Bibat: Batteries-include Bayesian Analysis Template

Teddy Groves

Despite their suitability for many scientific problems and the existence of sound theoretical and computational frameworks, adoption of Bayesian workflow in science remains a challenge. One reason for the difficulty is that it is difficult to write software that implements non-trivial Bayesian workflows. We aimed to address this difficulty by developing Bibat, a Python package providing an interactive template that targets Bayesian statistical analysis projects. Bibat trivialises the otherwise painful process of structuring such a project and integrating together tools that help with individual parts of a Bayesian workflow, without compromising flexibility, scalability, interoperability or reproducibility. Bibat is available on the Python Package index, documented at https://bibat.readthedocs.io/ and developed at https://github.com/teddygroves/bibat/.

Many areas of science would benefit from replacing traditional statistical methodologies with Bayesian workflow, as described in, for example (Gelman et al. 2020; Grinsztajn et al. 2021; Gabry et al. 2019).

There are now software tools that address most individual aspects of a Bayesian workflow, from fetching and manipulating data to specifying, computing, storing, inspecting and documenting statistical inferences. However, writing a software project implementing a Bayesian workflow remains challenging due partly to the difficulty of orchestrating and configuring these elements.

Bibat is a Python package that addresses this difficulty by providing an interactive template for Bayesian workflow projects.

1 Installation and usage

Bibat is installed 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 for configuration information including project and repository name, author name, a short description and choices of open source licence options, documentation formats and whether to include tests and continuous integration.

Bibat then creates a folder with the chosen repository name in the current working directory, containing code that implements bibat's example analysis.

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, an extended vignette illustrating intended usage, full description of the python API and command line interface, instructions for contributing and a section discussing accessibility considerations.

3 Software

Bibat is implemented using Python 3. Its main dependencies are cookiecutter (Greenfeld et al. 2021), pydantic (Pydantic developers 2022) and click (Click Developers 2022). Bibat is continuously tested to ensure that it works on the operating systems Linux, macOS and Windows.

Bibat projects use the following tools:

- Python 3 (Van Rossum and Drake 2009) and standard scientific Python packages for data manipulation
- Stan (Carpenter et al. 2017) for specifying statistical models and performing inferences
- cmdstanpy (Stan Development Team 2022) for interfacing between Python and Stan
- arviz (Kumar et al. 2019) for storing and analysing completed inferences
- pydantic and pandera (Niels Bantilan 2020) for validation
- make (Stallman and McGrath 1991) for automation.
- Sphinx (Georg Brandl and the Sphinx team 2022) and Quarto (Allaire et al. 2022) for documentation.

4 How bibat addresses specific Bayesian workflow difficulties

This section describes some specific problems that often affect Bayesian workflow projects and which bibat helps to ameliorate.

4.1 Data modelling

Any statistical analysis requires a definition of a prepared dataset, whether implicitly or explicitly. Ideally the definition is explicit and allows for arbitrarily many prepared datasets, accommodates both tabular and non-tabular formats and provides functionality like validation and serialisation.

Bibat achieves this goal by providing abstract models for prepared data, preconfigured to work well together, integrated into a data preparation pipeline and able to be easily customised to suit any analysis.

Without a template such as bibat, users must either devise a new data modelling framework from scratch as part of an analysis, which is time-consuming, or else to miss out on either the flexibility or features that bibat's approach provides.

4.2 Reproducibility

Bibat ensures reproducibility by providing 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.

4.3 Collaboration

Bibat provides a preconfigured test environment, continuous integration, linting and pre-commit hooks, making it 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.

4.4 Installing dependencies

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.

4.5 Fitting modes

As part of a Bayesian workflow it is often 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 ubiquitous scenario by introducing an abstraction called "fitting mode" and a corresponding Pydantic base class `FittingMode`, along with and several subclasses for commonly used cases.

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`.

4.6 Configure fitting concisely.

Bibat provides a Pydantic class `InferenceConfiguration`, parsed from toml files in the `inferences` directory. By writing a toml file per inference the user can configure statistical model, dataset, fitting modes and computational settings (globally or per fitting mode).

This approach appropriately separates configuration from logic and avoids unnecessary duplication of configuration for runs with the same model and dataset.

4.7 Storing MCMC samples

Bibat provides code that automatically saves its output `InferenceData` objects in zarr format. This format distributes information from the same set of samples into multiple smaller files, thereby helping to avoid file size limits of online repository hosting services.

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 (Driven Data 2022), 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 differently from bibat. For example, bambi (Capretto et al. 2020) and brms (Bürkner 2017) aim to make implementing Bayesian workflows easier by trivialising the task of specifying a

Bayesian generalised linear model and fitting it to a single tabular dataset. This approach does not address several difficulties that bibat does address,

References

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

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.

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

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/.

Greenfeld, Audrey Roy, Dainiel Roy Greenfeld, Raphael Pierzina, et al. 2021. "Cookiecutter." https://pypi.org/project/cookiecutter/.

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.

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/.

Stallman, Richard M, and Roland McGrath. 1991. "GNU Make-A Program for Directing Recompilation."

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

Van Rossum, Guido, and Fred L. Drake. 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.