





# SBIAX: Density-estimation simulation-based inference in JAX.

Jed Homer<sup>1\*</sup>

<sup>1</sup> Ludwig-Maximilians-Universität München, Faculty for Physics, University Observatory, Scheinerstrasse 1, München, Deutschland.  \* These authors contributed equally.

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

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished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](#)).

## In partnership with



AMERICAN  
ASTRONOMICAL  
SOCIETY

This article and software are linked with research article DOI [10.3847/xxxxx](https://doi.org/10.3847/xxxxx) <- [update this with the DOI from AAS once you know it.](#), published in the Astrophysical Journal <- The name of the AAS journal..

## Summary

In a typical Bayesian inference problem, the data likelihood is not known. In such analyses, density-estimation simulation-based inference methods can derive a posterior, which typically involves

- simulating a set of data and model parameters  $\{(\xi, \pi)_0, \dots, (\xi, \pi)_N\}$ ,
- obtaining a measurement  $\hat{\xi}$ ,
- compressing the simulations and the measurements - usually with a neural network or linear compression - to a set of summaries  $\{(x, \pi)_0, \dots, (x, \pi)_N\}$  and  $\hat{x}$ ,
- fitting an ensemble of normalising flow or similar density estimation algorithms (e.g. a Gaussian mixture model),
- the optional optimisation of the parameters for the architecture and fitting hyperparameters of the algorithms,
- sampling the ensemble posterior (using an MCMC sampler if the likelihood is fit directly) conditioned on the datavector to obtain parameter constraints on the parameters of a physical model,  $\pi$ .

sbiAx is a code for implementing each of these steps.

As shown in Homer 2024, SBI is shown to successfully obtain the correct posterior widths and coverages given enough simulations which agree with the analytic solution.

## Statement of need

Simulation-based inference (SBI) covers a broad class of statistical techniques such as Approximate Bayesian Computation (ABC), Neural Ratio Estimation (NRE), Neural Likelihood Estimation (NLE) and Neural Posterior Estimation (NPE). These techniques can derive posterior distributions conditioned of noisy data vectors in a rigorous and efficient manner. In particular, density-estimation methods have emerged as a promising method, given their efficiency, using generative models to fit likelihoods or posteriors directly using simulations.

In the field of cosmology, SBI is of particular interest due to complexity and non-linearity of models for the expectations of non-standard summary statistics of the large-scale structure, as well as the non-Gaussian noise distributions for these statistics. The assumptions required for the complex analytic modelling of these statistics as well as the increasing dimensionality of data returned by spectroscopic and photometric galaxy surveys limits the amount of information that can be obtained on fundamental physical parameters. Therefore, the study and research into current and future statistical methods for Bayesian inference is of paramount importance for the field of cosmology.

The software we present, sbiax, is designed to be used by machine learning and physics researchers for running Bayesian inferences using density-estimation SBI techniques. These models can be fit easily with multi-accelerator training and inference within the code. This code - written in jax (?) - allows for seamless integration of cutting edge generative models to SBI, including continuous normalising flows (Grathwohl et al., 2018), matched flows (Lipman et al., 2023) and masked autoregressive flows (Papamakarios et al., 2018; Ward, [release year of version]). The code features integration with the optuna (Akiba et al., 2019) hyperparameter optimisation framework which would be used to ensure consistent analyses, blackjax (Cabezas et al., 2024) for fast MCMC sampling and neural network compression methods with equinox (Kidger & Garcia, 2021). The design of sbiax allows for new density estimation algorithms to be trained and sampled from.

## Density estimation with normalising flows

The use of density-estimation in SBI has been accelerated by the advent of normalising flows. These models parameterise a change-of-variables  $y = f_\phi(x; \pi)$  between a simple base distribution (e.g. a multivariate unit Gaussian  $\mathcal{G}[z|0, \mathbb{I}]$ ) and an unknown distribution  $q(x|\pi)$  (from which we have simulated samples  $x$ ). Naturally, this is of particular importance in inference problems in which the likelihood is not known. The change-of-variables is fit from data by training neural networks to model the transformation in order to maximise the log-likelihood of the simulated data  $x$  conditioned on the parameters  $\pi$  of a simulator model. The mapping is expressed as

$$y = f_\phi(x; \pi),$$

where  $\phi$  are the parameters of the neural network. The log-likelihood of the flow is expressed as

$$\log p_\phi(x|\pi) = \log \mathcal{G}[f_\phi(x; \pi)|0, \mathbb{I}] + \log |\mathbf{J}_{f_\phi}(x; \pi)|,$$

This density estimate is fit to a set of  $N$  simulation-parameter pairs  $\{(\xi, \pi)_0, \dots, (\xi, \pi)_N\}$  by minimising a Monte-Carlo estimate of the KL-divergence

$$\begin{aligned} \langle D_{KL}(q||p_\phi) \rangle_{\pi \sim p(\pi)} &= \int d\pi p(\pi) \int dx q(x|\pi) \log \frac{q(x|\pi)}{p_\phi(x|\pi)}, \\ &= \int d\pi \int dx p(\pi, x) [\log q(x|\pi) - \log p_\phi(x|\pi)], \\ &\geq - \int d\pi \int dx p(\pi, x) \log p_\phi(x|\pi), \\ &\approx - \frac{1}{N} \sum_{i=1}^N \log p_\phi(x_i|\pi_i), \end{aligned} \tag{1}$$

where  $q(x|\pi)$  is the unknown likelihood from which the simulations  $x$  are drawn. This applies similarly for an estimator of the posterior (instead of the likelihood as shown here) and is the basis of being able to estimate the likelihood or posterior directly when an analytic form is not available. If the likelihood is fit from simulations, a prior is required and the posterior is sampled via an MCMC given some measurement. This is implemented within the code.

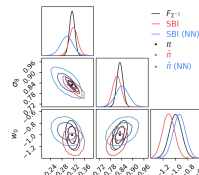
An ensemble of density estimators has a likelihood which is written as

$$p_{\text{ensemble}}(\xi|\pi) = \sum_i \alpha_i p_{\phi_i}(\hat{\xi}|\pi)$$

69 where

$$\alpha_i = \frac{\exp(p_{\phi_i}(\hat{\xi}|\pi))}{\sum_j \exp(p_{\phi_j}(\hat{\xi}|\pi))}$$

70 are the weights of each density estimator in the ensemble. This ensemble likelihood can be  
71 easily sampled with an MCMC sampler.



72 ??

## 73 Acknowledgements

74 We thank the developers of these packages for their work and for making their code available  
75 to the community.

## 76 References

- 77 Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation  
78 hyperparameter optimization framework. *The 25th ACM SIGKDD International Conference*  
79 *on Knowledge Discovery & Data Mining*, 2623–2631.
- 80 Cabezas, A., Corenflos, A., Lao, J., & Louf, R. (2024). *BlackJAX: Composable Bayesian*  
81 *inference in JAX*. <https://arxiv.org/abs/2402.10797>
- 82 Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., & Duvenaud, D. (2018).  
83 *FFJORD: Free-form continuous dynamics for scalable reversible generative models*. <https://arxiv.org/abs/1810.01367>
- 84
- 85 Kidger, P., & Garcia, C. (2021). Equinox: Neural networks in JAX via callable PyTrees and  
86 filtered transformations. *Differentiable Programming Workshop at Neural Information*  
87 *Processing Systems 2021*.
- 88 Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., & Le, M. (2023). *Flow matching for*  
89 *generative modeling*. <https://arxiv.org/abs/2210.02747>
- 90 Papamakarios, G., Pavlakou, T., & Murray, I. (2018). *Masked autoregressive flow for density*  
91 *estimation*. <https://arxiv.org/abs/1705.07057>
- 92 Ward, D. ([release year of version]). *FlowJAX: Distributions and normalizing flows in jax*  
93 ([version number]). <https://doi.org/10.5281/zenodo.10402073>