Bibat: Batteries-include Bayesian Analysis Template

Teddy Groves
Danish Technical University
Kongens Lyngby, Denmark

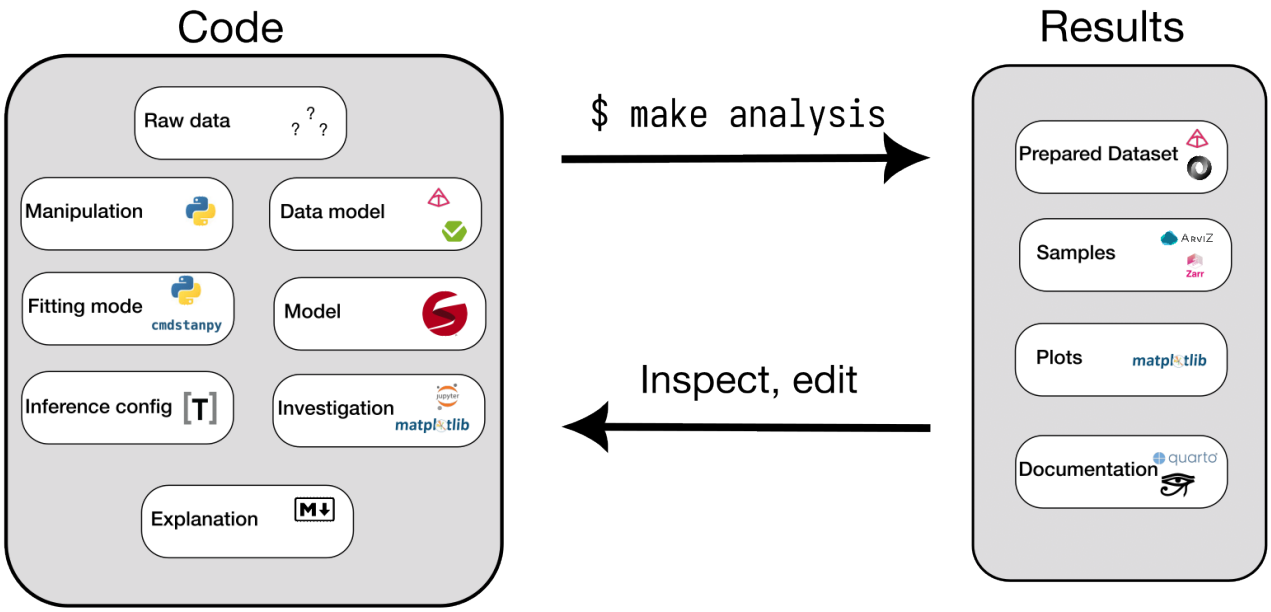

Figure 1: Schematic representation of a Bayesian workflow implemented using bibat. The author inspects their analysis's results, edits files corresponding to the boxes on the left, runs the command `make analysis`, then repeats. Note that this workflow is modular, accommodates plurality and results in a final analysis that can be fully reproduced using a single command.

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, briefly describes intended usage, discusses key design choices, and reviews several examples of bibat's use in scientific applications.

KDD, Aug 25–29, 2024, Barcelona, Esp

KEYWORDS

Python, Bayesian workflow, Methodology

ACM Reference Format:
Teddy Groves. 2024. Bibat: Batteries-include Bayesian Analysis Template. In *Proceedings of KDD.* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION: THE PROBLEM OF ORCHESTRATING BAYESIAN WORKFLOW SOFTWARE

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, dates at least to Box and Tiao [2]. There is increasing scholarly recognition of the need for a holistic view of "Bayesian workflow" [8, 9, 12]. Relatedly, software tools now exist that address most individual aspects of a Bayesian workflow: see Štrumbelj et al. [20] for a review of the state of the art.

Unfortunately, currently available tools typically address one, or at most a handful, of Bayesian workflow activities; it is left to the individual project team to orchestrate the components. 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 addresses this difficulty by providing a full-featured, high-quality Bayesian workflow project that can be extended to implement a wide range of statistical analyses.

2 HOW BIBAT WORKS

2.1 Installation and usage

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".

2.2 Documentation

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 these considerations.

3.1 Accommodate a wide range of statistical analyses

Bibat projects explicitly allow for plurality at the level of input datasets, prepared datasets, statistical models, fitting modes, computation methods and analyses. In addition, bibat ensures that there are minimal restrictions on the kind of 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.

3.2 Encourage reproducibility

Bibat provides a preconfigured makefile with a target `analysis` triggering 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.

Bibat also provides its Python code in the form of a package configured using modern conventions for specifying dependencies and configuring tooling, so that it is easy to maintain reproducibility as the analysis develops.

3.3 Use widely-adopted, open source and actively developed tools

Bibat projects are is written in modern Python and uses pydantic Pydantic developers [18] and pandera [17] for data modelling, Stan [5] for statistical inference, cmdstanpy [19] for python-Stan interface, arviz [16] for storing and analyzing inferences and sphinx [10] and quarto [1] for documentation.

Bibat itself is also written in modern Python and uses the popular tools cookiecutter [11], pydantic and click [6].

Bibat's makefile detects the current operating system and attempts to install cmdstan appropriately if necessary. This functionality addresses a common issue where researchers find it difficult to install Stan, especially on Windows.

3.4 Implement community standards and best practices for collaborative software development

Bibat projects include a preconfigured test environment, continuous integration, linting and pre-commit hooks, making them suitable for collaborative software development. 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 itself 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.

Bibat is part of the PyOpenSci ecosystem, allowing for community 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

4 CHALLENGES

This section describes some specific challenges that often affect Bayesian workflow projects and how bibat addresses them.

4.1 Address complexity using a modular, file based approach

As discussed in Gelman et al. [9], Bayesian workflows are complicated, featuring plurality, cyclicity and complexity at many levels. Bibat accommodates this by separating non-interacting analysis components into 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 approach it is possible to perform small sub analyses individually and to iteratively add components without needing to consider everything at once.

4.2 Fitting modes

As part of a Bayesian workflow it can be necessary to fit a model and dataset in different ways. 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.

Bibat accommodates this need by introducing an abstraction called "fitting mode". This abstraction allows bibat projects to handle fitting a model and dataset in different ways appropriately and flexibly. 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`. 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.

5 POST LAUNCH USE

Bibat has been used for several real statistical analyses.

Groves and Jooste [15] 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 by providing the fitting mode abstraction, which was very useful for comparing the different methods.

In Groves [13], 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.

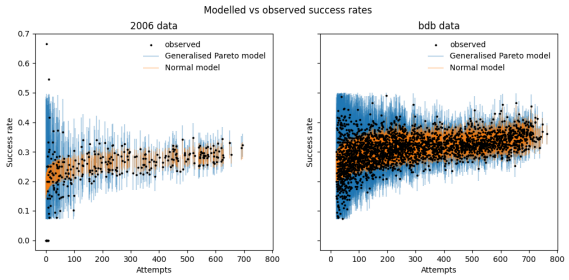

Figure 2: A graphical posterior predictive check produced as part of a bibat analysis that fit two statistical models to two datasets of baseball data. The coloured lines show each model's posterior predictive distributions and the black dots show the two observed datasets. See https://github.com/teddygroves/bibat/tree/main/bibat/examples/baseball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

In Groves [14], bibat was used to implement an analysis of cerebrovascular data from mice, involving two raw datasets, 6 prepared datasets, 15 models and 15 inferences. In this case bibat's modular design made it straightforward to iteratively add data transformations and models while maintaining reproducibility.

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

Bibat has a growing user community, with 16 GitHub stars at the time of writing.

6 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 [7], 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 [4] and brms [3] 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.

7 STATEMENTS AND DECLARATIONS

The authors have no financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

REFERENCES

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

[2] George E. P. Box and George C. Tiao. 1992. *Bayesian Inference in Statistical Analysis* (wiley classics library ed ed.). Wiley, New York.

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

[4] Tomás Capretto, Camen Piho, Ravin Kumar, Jacob Westfall, Tal Yarkoni, and Osvaldo A. Martin. 2020. Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python. arXiv:2012.10754 [stat.CO]

[5] Bob Carpenter, 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 (Jan. 2017), 1–32. https://doi.org/10.18637/jss.v076.i01

[6] Click Developers. 2022. Click: Python Composable Command Line Interface Toolkit. Pallets. https://pypi.org/project/click/

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

[8] Jonah Gabry, 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 (Feb. 2019), 389–402. https://doi.org/10.1111/rssa.12378

[9] Andrew Gelman, 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]* (Nov. 2020). arXiv:2011.01808 [stat] http://arxiv.org/abs/2011.01808

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

[11] Audrey Roy Greenfeld, Dainiel Roy Greenfeld, Raphael Pierzina, and others. 2021. Cookiecutter. https://pypi.org/project/cookiecutter/

[12] Léo Grinsztajn, Elizaveta Semenova, Charles C. Margossian, and Julien Riou. 2021. Bayesian Workflow for Disease Transmission Modeling in Stan. *Statistics in Medicine* 40, 27 (2021), 6209–6234. https://doi.org/10.1002/sim.9164

[13] Teddy Groves. 2022. Baseball. https://github.com/teddygroves/baseball

[14] Teddy Groves. 2024. Sphincter. https://github.com/teddygroves/sphincter

[15] Teddy Groves and Jason Jooste. 2023. Dgfreg. DTU Biosustain. https://github.com/biosustain/dgfreg

[16] Ravin Kumar, 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 (Jan. 2019), 1143. https://doi.org/10.21105/joss.01143

[17] Niels Bantilan. 2020. Pandera: Statistical Data Validation of Pandas Dataframes. In *Proceedings of the 19th Python in Science Conference*, Meghann Agarwal, Chris Calloway, Dillon Niederhut, and David Shupe (Eds.). 116–124. https://doi.org/10.25080/Majora-342d178e-010

[18] Pydantic developers. 2022. Pydantic. https://pypi.org/project/pydantic/

[19] Stan Development Team. 2022. CmdStanPy. https://github.com/stan-dev/cmdstanpy

[20] Erik Štrumbelj, 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. (2024).