Bibat: Batteries-include Bayesian Analysis Template

Teddy Groves[1]

[1]The Novo Nordisk Foundation Center for Biosustainability, DTU, Denmark

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/`. Bibat is free to use under the MIT license. 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 Box and Tiao (1992) and has recently received increasing scholarly recognition (Gelman et al. 2020; Grinsztajn et al. 2021; Gabry et al. 2019). Software tools now exist that address most individual aspects of a Bayesian workflow: see Strumbelj et al. (2024) 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

In order to use bibat, a user must first install the templating library copier (copier developers 2024), then choose a directory name and run this command from the command line:

```
$ copier copy gh:teddygroves/bibat my-chosen-directory-name
```

This command triggers an interactive form which prompts the user to select a range of customisation options. The new directory will then be created if necessary and filled with 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

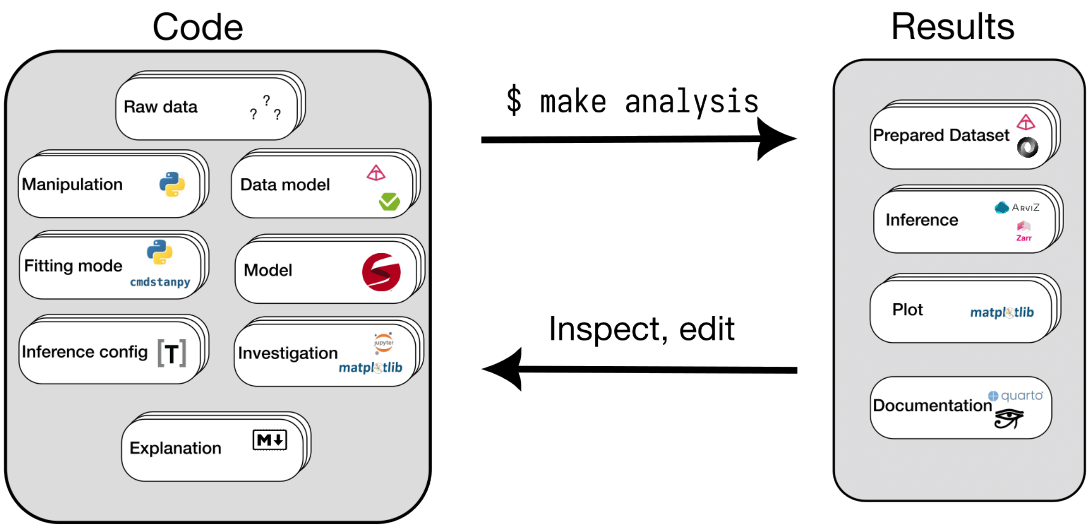

Figure 1: Schematic representation of a Bayesian workflow implemented using bibat. The author inspects their analysis's results, edits code corresponding to the boxes on the left, runs the command `make analysis`, then repeats. The diagram illustrates several key features of bibat: inference components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

of the result components. After inspecting these they repeat the process, leading to a cycle that ultimately results in a complete, easily reproducible analysis.

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 Gelman et al. (2020), 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 it is possible to use any statistical model that

Stan can compile. 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 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 a preconfigured Python project following modern conventions, making bibat analyses straightforward to replicated and extend for other researchers who are already familiar with these conventions.

Bibat integrates many widely-adopted open-source tools to implement the components of a Bayesian workflow. These include pydantic (Pydantic developers 2022) and pandera (Niels Bantilan 2020) for data modelling, Stan (Carpenter et al. 2017) for statistical inference, cmdstanpy (Stan Development Team 2022) for python-Stan interface, arviz (Kumar et al. 2019) for storing and analysing inferences and sphinx (Georg Brandl and the Sphinx team 2022) and quarto (Allaire et al. 2022) for documentation.

To encourage collaborative development of Bayesian workflow projects, Bibat projects include a preconfigured test environment, continuous integration, linting and pre-commit hooks. In addition, bibat includes documentation as a first class, integrated component of the analysis, helping to keep it in sync with the other components.

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 projects to handle fitting a model to a dataset in different ways. This is often necessary as part of a Bayesian workflow: for example, one might perform MCMC sampling of both the prior and posterior distributions, or perform multiple leave-out-one-fold fits for cross-validation, or compare MCMC-based posterior inference with an alternative computation method.

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 is called "prior", contains a function that runs MCMC sampling with the `likelihood` input variable set to `0`, returning a `CmdStanMCMC` instance, and specifies that this result should be written to the `InferenceData` group `prior`. Bibat provides fitting modes corresponding for prior sampling, posterior sampling and k-fold posterior sampling. Users can easily implement additional fitting modes or modify the `FittingMode` class to achieve even richer functionality. Fitting modes can be referenced by name from the file that configures an inferece: for example, the following lines indicate that the inference should be run in prior, posterior and kfold modes:

```
modes = ["prior", "posterior", "kfold"]
```

Fitting modes allow bibat projects to succinctly but flexibly declare how to perform inferences, and allow results corresponding to the same inference to be stored alongside each other appropriately.

5 Comparison with alternative software

Other than bibat, we are not aware of any 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 (Driven Data 2022), but these are of limited use compared with a specialised template due to the many specificities of Bayesian workflow. cookiecutter-cmdstanpy-wrapper (Ward 2024) is an interactive template that targets a different use case than Bayesian workflow projects, namely setting up a Python package that provides pre-compiled Stan models.

There is some software that addresses the general task of facilitating Bayesian workflow, but using a different approach from bibat's. For example, bambi (Capretto et al. 2020) and brms (Bürkner 2017) 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 Limitations

Using bibat effectively requires familiarity with Pydantic, pandera, arviz, Stan and managing a medium-sized Python project. Many statistical analysis projects do not require using these tools, for example if data preparation or validation is trivial, if custom statistical models are not required, or if the analysis can be carried out by a single script. Practitioners who wish to implement such Bayesian workflows may prefer to simply write their software from scratch rather than use bibat, using tools like bambi or brms to ensure that the software challenge remains manageable.

Similarly, bibat accommodates plural inferences, fitting modes and datasets, but many analyses are singular in at least one of these components and could therefore be implemented more simply and concisely than an equivalent bibat project. On the other hand, it is typically difficult to predict in advance which components of a Bayesian workflow will be plural, and costly to re-write a project after mistakenly assuming that a component will be singular. While we acknowledge that accommodating potentially unneeded plurality is an important limitation of bibat, we nonetheless think that it is the correct choice for a general-purpose template.

Another limitation of bibat is that it makes many opinionated choices about which tools to use. In particular, languages other than Python, inference frameworks other than Stan and validation frameworks other than Pydantic are not supported. We think that it is on the whole good for templates to be opinionated, as unopinionated templates are necessarily more complicated; this limitation of bibat is therefore best addressed by the development of additional analysis templates that make different choices.

7 Case studies

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

Groves and Jooste (2023) 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 Groves (2022), 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 any available formula-based regression packages. Further, bibat's modular design made it easier to implement this medium-sized analysis with two datasets, two models and six inferences.

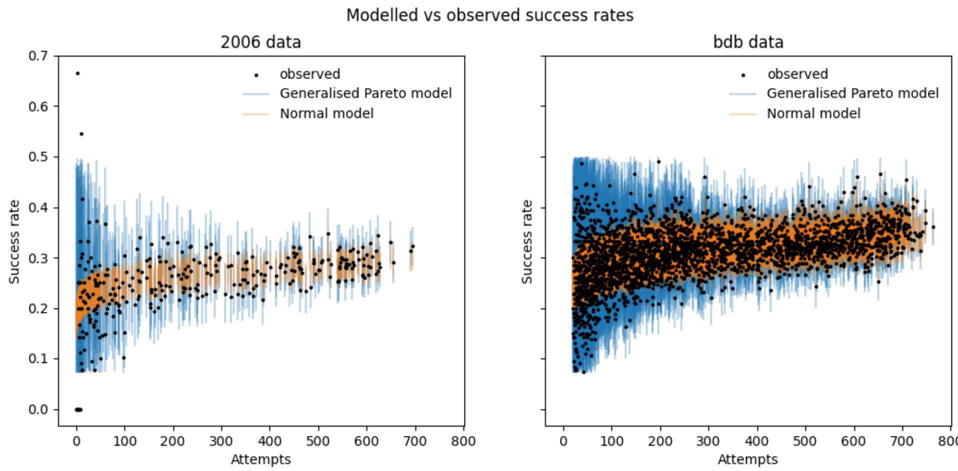

Figure 2: A graphical posterior predictive check produced as part of a bibat analysis that fit two statistical models to two datasets of baseball data. The coloured lines show each model's posterior predictive distributions and the black dots show the two observed datasets. See `https://github.com/teddygroves/bibat/tree/main/bibat/examples/baseball`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

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

8 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`.

9 Broader impact statement

After careful reflection, the authors have determined that this work presents no notable negative impacts to society or the environment.

Allaire, J. J., Charles Teague, Carlos Scheidegger, Yihui Xie, and Christophe Dervieux. 2022. "Quarto." `https://doi.org/10.5281/zenodo.5960048`.

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

Bürkner, Paul-Christian. 2017. "Brms: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80 (1): 1–28. `https://doi.org/10.18637/jss.v080.i01`.

Capretto, Tomás, Camen Piho, Ravin Kumar, Jacob Westfall, Tal Yarkoni, and Osvaldo A. Martin. 2020. "Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python." https://arxiv.org/abs/2012.10754.

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1): 1–32. https://doi.org/10.18637/jss.v076.i01.

copier developers. 2024. "Copier." copier-org. https://github.com/copier-org/copier.

Driven Data. 2022. "Cookiecutter-Data-Science." https://github.com/drivendata/cookiecutter-data-science/.

Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman. 2019. "Visualization in Bayesian Workflow." *Journal of the Royal Statistical Society Series A: Statistics in Society* 182 (2): 389–402. https://doi.org/10.1111/rssa.12378.

Gelman, Andrew, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. 2020. "Bayesian Workflow." *arXiv:2011.01808 [Stat]*, November. http://arxiv.org/abs/2011.01808.

Georg Brandl and the Sphinx team. 2022. "Sphinx." https://www.sphinx-doc.org/.

Grinsztajn, Léo, Elizaveta Semenova, Charles C. Margossian, and Julien Riou. 2021. "Bayesian Workflow for Disease Transmission Modeling in Stan." *Statistics in Medicine* 40 (27): 6209–34. https://doi.org/10.1002/sim.9164.

Groves, Teddy. 2022. "Baseball." https://github.com/teddygroves/baseball.

Groves, Teddy, and Jason Jooste. 2023. "Dgfreg." DTU Biosustain. https://github.com/biosustain/dgfreg.

Kumar, Ravin, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. 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.

Niels Bantilan. 2020. "Pandera: Statistical Data Validation of Pandas Dataframes." In *Proceedings of the 19th Python in Science Conference*, edited by Meghann Agarwal, Chris Calloway, Dillon Niederhut, and David Shupe, 116–24. https://doi.org/10.25080/Majora-342d178e-010.

Pydantic developers. 2022. "Pydantic." https://pypi.org/project/pydantic/.

Stan Development Team. 2022. "CmdStanPy." https://github.com/stan-dev/cmdstanpy.

Strumbelj, Erik, Alexandre Bouchard-Côté, Jukka Corander, Andrew Gelman, Håvard Rue, Lawrence Murray, Henri Pesonen, Martyn Plummer, and Aki Vehtari. 2024. "Past, Present and Future of Software for Bayesian Inference." *Statistical Science* 39 (1): 46–61.

Ward, Brian. 2024. "WardBrian/Cookiecutter-Cmdstanpy-Wrapper." https://github.com/WardBrian/cookiecutter-cmdstanpy-wrapper.

Submission Checklist

1. For all authors…
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section "Limitations"
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] I can't think of any particular negative societal impacts of a Bayesian workflow template
   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes]
2. If you ran experiments…
   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [N/A]
   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [N/A]
   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [N/A]
   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [N/A]
   (e) Did you report the statistical significance of your results? [N/A]
   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A]
   (g) Did you compare performance over time and describe how you selected the maximum duration? [N/A]
   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
   (i) Did you run ablation studies to assess the impact of different components of your approach? [N/A]
3. With respect to the code used to obtain your results…
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] See `https://bibat.readthedocs.io/en/latest/_static/report.html` for instructions to reproduce the main example.
   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] See section "Generating Stan inputs" here `https://bibat.readthedocs.io/en/latest/_static/report.html#preparing-the-data`
   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] See pyopensci review `https://github.com/pyOpenSci/software-submission/issues/83`
   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [No] This is unnecessary as the results are easy to reproduce and the results are large files that would be awkward to store online

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [No] To reproduce the figures from raw data run "make analysis" from the folder `example_projects/baseball`, as described in the instructions here `https://bibat.readthedocs.io/en/latest/_static/report.html`

4. If you used existing assets (e.g., code, data, models)...

   (a) Did you cite the creators of used assets? [Yes]

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [N/A]

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you created/released new assets (e.g., code, data, models)...

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes]

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] `https://github.com/teddygroves/bibat/`

6. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]