




pymcmcstat: A Python Package for Bayesian Inference Using Delayed Rejection Adaptive Metropolis

Paul R. Miles¹

¹ Department of Mathematics, North Carolina State University

DOI: [10.21105/joss.01417](https://doi.org/10.21105/joss.01417)

Software

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

Submitted: 22 April 2019

Published: 06 June 2019

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Summary

Many scientific problems require calibrating a set of model parameters to fit a set of data. Various approaches exist for performing this calibration, but not all of them account for underlying uncertainty within the problem. Examples of this uncertainty include random noise within experimental measurements as well as errors due to model discrepancy; i.e., missing physics in the model. A Bayesian framework provides a natural perspective from which to perform model calibration to accommodate this uncertainty. To utilize this approach, we make several assumptions regarding the problem.

1. Parameters (q) are treated as random variables with underlying distributions instead of unknown but fixed values.
2. Observations $F^{data}(i)$ are expected to be equal to the model response $F(i; q)$ plus independent and identically distributed random errors $\epsilon_i \rightarrow F^{data}(i) = F(i; q) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

The goal of the calibration process is to infer the parameters' posterior distributions given a set of observations - $\pi(q|F^{data}(i))$. Once these parameter distributions are known, the Bayesian approach provides a natural framework in which to consider how this uncertainty propagates and affects model predictions.

We designed the `pymcmcstat` package for engineers and scientists interested in using Bayesian methods to quantify model parameter uncertainty. Furthermore, we had the goal of providing a Python platform for researchers familiar with the MATLAB toolbox `mcmcstat` developed by Marko Laine. To accommodate a diverse audience, we constructed several [tutorials](#) to guide the user through the various stages of setting up a problem, such as defining the data structure, model parameters, and simulation options. Currently, the package is limited to Gaussian likelihood and prior functions; however, these are still suitable for a wide variety of scientific problems. As many individuals are not necessarily familiar with the statistical nomenclature behind the Bayesian approach, the package simply requires the user to define a function that calculates the sum-of-squares error with respect to the model and the observations which is consistent with the second assumption listed above. Information known *a priori* about the parameter distributions is defined in the prior function; however, the default program behavior is to use a uniform prior which is a common approach for these types of problems.

Bayesian inference is a powerful tool for quantifying model input uncertainty, and Markov Chain Monte Carlo (MCMC) methods present a computationally tractable means for constructing posterior distributions for input parameters (Smith, 2014). Within MCMC, a Metropolis algorithm chooses whether to accept or reject proposed parameter values. This approach to parameter acceptance allows the algorithm to effectively sample the parameter space and avoid issues that often arise due to local minima during the calibration

process. A variety of Metropolis algorithms can be used within MCMC. Ideally, information learned about the posterior distribution as candidate parameters are accepted will be used to update the proposal distribution. This can be accomplished via a variety of adaptive Metropolis (AM) algorithms (Andrieu & Thoms, 2008, Haario, Saksman, Tamminen, & others (2001), Roberts & Rosenthal (2009)). In addition to improving the proposal distribution via adaption, it is often advantageous to incorporate delayed rejection (DR) in order to stimulate mixing (Haario, Laine, Mira, & Saksman, 2006). Both mechanisms have been demonstrated to significantly improve the performance of MCMC simulations.

In the Python package `pymcmcstat`, we offer several Metropolis-based algorithms for parameter estimation. These algorithms include:

- Metropolis-Hastings (MH)
- Adaptive-Metropolis (AM): Adapts parameter covariance matrix at specified intervals.
- Delayed-Rejection (DR): Delays rejection by sampling from a narrower proposal distribution.
- Delayed Rejection Adaptive Metropolis (DRAM): DR + AM

The default program employs 2-stage DRAM; however, it is capable of accommodating n -stage delayed rejection. The specific algorithms implemented for adaptive Metropolis and delayed rejection are outlined in (Haario et al., 2001, Haario et al. (2006)).

To the author's knowledge, the package is currently being used for several scientific projects, including radiation source localization using 3D transport models and fractional-order viscoelasticity models of dielectric elastomers.

Acknowledgements

This work was sponsored in part by the NNSA Office of Defense Nuclear Nonproliferation R&D through the Consortium for Nonproliferation Enabling Capabilities.

References

- Andrieu, C., & Thoms, J. (2008). A tutorial on adaptive mcmc. *Statistics and computing*, 18(4), 343–373. doi:[10.1007/s11222-008-9110-y](https://doi.org/10.1007/s11222-008-9110-y)
- Haario, H., Laine, M., Mira, A., & Saksman, E. (2006). DRAM: Efficient adaptive mcmc. *Statistics and computing*, 16(4), 339–354. doi:[10.1007/s11222-006-9438-0](https://doi.org/10.1007/s11222-006-9438-0)
- Haario, H., Saksman, E., Tamminen, J., & others. (2001). An adaptive metropolis algorithm. *Bernoulli*, 7(2), 223–242. Retrieved from <https://projecteuclid.org/euclid.bj/1080222083>
- Roberts, G. O., & Rosenthal, J. S. (2009). Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2), 349–367. doi:[10.1198/jcgs.2009.06134](https://doi.org/10.1198/jcgs.2009.06134)
- Smith, R. C. (2014). *Uncertainty quantification: Theory, implementation, and applications*. SIAM.