

# Taweret: a Python package for Bayesian model mixing

K. Ingles<sup>1\*</sup>, D. Liyanage<sup>2\*</sup>, A. C. Semposki<sup>3\*</sup>, and J. C. Yannotty<sup>4\*</sup>

<sup>1</sup> Illinois Center for Advanced Study of the Universe & Department of Physics, University of Illinois Urbana-Champaign, USA <sup>2</sup> Department of Physics, The Ohio State University, USA <sup>3</sup> Department of Physics and Astronomy & Institute of Nuclear and Particle Physics, Ohio University, USA <sup>4</sup> Department of Statistics, The Ohio State University, USA ¶ Corresponding author \* These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- Review
- Repository
- Archive

Editor: Rachel Kurchin

## Reviewers:

- @julienmalard
- @gchure

Submitted: 31 October 2023

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

Uncertainty quantification using Bayesian methods is a growing area of research. Bayesian model mixing (BMM) is a recent development which combines the predictions from multiple models such that each the fidelity of individual models is preserved in the final result. Practical tools and analysis suites that facilitate such methods are therefore needed. Taweret<sup>1</sup> introduces BMM to existing Bayesian uncertainty quantification efforts. Currently Taweret contains three individual Bayesian model mixing techniques, each pertaining to a different type of problem structure; we encourage the future inclusion of user-developed mixing methods. Taweret's first use case is in nuclear physics, but the package has been structured such that it should be adaptable to any research engaged in model comparison or model mixing.

## Statement of need

In physics applications, multiple models with different physics assumptions can be used to describe an underlying system of interest. Though each model may have similar predictive accuracy on average, the fidelity of the approximation across a subset of the domain may differ drastically for each of the models under consideration. In such cases, inference and prediction based on a single model may be unreliable. One strategy for improving accuracy is to combine, or “mix”, the predictions from each model using a linear combination or weighted average with input-dependent weights. This approach is intended to improve reliability of inference and prediction and properly quantify model uncertainties. When operating under a Bayesian framework, this technique is referred to as Bayesian model mixing (BMM). In general, model mixing techniques are designed to combine the individual mean predictions or density estimates from the  $K$  models under consideration. For example, *mean-mixing* techniques predict the underlying system by

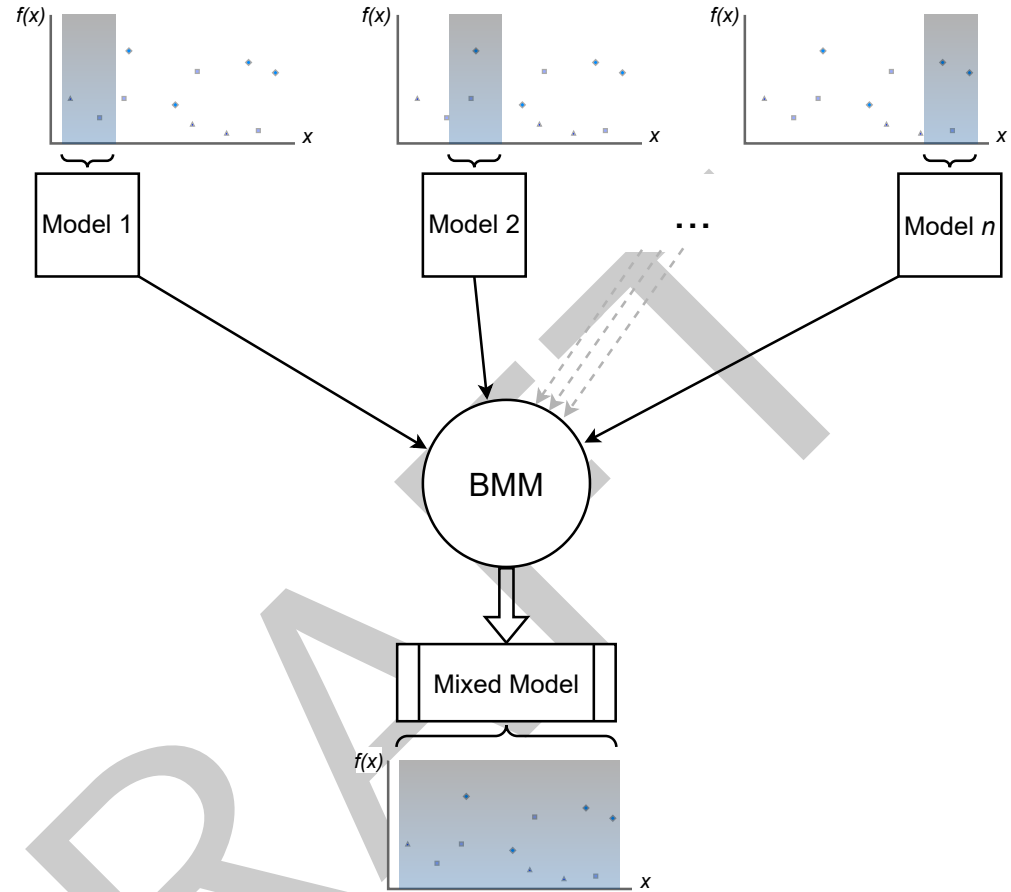
$$E[Y | x] = \sum_{k=1}^K w_k(x) f_k(x),$$

where  $E[Y | x]$  denotes the mean of  $Y$  given the vector of input parameters  $x$ ,  $f_k(x)$  is the mean prediction under the  $k^{\text{th}}$  model  $\mathcal{M}_k$ , and  $w_k(x)$  is the corresponding weight function. The *density-mixing* approach estimates the underlying predictive density by

$$p(Y_0 | x_0, Y) = \sum_{k=1}^K w_k(x_0) p(Y_0 | x_0, Y, \mathcal{M}_k),$$

<sup>1</sup>Taweret is the Egyptian goddess, known as the protector of children and women, whose body is a fusion of a hippo, lion and crocodile which represent her ferocity. Similarly, Taweret, the package, seeks to fuse models together to represent observed phenomena.

where  $p(Y_0 | x_0, Y, \mathcal{M}_k)$  represents the predictive density of a future observation  $Y_0$  with respect to the  $k^{\text{th}}$  model  $\mathcal{M}_k$  at a new input  $x_0$ . In either BMM setup, a key challenge is defining  $w_k(x)$ —the functional relationship between the inputs and the weights.



**Figure 1:** Schematic of Bayesian model mixing. Each model has region of parameter space where it has a high fidelity, but all models are meant to describe the same phenomenon. To obtain a model that works well for all of parameter space, we combine them using Bayesian model mixing methods

This work introduces Taweret, a Python package for Bayesian model mixing that includes three novel approaches for combining models, each of which defines the weight function in a unique way (see Table 1 for a comparison of each method). This package has been developed as an integral piece of the Bayesian Analysis of Nuclear Dynamics (BAND) collaboration's software. BAND is a multi-institutional effort to build a cyber-infrastructure framework for use in the nuclear physics community (Beyer et al., 2023; Phillips et al., 2021). The software is designed to lower the barrier for researchers to employ uncertainty quantification in their experiments, and to integrate, as best as possible, with the community's current standards concerning coding style (pep8). Bayesian model mixing is one of BAND's four central pillars in this framework (the others being emulation, calibration, and experimental design).

In addition to this need, we are aware of several other fields outside of physics that use techniques such as model stacking and Bayesian model averaging (BMA) (Fragoso et al., 2018), e.g., statistics (Yao et al., 2018, 2022), meteorology (Sloughter et al., 2007), and neuroscience (FitzGerald et al., 2014). It is expected that the Bayesian model mixing methods presented in Taweret can also be applied to use cases within these fields. Statisticians have developed several versatile BMA/stacking packages, e.g. (Raftery et al., 2022; Vehtari et al.,

2017). However, the only BMM-based package available is SAMBA—a BAND collaboration effort that was developed for testing BMM methods on a toy model (Semposki et al., 2022a). Taweret’s increased functionality, user-friendly structure, and diverse selection of mixing methods make it a marked improvement over SAMBA.

## Structure

### Overview of methods

**Table 1:** A summary of the three BMM approaches currently implemented in Taweret. Note that  $K \geq 2$ . Following the method name and the type of mixing model, the *Number of inputs* column details the dimensions of the parameter which the mixing weights depend on (e.g., in heavy-ion collisions this is the centrality bin); the *Number of outputs* details how many observables the models themselves can have to compute the model likelihood (e.g., in heavy-ion collisions this can include charge multiplicities, transverse momentum distributions, transverse momentum fluctuations, etc.); the *Number of models* column details how many models the mixing method can combine, and the *Weight functions* column describes the available parameterization of how the mixing weights depend on the input parameter.

Method	Type	Number of inputs	Number of outputs	Number of models	Weight functions
Bivariate linear mixing	Mean & Density	1	$\geq 1$	2	<ul style="list-style-type: none"> <li>Step,</li> <li>Sigmoid,</li> <li>Asymmetric 2-step Precision weighting</li> </ul>
Multivariate mixing	Mean	1	1	$K$	
BART mixing	Mean	$\geq 1$	1	$K$	Regression trees

### Bivariate linear mixing

The full description of this mixing method and several of its applications in relativistic heavy-ion collision physics can be found in the Ph.D. thesis of D. Liyanage (Liyanage, 2023). The bivariate linear mixing method can mix two models either using a density-mixing or a mean-mixing strategy. Currently, this is the only mixing method in Taweret that can also calibrate the models while mixing. Each physics-based model we consider may have unknown parameters which have physical meaning. In this context, Bayesian calibration corresponds to the process of using observational data to learn the values (and more generally, the posterior distributions) of these unknown parameters. Most approaches in model mixing and model averaging use a two-step approach: (step 1) fit individual models using a subset of the data; (step 2) mix the predictions from each model (the results from step 1) using the other subset of the data to learn the weights. Therefore, this joint calibration of and mixing idea looks to do everything at once, rather than use the two step process.

The user may choose among the following mixing functions:

- step:  $\Theta(\beta_0 - x)$
- sigmoid:  $\exp[(x - \beta_0)/\beta_1]$
- asymmetric 2-step:  $\alpha\Theta(\beta_0 - x) + (1 - \alpha)\Theta(\beta_1 - x)$ .

Here  $\Theta$  denotes the Heaviside step function,  $\beta_0$  and  $\beta_1$  determine the shape of the weight function and are inferred from the experimental data, and  $x$  is the model input parameter (which is expected to be 1-dimensional for this mixing method).

## 79 Multivariate model mixing

80 Another Bayesian model mixing method incorporated into Taweret was originally published in  
 81 (Sempowski et al., 2022a), and was the focus of the BMM Python package SAMBA (Sempowski  
 82 et al., 2022b). It can be described as combining models by weighting each of them by  
 83 their precision, defined as the inverse of their respective variances. The posterior predictive  
 84 distribution (PPD) of the mixed model is a Gaussian and can be expressed as

$$\mathcal{M}_{\dagger} \sim \mathcal{N}(f_{\dagger}, Z_P^{-1}) : \quad f_{\dagger} = \frac{1}{Z_P} \sum_{k=1}^K \frac{1}{\sigma_k^2} f_k, \quad Z_P \equiv \sum_{k=1}^K \frac{1}{\sigma_k^2},$$

85 where  $\mathcal{N}(\mu, \sigma^2)$  is a normal distribution with mean  $\mu$  and variance  $\sigma^2$ ,  $Z_P$  is the precision of  
 86 the models, and each individual model is assumed to possess a Gaussian form such as

$$\mathcal{M}_k \sim \mathcal{N}(f_k(x), \sigma_k^2(x)).$$

87 Here,  $f_k(x)$  is the mean of the model  $k$ , and  $\sigma_k^2(x)$  its variance, both at input parameter  $x$ .

88 In this method, the software receives the one-dimensional input space  $X$ , the mean of the  $k$   
 89 models at each point  $x \in X$  (hence it is a mean-based mixing procedure), and the variances  
 90 of the models at each point  $x \in X$ . Each model is assumed to have been calibrated prior to  
 91 being included in the mix. The ignorance of this mixing method with respect to how each  
 92 model was generated allows for any model to be used, including Bayesian Machine Learning  
 93 tools such as Gaussian Processes (Sempowski et al., 2022a) and Bayesian Neural Networks  
 94 (Kronheim et al., 2022).

## 95 Model mixing using Bayesian additive regression trees

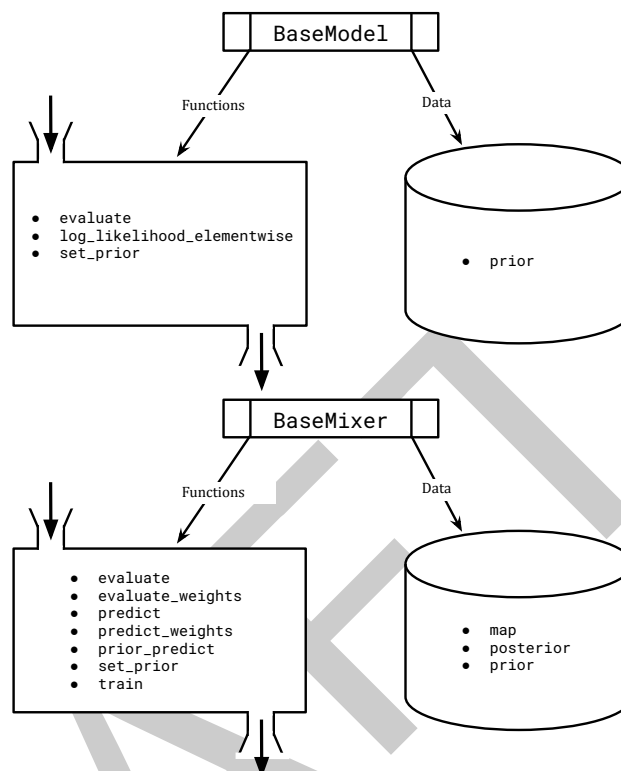
96 A third BMM approach implemented in Taweret adopts a mean-mixing strategy which models  
 97 the weight functions using Bayesian Additive Regression Trees (BART) conditional on the  
 98 mean predictions from a set of  $K$  models (Yannotty et al., 2023). This approach enables the  
 99 weight functions to be adaptively learned using tree bases and avoids the need for user-specified  
 100 basis functions (such as a generalized linear model). Formally, the weight functions are defined  
 101 by

$$w_k(x) = \sum_{j=1}^m g_k(x; T_j, M_j), \quad \text{for } k = 1, \dots, K$$

102 where  $g_k(x; T_j, M_j)$  defines the  $k^{\text{th}}$  output of the  $j^{\text{th}}$  tree,  $T_j$ , using the associated set of  
 103 parameters,  $M_j$ . Each weight function is implicitly regularized via a prior to prefer the interval  
 104  $[0, 1]$ . Furthermore, the weight functions are not required to sum-to-one and can take values  
 105 outside of the range of  $[0, 1]$ . This regularization approach is designed to maintain the flexibility  
 106 of the model while also encouraging the weight functions to take values which preserve desired  
 107 inferential properties.

108 This BART-based approach is implemented in C++ with the trees module in Taweret acting  
 109 as a Python interface. The C++ back-end implements Bayesian tree models and originates from  
 110 the *Open Bayesian Trees Project* (OpenBT) (Pratola et al., 2023). This module serves as an  
 111 example for how existing code bases can be integrated into the Taweret framework.

## 112 Overview of package structure



**Figure 2:** Diagram depicting the base classes, their methods (functions) and their properties (data).

113 Taweret uses abstract base classes to ensure compatibility and uniformity of models and mixing  
114 methods. The two base classes are **BaseModel** and **BaseMixer** located in the core folder (see  
115 Fig. 1 for a schematic); any model mixing method developed with Taweret is required to  
116 inherit from these. The former represents physics-based models that may include parameters  
117 which need to be determined by Bayesian inference. The latter, **BaseMixer**, represents a mixing  
118 method used to combine the predictions from the physics-based models using Bayesian model  
119 mixing.

120 The design philosophy for Taweret is to make it easy to switch between mixing methods  
121 without having to rewrite an analysis script. Thus, the base classes prescribe which functions  
122 need to be present for interoperability between mixing methods, and in particular, the models  
123 being called in the method. The functions required by **BaseModel** are

- 124 ■ **evaluate** - gives a point prediction for the model;
- 125 ■ **log\_likelihood\_elementwise** - calculates the log-likelihood, reducing along the last
- 126 axis of an array if the input array has multiple axes;
- 127 ■ **set\_prior** - sets priors for parameters in the model.

128 The functions required by **BaseMixer** are

- 129 ■ **evaluate** - gives point prediction for the mixed model given a set of parameters;
- 130 ■ **evaluate\_weights** - gives point prediction for the weights given a set of parameters;
- 131 ■ **map** - returns the maximum *a posteriori* estimate for the parameters of the mixed model
- 132 (which includes both the weights and any model parameters);

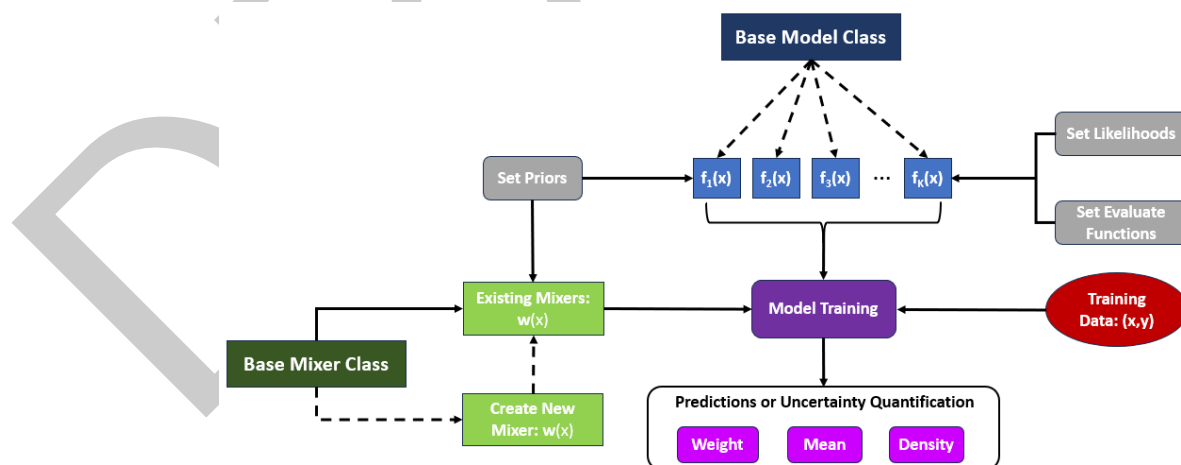
- 133     ▪ posterior - returns the chains of the sampled parameters from the mixed model;
- 134     ▪ predict - returns the posterior predictive distribution for the mixed model;
- 135     ▪ predict\_weights - returns the posterior predictive distribution for the model weights;
- 136     ▪ prior - returns the prior distributions (typically objects, not arrays);
- 137     ▪ prior\_predict - returns the prior predictive distribution for the mixed model;
- 138     ▪ set\_prior - sets the prior distributions for the mixing method;
- 139     ▪ train - executes the Bayesian model mixing step.

140   Following our design philosophy, the general workflow for an analysis using Taweret is described  
141   in Fig. 2. From this, one can see three sources of information are generally required for an  
142   analysis: a selected mixing method, a model set, and training data. Each of these sources are  
143   connected through the training phase, which is where the mixing weights are learned. This  
144   leads into the prediction phase, where final predictions are obtained for the overall system and  
145   the weight functions. This process is summarized in the code snippet below. This workflow is  
146   preserved across the various methods implemented in Taweret and is intended to be maintained  
147   for future mixing methods included in this work.

```
from mix.mix_method import MixMethod
from models.my_model import MyModel

mixer = MixMethod(models={'model_1': MyModel(...), ...})
mixer.set_prior(...)
mixer.train(...)
mixer.predict(...)
mixer.predict_weights(...)
```

148   Extending Taweret with a custom class or model simply requires that you inherit from the  
149   base classes and implement the required functions.



**Figure 3:** The general workflow for an analysis using Taweret. (Blue) The user must define each of the  $K$  models as a class inherited from `BaseModel`. (Green) The user can select an existing mixing method from Taweret (solid) or contribute a new method (dashed). (Purple) The model is trained using a set of training data (red), the model set (blue), and the selected mixing method (green). Predictions and uncertainty quantification follow from the training process.

## Taweret moving forward

There are certainly many improvements that can be made to Taweret. An obvious one is a generalization of the bivariate linear mixing; this could be the mixing of an arbitrary number of models at the density level. Complementary to this density mixing method is a stochastic, mean-mixing method of arbitrary number  $K$  of models. An extension of the Multivariate Mixing method to multi-dimensional input and output spaces, correlated models, as well as calibration during mixing, is anticipated in future releases. Lastly, to facilitate the utilization of this growing framework, we hope to enable continuous integration routines for individuals contributing and create docker images that will run Taweret.

## Contributions

All authors have contributed to the development of the Taweret framework, while individual mixing methods were developed by D. Liyanage (bivariate linear mixing), A. Semposki (multivariate model mixing), and J. Yanotty (additive regression trees). Ongoing work by K. Ingles will be included in the next version release of Taweret.

## Disclosure statement

The authors of this work are not aware of any conflicts of interest that would affect this research.

## Acknowledgments

We thank Daniel R. Phillips, Ulrich Heinz, Matt Pratola, Kyle Godbey, Stefan Wild, Sunil Jaiswal, and all other BAND members for crucial feedback and discussion during the development stage of this package. This work is supported by the CSSI program Award OAC-2004601 (DL, ACS, JCY). ACS also acknowledges support from the Department of Energy (contract no. DE-FG02-93ER40756).

## References

- Beyer, K., Buskirk, L., Chan, M. Y.-H., Chang, T. H., DeBoer, R. J., Furnstahl, R. J., Giuliani, P., Godbey, K., Ingles, K., Liyanage, D., Nunes, F. M., Odell, D., Phillips, D. R., Plumlee, M., Pratola, M. T., Semposki, A. C., Sürer, Ö., Wild, S. M., & Yannotty, J. C. (2023). *BANDFramework: An open-source framework for Bayesian analysis of nuclear dynamics* (Version 0.3.0). <https://github.com/bandframework/bandframework>
- FitzGerald, T. H. B., Dolan, R. J., & Friston, K. J. (2014). Model averaging, optimal inference, and habit formation. *Frontiers in Human Neuroscience*, 8:457. <https://doi.org/10.3389/fnhum.2014.00457>
- Fragoso, T. M., Bertoli, W., & Louzada, F. (2018). Bayesian model averaging: A systematic review and conceptual classification. *International Statistical Review*, 86(1), 1–28. <https://doi.org/10.1111/insr.12243>
- Kronheim, B., Kuchera, M., & Prosper, H. (2022). TensorBNN: Bayesian inference for neural networks using TensorFlow. *Computer Physics Communications*, 270, 108168. <https://doi.org/10.1016/j.cpc.2021.108168>
- Liyanage, D. (2023). *Multifaceted study of ultrarelativistic heavy ion collisions* [PhD thesis, Ohio State University; The Ohio State University]. [http://rave.ohiolink.edu/etdc/view?acc\\_num=osu1689508609203123](http://rave.ohiolink.edu/etdc/view?acc_num=osu1689508609203123)



- 191 Phillips, D. R., Furnstahl, R. J., Heinz, U., Maiti, T., Nazarewicz, W., Nunes, F. M., Plumlee,  
192 M., Pratola, M. T., Pratt, S., Viens, F. G., & Wild, S. M. (2021). Get on the BAND  
193 Wagon: A Bayesian Framework for Quantifying Model Uncertainties in Nuclear Dynamics.  
194 *Journal of Physics G*, 48(7), 072001. <https://doi.org/10.1088/1361-6471/abf1df>
- 195 Pratola, M. T., McCulloch, R. E., Chipman, H. A., & Horiguchi, A. (2023). *Open Bayesian*  
196 *Trees Project*. <https://bitbucket.org/mpratola/openbt/wiki/Home>
- 197 Raftery, A., Hoeting, J., Volinsky, C., Painter, I., & Yeung, K. Y. (2022). *BMA: Bayesian*  
198 *model averaging*. <https://CRAN.R-project.org/package=BMA>
- 199 Semposki, A. C., Furnstahl, R. J., & Phillips, D. R. (2022a). Interpolating between small-  
200 and large- $g$  expansions using Bayesian model mixing. *Physical Review C*, 106(4), 044002.  
201 <https://doi.org/10.1103/PhysRevC.106.044002>
- 202 Semposki, A. C., Furnstahl, R. J., & Phillips, D. R. (2022b). *SAMBA: SAndbox for Mixing*  
203 *using Bayesian Analysis*. <https://github.com/aseposki/SAMBA>
- 204 Sloughter, J. M., Raftery, A. E., Gneiting, T., & Fraley, C. (2007). Probabilistic quantitative  
205 precipitation forecasting using Bayesian model averaging. *Monthly Weather Review*, 135,  
206 3209–3220. <https://doi.org/10.1175/MWR3441.1>
- 207 Vehtari, A., Gelman, A., & Gabry, J. (2017). *loo: Efficient leave-one-out cross-validation and*  
208 *WAIC for Bayesian models*. <https://mc-stan.org/loo/>
- 209 Yannotty, J. C., Santner, T. J., Furnstahl, R. J., & Pratola, M. T. (2023). Model mixing  
210 using Bayesian additive regression trees. In *Technometrics* (pp. 1–12). Taylor & Francis.  
211 <https://doi.org/10.1080/00401706.2023.2257765>
- 212 Yao, Y., Pirš, G., Vehtari, A., & Gelman, A. (2022). Bayesian Hierarchical Stacking: Some  
213 Models Are (Somewhere) Useful. *Bayesian Analysis*, 17(4), 1043–1071. <https://doi.org/10.1214/21-BA1287>
- 214  
215 Yao, Y., Vehtari, A., Simpson, D., & Gelman, A. (2018). Using stacking to average bayesian  
216 predictive distributions (with discussion). *Bayesian Analysis*, 13 (3) 917 - 1007. <https://doi.org/10.1214/17-BA1091>
- 217