Bibat: Batteries-include Bayesian Analysis Template

Teddy Groves
Danish Technical University
tedgro@biosustain.dtu.dk

Abstract Bayesian statistical workflow offers a powerful way to learn from data, but software software projects that implement complex Bayesian workflows in practice are unusual, partly due to the difficulty of orchestrating Bayesian statistical software. Bibat addresses this challenge by providing a full-featured, scalable Bayesian statistical analysis project using an interactive template. Bibat is available on the Python Package index, documented at https://bibat.readthedocs.io/ and developed at https://github.com/teddygroves/bibat/. This paper explains the motivation for bibat, describes intended usage, discusses bibat's design, compares bibat with similar software, highlights several examples of bibat's use in science and provides links to community resources associated with bibat.

1 Introduction: the problem of orchestrating Bayesian workflow software

The term "Bayesian workflow" captures the idea that Bayesian statistical analysis comprises not just inference, but also specific approaches to related activities like data preparation, model design, diagnosis, debugging and criticism. This idea can be found in [1] and has recently received increasing scholarly recognition of [2]–[4]. Software tools now exist that address most individual aspects of a Bayesian workflow: see [5] for a review of the state of the art.

Unfortunately, each tool typically addresses one, or at most a few, of the many activities that comprise a real Bayesian workflow software project; it is left to the individual project team to orchestrate all of the tools they require. Writing software that performs this orchestration can be time-consuming and tricky, especially in the common scenario where it is not initially clear how many, or what kind of, statistical models, datasets, data manipulations or investigations an analysis will require.

Bibat is a new tool that addresses the difficulty of orchestrating Bayesian workflow software by providing a full-featured, high-quality project that can be extended to implement a wide range of statistical analyses.

2 How bibat works

Bibat can be installed on the Windows, Linux or macOS command line on by running the command `pip install bibat` and then used by running the command `bibat`. This command triggers an interactive form which prompts the user to select a range of customisation options. Bibat then creates a new directory containing code that implements an example analysis, with customisations reflecting the user's choices. This analysis works immediately, and can be reproduced with the single command `make analysis` without the need for any further action by the user: in this sense bibat comes with batteries included.

Figure 1 illustrates the components of a bibat-based Bayesian workflow and shows how it proceeds: the project team edits the code components, then runs `make analysis` triggering creation of the result components. After inspecting these they repeat the process, leading to a cycle that ultimately results in a complete, easily reproducible analysis.

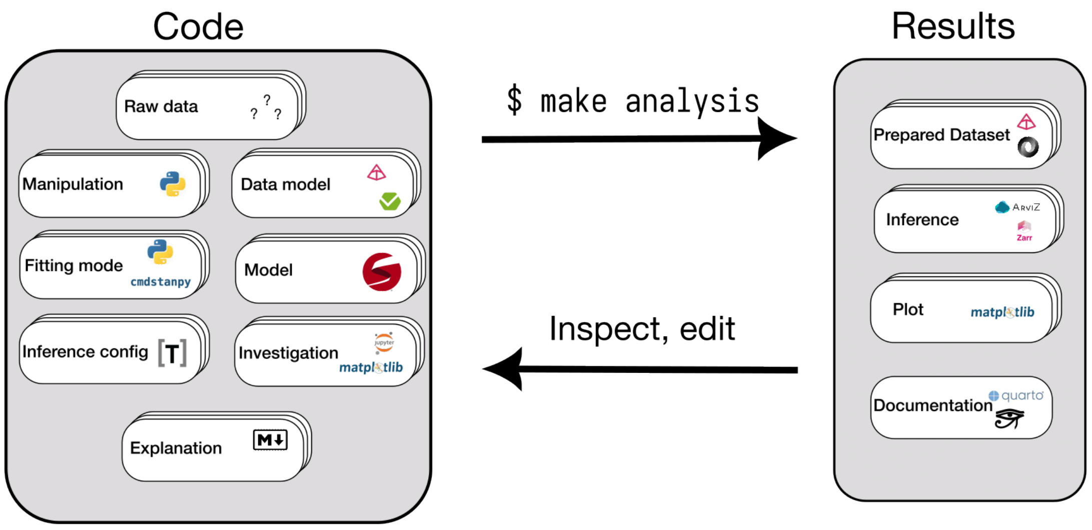

Figure 1: Schematic representation of a Bayesian workflow implemented using bibat. The author inspects their analysis's results, edits code corresponding to the boxes on the left, runs the command `make analysis`, then repeats. The diagram illustrates several key features of bibat: inference components are modular and plural, the overall workflow is iterative and cyclical and the whole analysis can be executed with a single command.

Bibat is documented at https://github.com/teddygroves/bibat/. The documentation website includes instructions for getting started, a detailed explanation of bibat's concepts and an extended vignette illustrating how to implement a complex statistical analysis starting from bibat's example analysis usage. In addition, the documentation site contains a full description of bibat's python API and command line interface, instructions for contributing and a section discussing accessibility considerations.

3 Design choices

Bibat's design was informed by the aims to accommodate the many sources of complexity and in a Bayesian workflow project, to ensure easy reproducibility and to integrate many open-source, widely-adopted and powerful Bayesian workflow tools and to encourage collaborative development.

As discussed in [2], Bayesian workflows are complicated, featuring plurality, cyclicity and complexity at many levels. As a specialised Bayesian workflow template, a key goal for bibat was to manage this complexity. Bibat achieves this aim by separating non-interacting analysis components into separate, potentially plural modules and by serialising data to files wherever possible. Prepared datasets, statistical models, inference configurations, inference results, plots and analyses all have file representations. Fitting modes, data manipulations and data models are modularised in code through the use of appropriately structured data classes and functions. Thanks to this modular approach it is possible to perform small sub analyses individually and to iteratively expand the analysis by adding components without needing to consider everything at once. In addition, bibat ensures that there are minimal restrictions on the components: for example, datasets need not be singular or tabular and statistical models need not be representable in formula syntax. Thanks to these accommodations a project team using bibat should typically not need to foresee the ultimate requirements of their analysis before starting the project.

Bibat encourages reproducibility mainly by providing a preconfigured makefile with a target `analysis` that triggers creation of an isolated environment, installation of dependencies, data preparation, statistical computation and analysis of results. In this way a bibat analysis can be reproduced on most platforms using a single command. In particular, this target attempts to install cmdstan if necessary, using a recipe tailored to the host operating system. This functionality addresses a common issue where researchers find it difficult to install Stan, especially on Windows. A second way in which bibat encourages reproducibility is by providing its Python code in the form of a preconfigured package. Thanks to the use of modern conventions for specifying dependencies and configuring tooling, we expect that this will make bibat analyses easier for other researchers to work with, thereby facilitating replication while also making initial development more convenient.

Bibat integrates many widely-adopted, open-source and powerful tools to implement the components of a Bayesian workflow. Bibat projects are written in modern Python and uses pydantic [6] and pandera [7] for data modelling, Stan [8] for statistical inference, cmdstanpy [9] for python-Stan interface, arviz [10] for storing and analyzing inferences and sphinx [11] and quarto [12] for documentation. Bibat itself is also written in modern Python and uses the previously mentioned software as well as the popular popular template tool copier [13].

To encourage collaborative development of Bayesian workflow projects, Bibat projects include a preconfigured test environment, continuous integration, linting and pre-commit hooks. In addition, including documentation as a first class component of the analysis addresses a common problem in academic statistics projects where the paper gets out of sync with the code. Bibat is continuously tested to ensure that it works on the operating systems Linux, macOS and Windows.

Bibat's continuous integration runs a test suite as well as an end-to-end functional test on all supported Python versions.

4 Fitting modes

The most novel part of bibat's design is the introduction of an abstraction called "fitting mode", which allows bibat project to handle different ways of fitting a model to a dataset appropriately and flexibly. This is often necessary as part of a Bayesian workflow: for example, one might perform MCMC sampling of both the prior and posterior distributions, perform multiple leave-out-one-fold fits for cross-validation or need to compare MCMC sampling with an optimisation-based alternative.

Fitting modes in bibat projects take the form of instances of the class `FittingMode`. Each fitting mode contains a name, a function that fits a prepared dataset and instructions for how and where to save the results. For example, the provided prior sampling fitting mode creates a Stan input dictionary with the `likelihood` data variable set to `0`, performs MCMC sampling and writes data to the `InferenceData` group `prior`. This design allows multiple fits to be stored alongside each other naturally, facilitating many common Bayesian workflow analyses, and can easily be configured to prevent unnecessary computations.

Bibat provides fitting modes corresponding for prior sampling, posterior sampling and k-fold posterior sampling. Users can easily add additional fitting modes by copying these examples, or modify the `FittingMode` class to achieve even richer functionality.

5 Comparison with alternative software

Other than bibat, there is currently no interactive template that specifically targets Bayesian workflow projects. There are some templates that arguably encompass Bayesian workflow as a special case of data analysis project, such as cookiecutter-data-science [14], but these are of limited use compared with a specialised template due to the many specificities of Bayesian workflow.

There is some software that addresses the general task of facilitating Bayesian workflow, but using a different approach from bibat's. For example, bambi [15] and brms [16] aim to make implementing Bayesian workflows easier by providing ergonomic ways to specify and fit Bayesian regression models to tabular datasets. Bibat is complementary with these packages, as it targets use cases that they do not support, such as analyses where complex datasets or custom models might be required.

6 Case studies

The following cases illustrate how bibat has been used in practice to facilitate Bayesian workflow projects.

[17] used bibat to compare a Bayesian and two non-Bayesian approaches to modelling a biochemical thermodynamics dataset. Bibat facilitated this analysis even though it was not very large—the final analysis contained one dataset, three models and three inferences—because the fitting mode abstraction allowed for straightforward comparison of the different methods. Additionally, bibat

made it easier to iteratively investigate and discard models that did not form part of the final analysis.

In [18], Bibat was used to implement a sports analysis involving two datasets, two models and four inferences, demonstrating that the generalised Pareto distribution can be used to describe hitting ability in baseball. This analysis is now included in bibat as an illustration, along with an accompanying tutorial. An illustrative graphic from this analysis is shown in Figure 2.

In this case bibat was useful because of its ability to implement arbitrary statistical models, as latent generalised Pareto distributions are not supported by generic regression packages like brms and bambi. Further, bibat's modular design made it easier to implement this medium-sized analysis with two datasets, two models and six inferences.

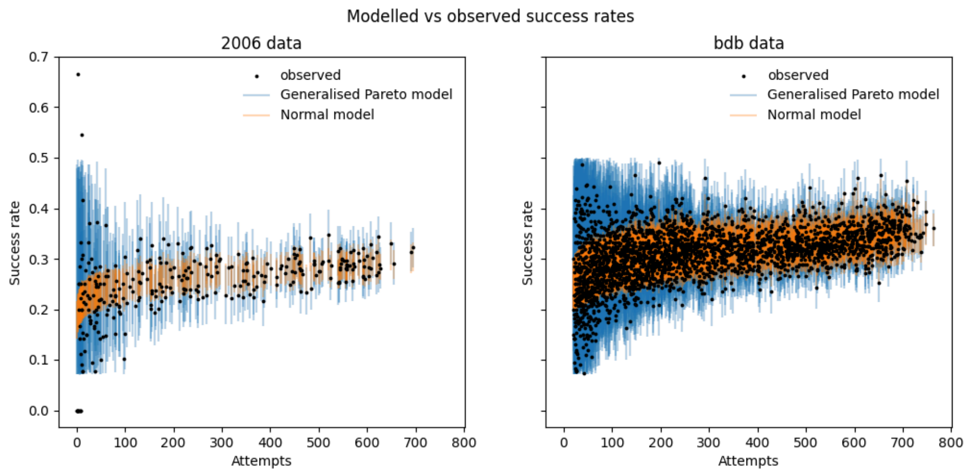

Figure 2: A graphical posterior predictive check produced as part of a bibat analysis that fit two statistical models to two datasets of baseball data. The coloured lines show each model's posterior predictive distributions and the black dots show the two observed datasets. See https://github.com/teddygroves/bibat/tree/main/bibat/examples/baseball for the full analysis.

In [19], bibat was used to implement a large analysis of cerebrovascular data from mice, involving two raw datasets, 6 prepared datasets, 15 models and 15 inferences. Again it was not possible to implement the analysis using formula-based packages due to the need for custom modelling approaches to take into account expert knowledge about both the target system and the measurement apparatus. Due to the number of datasets and models that needed to be considered, implementing this analysis without using bibat or a similar structured approach would have required a combination of more time and/ or compromises in software quality.

These cases illustrate that bibat can be useful in a variety of real Bayesian workflows, with different sizes, subject matters and emphases.

7 Community

Bibat is developed in public and encourages community contribution. Please see the contributing page https://github.com/teddygroves/bibat/blob/main/CONTRIBUTING.md and code of conduct https://github.com/teddygroves/bibat/blob/main/CODE_OF_CONDUCT.md if you would like to help develop bibat.

Bibat has a growing user community, with 16 GitHub stars at the time of writing, and is affiliated with cmdstanpy through a link on its documentation website. Bibat is also affiliated with the Python scientific software community PyOpenSci, allowing for help with maintenance as well as peer review for code and documentation quality, usability and accessibility. The PyOpenSci peer review for bibat can be found here: https://github.com/pyOpenSci/software-submission/issues/83.

8 Author contributions

9 Acknowledgements

Bibliography

[1] G. E. P. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis*, Wiley classics library ed. in A Wiley-Interscience Publication. New York: Wiley, 1992.

[2] A. Gelman *et al.*, "Bayesian Workflow", *arXiv:2011.01808 [stat]*, Nov. 2020, Accessed: Nov. 05, 2020. [Online]. Available: http://arxiv.org/abs/2011.01808

[3] L. Grinsztajn, E. Semenova, C. C. Margossian, and J. Riou, "Bayesian Workflow for Disease Transmission Modeling in Stan", *Statistics in Medicine*, vol. 40, no. 27, pp. 6209–6234, 2021, doi: 10.1002/sim.9164.

[4] J. Gabry, D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman, "Visualization in Bayesian Workflow", *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 182, no. 2, pp. 389–402, Feb. 2019, doi: 10.1111/rssa.12378.

[5] E.ˇStrumbelj *et al.*, "Past, Present, and Future of Software for Bayesian Inference", 2024.

[6] Pydantic developers, "Pydantic". [Online]. Available: https://pypi.org/project/pydantic/

[7] Niels Bantilan, "Pandera: Statistical Data Validation of Pandas Dataframes", in *Proceedings of the 19th Python in Science Conference*, M. Agarwal, C. Calloway, D. Niederhut, and David Shupe, Eds., 2020, pp. 116–124. doi: 10.25080/Majora-342d178e-010.

[8] B. Carpenter *et al.*, "Stan: A Probabilistic Programming Language", *Journal of Statistical Software*, vol. 76, no. 1, pp. 1–32, Jan. 2017, doi: 10.18637/jss.v076.i01.

[9] Stan Development Team, "CmdStanPy". [Online]. Available: https://github.com/stan-dev/cmdstanpy

[10] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, "ArviZ a Unified Library for Exploratory Analysis of Bayesian Models in Python", *Journal of Open Source Software*, vol. 4, no. 33, p. 1143, Jan. 2019, doi: 10.21105/joss.01143.

[11] Georg Brandl and the Sphinx team, "Sphinx". [Online]. Available: https://www.sphinx-doc.org/

[12] J. Allaire, C. Teague, C. Scheidegger, Y. Xie, and C. Dervieux, “Quarto”. [Online]. Available: https://github.com/quarto-dev/quarto-cli

[13] copier developers, “Copier”. Accessed: Feb. 15, 2024. [Online]. Available: https://github.com/copier-org/copier

[14] Driven Data, “Cookiecutter-Data-Science”. [Online]. Available: https://github.com/drivendata/cookiecutter-data-science/

[15] T. Capretto, C. Piho, R. Kumar, J. Westfall, T. Yarkoni, and O. A. Martin, “Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python”. 2020.

[16] P.-C. Bürkner, “Brms: An R Package for Bayesian Multilevel Models Using Stan”, *Journal of Statistical Software*, vol. 80, no. 1, pp. 1–28, 2017, doi: 10.18637/jss.v080.i01.

[17] T. Groves and J. Jooste, “Dgfreg”. Accessed: Jan. 29, 2024. [Online]. Available: https://github.com/biosustain/dgfreg

[18] T. Groves, “Baseball”. Accessed: Jan. 29, 2024. [Online]. Available: https://github.com/teddygroves/baseball

[19] T. Groves, “Sphincter”. Accessed: Jan. 29, 2024. [Online]. Available: https://github.com/teddygroves/sphincter