posterior remains in the same distribution family after observing data (Raiffa & Schlaifer, 1961). Conjugate priors provide closed-form posterior updates and posterior predictive distributions, eliminating the need for numerical integration (Fink, 1997). Because these updates are analytic rather than iterative, posterior computation cost is independent of dataset size once summary statistics have been computed—enabling real-time interactive exploration and rapid model iteration. `conjugate-models` implements the conjugate pairs cataloged in Fink's compendium (Fink, 1997) and Wikipedia's conjugate prior table (Wikipedia contributors, 2026). The complete list of supported models is maintained at the online documentation.

Python lacked a dedicated, user-friendly package making Bayesian conjugate inference accessible, idiomatic, and didactically powerful—especially one offering robust integration with common data tooling and interactive resources for teaching and exploratory work. Existing educational tools for Bayesian statistics often lack interactivity or require complex setup, making the intuitive nature of conjugate priors difficult to convey to students and practitioners new to Bayesian methods.

Features & API Overview

`conjugate-models` provides an intuitive, pipeable API compatible with NumPy arrays (Harris et al., 2020), pandas DataFrames/Series (The pandas development team, 2026), Polars DataFrames (Polars Contributors, 2024) (for element-wise operations), and general numerical

types. The package includes vectorized and indexable operations for batch and multi-arm inference, built-in plotting for posterior, prior, and predictive distributions, and connection to SciPy distributions for interoperability (Virtanen et al., 2020).

A typical workflow follows a consistent pattern:

```
from conjugate.distributions import SomeDistribution
from conjugate.helpers import some_model_inputs
from conjugate.models import some_model

observed_data = ...

prior: SomeDistribution = ...
posterior: SomeDistribution = some_model(
    **some_model_inputs(observed_data),
    prior=prior,
)
```

Example Usage

Sequential Bayesian Updates

The package naturally supports sequential Bayesian learning, where each posterior becomes the next prior. For example, with a binomial model and beta prior, users can iteratively update beliefs as new trial data arrives, with each posterior distribution serving as the prior for the next update. This enables real-time learning applications where beliefs continuously evolve with incoming evidence. A complete worked example of sequential updating with beta-binomial conjugacy is available in the online documentation.

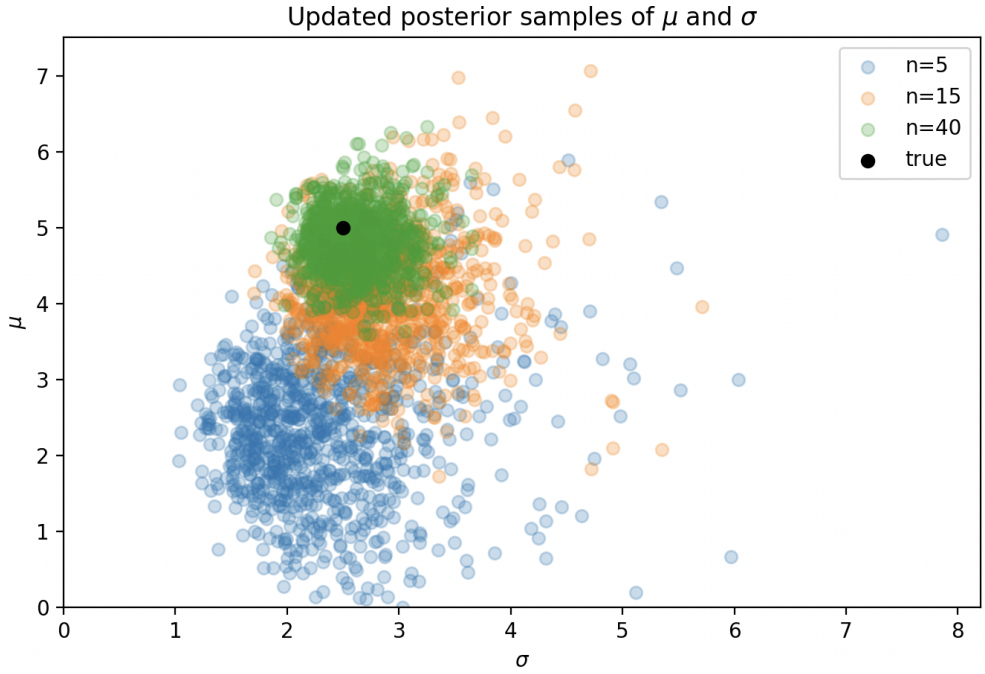

Figure 1: Sequential Bayesian updates showing prior, intermediate posterior, and final posterior distributions. Each update incorporates new evidence while maintaining uncertainty.

Thompson Sampling for Minimizing Wait Times

Thompson sampling is effective for exploration-exploitation problems where the goal is optimization. The package's vectorized operations and SciPy integration enable sophisticated applications such as multi-armed bandit problems, where posterior samples from conjugate updates guide exploration-exploitation decisions across multiple arms simultaneously. For instance, exponential-gamma conjugate pairs can model wait times across different service options, with Thompson sampling using posterior rate samples to balance exploration of uncertain options with exploitation of promising ones. A complete implementation demonstrating Thompson sampling with exponential-gamma conjugate updates for wait time minimization is available in the online documentation.

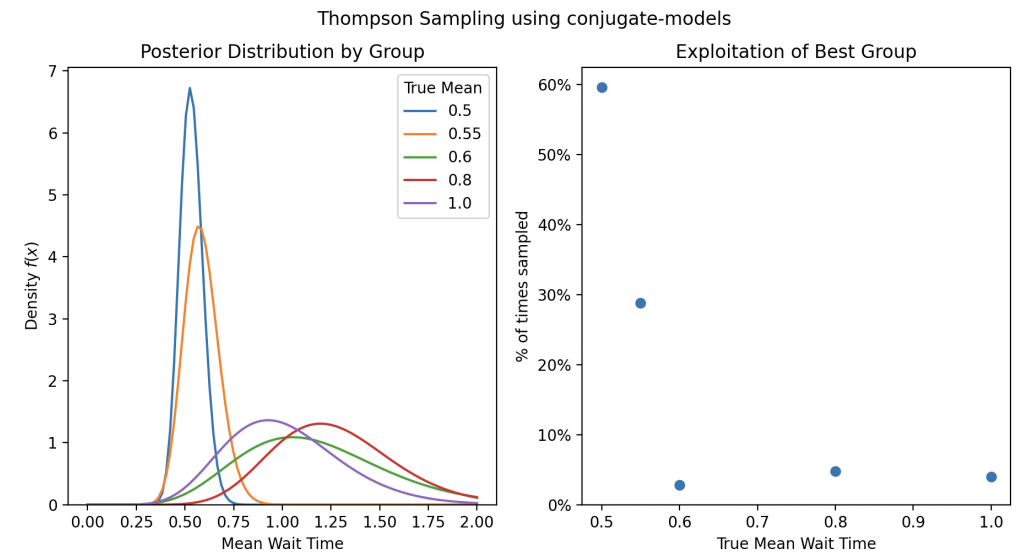

Figure 2: Thompson sampling results showing posterior distributions and exploitation rates. The algorithm successfully identifies and favors the group with the lowest wait time (highest rate), demonstrating effective exploration-exploitation balance.

Related Work

Several Python packages address aspects of Bayesian inference. ArviZ (Kumar et al., 2019) excels at posterior analysis and visualization but focuses on Markov chain Monte Carlo (MCMC)/variational inference outputs rather than conjugate models. Bambi (Capretto et al., 2022) provides a high-level interface for Bayesian linear models via PyMC but targets more complex model specifications. scikit-learn (Pedregosa et al., 2011) includes some Bayesian methods but emphasizes frequentist machine learning. General probabilistic programming languages like PyMC (Abril-Pla et al., 2023) and Stan (Stan Development Team, 2025) offer comprehensive modeling capabilities but introduce substantial overhead for simple conjugate updates.

`conjugate-models` distinguishes itself by wrapping `scipy.stats` distributions with conjugate update semantics, plotting interfaces, and seamless integration into standard Bayesian workflows (Gelman et al., 2020). Its focused scope enables immediate posterior computation without MCMC or optimization, making it ideal for interactive exploration, education, and rapid prototyping of conjugate models.

Limitations

This package specifically targets conjugate prior-likelihood pairs, so non-conjugate models require other tools like PyMC (Abril-Pla et al., 2023) or Stan (Stan Development Team, 2025). Model updates operate on sufficient statistics rather than raw data, requiring users to compute summary statistics beforehand. Distribution and parameter names follow established conventions from statistical literature (Fink, 1997), which may differ from other packages. Community support is available through GitHub Issues and Discussions.

Software Design

`conjugate-models` adopts a distribution-centric design where each probability distribution is a first-class object wrapping `scipy.stats` distributions while adding conjugate-specific functionality. Key design decisions include:

Compositional API: Rather than monolithic model classes, the package separates distributions (`conjugate.distributions`) from update logic (`conjugate.models`), allowing users to compose workflows naturally. This mirrors the mathematical structure where conjugate updates transform one distribution into another of the same family.

SciPy Integration: The `dist` property on each distribution class provides direct access to `scipy.stats` objects, enabling interoperability with the broader scientific Python ecosystem without reimplementing standard functionality like probability density function (PDF) computation, sampling, and statistical methods.

Vectorized Operations: Parameters accept array-like inputs (NumPy arrays, pandas/Polars columns), enabling batch inference for multi-arm problems like Thompson sampling without explicit loops. This design choice prioritizes performance for real-world applications involving multiple simultaneous models.

Mixin Architecture: Plotting capabilities are added via mixins (e.g., `ContinuousPlotDistMixin`, `DiscretePlotMixin`), keeping core distribution classes focused while enabling rich visualization. The `SliceMixin` provides indexing support for vectorized parameters.

Helper Function Design: The `helpers` module provides functions to extract sufficient statistics from raw observational data, bridging the gap between real-world datasets and the mathematical abstractions required for conjugate updates.

These design choices prioritize clarity and composability over abstraction, making the mathematical structure of conjugate inference explicit in the code while maintaining compatibility with the broader scientific Python ecosystem.

Research Impact Statement

`conjugate-models` addresses a specific gap in the Python ecosystem for lightweight, immediate Bayesian inference without MCMC overhead. Evidence of research impact includes:

Community Adoption: The package is available on PyPI with sustained development over 2+ years (270+ commits since June 2023), comprehensive test coverage (94%), and active maintenance demonstrated through regular releases and bug fixes.

Educational Value: The package's didactic design supports teaching Bayesian concepts through immediate, interactive feedback, making Bayesian inference more accessible to students and practitioners.

Documentation and Examples: Comprehensive documentation with 15+ worked examples demonstrates real-world applications from A/B testing to Thompson sampling, supporting both educational use and practical implementation.

Technical Contributions: The package implements the complete catalog of conjugate pairs from Fink's compendium (Fink, 1997) with modern Python practices, providing a reference implementation for conjugate Bayesian inference that was previously unavailable in a single, cohesive package.

Integration Capabilities: Seamless compatibility with NumPy, pandas, Polars, and SciPy enables integration into existing data science workflows without requiring users to learn new data structures or abandon familiar tools.

The package serves researchers, educators, and practitioners who need rapid, interpretable Bayesian inference for conjugate models, complementing rather than competing with general-purpose probabilistic programming frameworks.

Acknowledgments

We thank the scientific Python community, particularly the maintainers and contributors of NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), and Matplotlib, whose foundational libraries make this package possible. The examples showcase integration with pandas (The pandas development team, 2026), Polars (Polars Contributors, 2024), and PyMC (Abril-Pla et al., 2023), demonstrating the collaborative spirit of the open-source ecosystem.

AI Usage Disclosure

Generative AI tools were used during the development of this software and the preparation of this manuscript. All AI-assisted outputs were reviewed, edited, and validated by the human author, who made all core design decisions.

Software Development: GitHub Copilot Pro was used via the GitHub web interface for code suggestions and code review assistance. The majority of the codebase was written directly by the author without AI assistance.

Documentation: AI tools assisted with portions of the package documentation. The majority of documentation was written directly by the author.

Paper Authoring: opencode (version 1.0.220) with Claude Opus 4.5 was used to:

- Gather and organize information from existing documentation and the codebase during paper drafting
- Simulate peer review feedback via agent prompts to identify areas for improvement
- Iterate on paper structure, content, and clarity

Human Oversight: The author made all architectural and design decisions for the software, determined the scientific content and framing of the paper, and reviewed and validated all AI-assisted outputs before inclusion. No AI-generated code or text was included without human verification and editing.

Availability

`conjugate-models` is available on PyPI: `pip install conjugate-models`. The package is open-source under the MIT license at https://github.com/williambdean/conjugate, with contribution guidelines at CONTRIBUTING.md. Live documentation is available at https://williambdean.github.io/conjugate/.

References

Abril-Pla, O., Andreani, V., Carroll, C., Dong, L., Fonnesbeck, C. J., Kochurov, M., Kumar, R., Lao, J., Luhmann, C. C., Martin, O. A., Osthege, M., Vieira, R., Wiecki, T., & Zinkov, R. (2023). PyMC: A modern and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, *9*(e1516). https://doi.org/10.7717/peerj-cs.1516

Capretto, T., Piho, C., Kumar, R., Westfall, J., Yarkoni, T., & Martin, O. A. (2022). Bambi: A simple interface for fitting Bayesian linear models in Python. *Journal of Statistical Software*, *15*(103), 1–29. https://doi.org/10.18637/jss.v103.i15

Fink, D. (1997). *A compendium of conjugate priors*. Dalhousie University. https://courses.physics.ucsd.edu/2018/Fall/physics210b/REFERENCES/conjugate_priors.pdf

Gelman, A., Gabry, J., & Vehtari, A. (2020). Bayesian workflow. *arXiv Preprint arXiv:2011.01808*. https://arxiv.org/abs/2011.01808

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kumar, R., Carroll, C., Hartikainen, A., & Martin, O. (2019). ArviZ a unified library for exploratory analysis of Bayesian models in Python. *Journal of Open Source Software*, *4*(33), 1143. https://doi.org/10.21105/joss.01143

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Polars Contributors. (2024). *Polars: Fast DataFrame library*. https://doi.org/10.5281/zenodo.7697217

Raiffa, H., & Schlaifer, R. (1961). *Applied statistical decision theory*. Division of Research, Graduate School of Business Administration, Harvard University.

Stan Development Team. (2025). *Stan reference manual*. https://mc-stan.org

The pandas development team. (2026). *pandas-dev/pandas: Pandas* (Version v3.0.2). Zenodo. https://doi.org/10.5281/zenodo.19340003

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wikipedia contributors. (2026). *Conjugate prior*. https://en.wikipedia.org/wiki/Conjugate_prior