rmcmc: Robust Markov chain Monte Carlo methods in R

Matthew M. Graham[1] and **Samuel Livingstone**[1]

1 University College London

DOI: 10.21105/joss.08594

Software

- Review
- Repository
- Archive

Editor: Sehrish Kanwal

Reviewers:

- @AntoineSoetewey
- @larryshamalama

Submitted: 19 February 2025
Published: 31 October 2025

Summary

Generating samples from a probability distribution is a common requirement in many disciplines. In Bayesian inference, for example, the distribution of interest is the posterior over parameters of a model, given data assumed to be generated from the model. Expectations with respect to the posterior can be estimated using samples, allowing computation of posterior means, variances and other quantities of interest. *Markov chain Monte Carlo* (MCMC) methods are a general class of algorithms for approximately sampling from probability distributions.

`rmcmc` is an R package providing implementations of MCMC methods for sampling from distributions on $\mathbb{R}^d$ for $d \geq 1$. It provides a general purpose implementation of the Barker proposal (Livingstone & Zanella, 2022), a gradient-based MCMC algorithm inspired by the Barker accept-reject rule (Barker, 1965). `rmcmc` also provides implementations of other MCMC algorithms based on random walk, Langevin and Hamiltonian dynamics. It has a flexible interface for performing adaptive MCMC so that algorithmic tuning parameters can be learned in a bespoke manner (Andrieu & Thoms, 2008; Haario et al., 2001). The key function provided by the package is `sample_chain`, which samples a Markov chain with a user-specified stationary distribution. The chain is sampled by generating proposals and accepting or rejecting them using the Metropolis–Hastings (Hastings, 1970; Metropolis et al., 1953) algorithm. During an initial warm-up stage, the parameters of the proposal distribution can be adapted. Schemes are available to both tune:

- the scale of the proposals by coercing the average acceptance rate to a target value,
- the shape of the proposals to match covariance estimates under the target distribution.

Statement of need

For target distributions with smooth density functions, gradient-based MCMC methods can provide improved sampling efficiency over simpler schemes such as *random-walk Metropolis* (RWM) in high-dimensional settings (Beskos et al., 2013; Roberts & Rosenthal, 1998; Roberts & Tweedie, 1996). For methods such as *Metropolis adjusted Langevin algorithm* (MALA) (Besag, 1994; Rossky et al., 1978) and *Hamiltonian Monte Carlo* (HMC) (Duane et al., 1987; Neal, 2011), this improved efficiency comes with a cost. Specifically these methods exhibit decreased robustness to tuning of the algorithm's parameters, compared to non-gradient based methods such as RWM (Livingstone & Zanella, 2022).

The Barker proposal provides a middle road. It offers similar efficiency in high-dimensional settings as other gradient-based methods such as MALA (Vogrinc et al., 2023), while allowing easier adaptive tuning of the algorithm's parameters. It also provides improved robustness to certain forms of irregularity, such as skewness, in the target distribution (Hird et al., 2020).

`rmcmc` fills a niche in the R statistical computing ecosystem, providing general purpose implementations of the Barker proposal, as well as RWM, MALA, and HMC. The package also supports several schemes for adapting the proposal parameters. Significant flexibility is available in specifying the log density of the target distribution, and its gradient. Users can:

- directly define functions to compute the log density and its gradient;
- specify a formula for the log density which will then be symbolically differentiated using the `deriv` function in the base R `stats` package (R Core Team, 2024);
- or define a model using Stan's modelling syntax via the R interface of BridgeStan (Carpenter et al., 2017; Roualdes et al., 2023).

The package has a modular design, allowing users to easily try out different algorithmic components and options, and to extend the package with new algorithms. This is exemplified in the Barker proposal implementation, which supports customizing the distribution over the auxiliary variables used in generating the proposal. As discussed in Vogrinc et al. (2023) and illustrated in a package vignette, this can significantly improve sampling efficiency in some cases.

`rmcmc` has a pure R codebase with minimal required dependencies, making it a lightweight addition to other projects. The package also interfaces with several others in addition to BridgeStan. For instance, a wrapper around the *robust adaptive Metropolis* (RAM) (Vihola, 2012) adaptation scheme in the `ramcmc` package is provided (Helske, 2021). The sampled chain traces can also be directly passed to functions for computing summary statistics and convergence diagnostics in the `posterior` or `coda` packages (Bürkner et al., 2024; Plummer et al., 2006).

Related software

Several other R packages provide implementations of MCMC methods. `mcmc` (Geyer & Johnson, 2023) provides a general purpose implementation of RWM, along with a 'morphometric' variant (Johnson & Geyer, 2012), which reparametrizes the target distribution to improve efficiency. `MCMCpack` (Martin et al., 2011) focuses on providing customized MCMC methods for specific classes of statistical model that exploit the structure of the model, though it does also provide a general purpose RWM implementation. `fmcmc` (Yon & Marjoram, 2019) is similar in design to `rmcmc`. It provides a modular framework with a variety of pre-defined proposals such as Gaussian and uniform RWM, and adaptive methods such as RAM and adaptive Metropolis (Haario et al., 2001). None of `mcmc`, `MCMCpack` or `fmcmc` provide gradient-based MCMC methods, which can significantly improve sampling efficiency, as discussed above.

The GitHub repository `gzanella/barker` (Zanella, 2019) contains code to recreate the numerical experiments in Livingstone & Zanella (2022), and provides a basic implementation of the Barker proposal. The R scripts in the repository are not, however, structured into a package, complicating reuse in other projects. The implementation also only provides support for sampling from the proposal and evaluating its log density ratio.

Stan and NIMBLE (de Valpine et al., 2017), with associated R interfaces `rstan` (Stan Development Team, 2024) and `nimble` (de Valpine et al., 2024), are *probabilistic programming languages* (PPLs), domain specific languages for the specification of probabilistic models. Both Stan and NIMBLE also provide implementations of a variety algorithms to perform inference in models defined via their PPLs, and in particular both offer gradient-based MCMC methods.

Stan's default MCMC implementation is a HMC method, which dynamically sets the trajectory lengths when simulating Hamiltonian dynamics to generate proposals (Betancourt, 2017; Hoffman & Gelman, 2014). Stan also includes schemes for adapting an algorithm's scale (step-size) and shape (metric) parameters, but with limited user-flexibility. Stan also does not currently provide an implementation of the Barker proposal. `rmcmc` does also offer a basic HMC implementation, but currently only supports HMC with static or randomized trajectory lengths. One of the proposal scale adaptation schemes implemented in `rmcmc`, is a dual-averaging algorithm (Hoffman & Gelman, 2014; Nesterov, 2009). This matches the corresponding scheme in Stan, but in contrast to Stan alternative schemes are available in `rmcmc`.

NIMBLE supports defining both models and statistical algorithms in its PPL. It provides implementations of a variety of MCMC methods including, RWM, HMC (Turek et al., 2024), and the Barker proposal. Compared to `rmcmc`, NIMBLE's Barker proposal implementation is tightly integrated into the broader NIMBLE framework. This means it can only easily be applied to models specified in NIMBLE. NIMBLE's Barker proposal implementation provides support for adapting the proposal scale and shape parameters. However, unlike `rmcmc`, the adaptation scheme provided is fixed, without an ability to swap in alternative adaptation schemes.

Acknowledgements

Development of `rmcmc` was supported by a UK Engineering and Physical Sciences Research Council New Investigator Award EP/V055380/1.

References

Andrieu, C., & Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, *18*, 343–373. https://doi.org/10.1007/s11222-008-9110-y

Barker, A. A. (1965). Monte Carlo calculations of the radial distribution functions for a proton electron plasma. *Australian Journal of Physics*, *18*(2), 119–134. https://doi.org/10.1071/ph650119

Besag, J. (1994). Comments on "Representations of knowledge in complex systems" by U. Grenander and MI Miller. *Journal of the Royal Statistical Society: Series B (Methodological)*, *56*(4), 549–581.

Beskos, A., Pillai, N. S., Roberts, G. O., Sanz-Serna, J., & Stuart, A. (2013). Optimal tuning of the Hybrid Monte-Carlo algorithm. *Bernoulli*, *19*(5a), 1501–1534. https://doi.org/10.3150/12-bej414

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv Preprint arXiv:1701.02434*.

Bürkner, P.-C., Gabry, J., Kay, M., & Vehtari, A. (2024). *posterior: Tools for working with posterior distributions*. https://doi.org/10.32614/cran.package.posterior

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*, 1–32.

de Valpine, P., Paciorek, C., Turek, D., Michaud, N., Anderson-Bergman, C., Obermeyer, F., Wehrhahn Cortes, C., Rodrìguez, A., Temple Lang, D., & Paganin, S. (2024). *NIMBLE: MCMC, particle filtering, and programmable hierarchical modeling* (Version 1.3.0) [Computer software]. https://doi.org/10.5281/zenodo.1211190

de Valpine, P., Turek, D., Paciorek, C., Anderson-Bergman, C., Temple Lang, D., & Bodik, R. (2017). Programming with models: Writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, *26*, 403–413. https://doi.org/10.1080/10618600.2016.1172487

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, *195*(2), 216–222.

Geyer, C. J., & Johnson, L. T. (2023). *mcmc: Markov chain Monte Carlo* (R package version 0.9.8). https://github.com/cjgeyer/mcmc

Haario, H., Saksman, E., & Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, *7*(2), 223–242.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97–109. https://doi.org/10.1093/oso/9780198509936.003.0015

Helske, J. (2021). *ramcmc: Robust adaptive Metropolis algorithm*. https://doi.org/10.32614/cran.package.ramcmc

Hird, M., Livingstone, S., & Zanella, G. (2020). A fresh take on "Barker dynamics" for MCMC. *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, 169–184. https://doi.org/10.1007/978-3-030-98319-2_8

Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*(1), 1593–1623.

Johnson, L. T., & Geyer, C. J. (2012). Variable transformation to obtain geometric ergodicity in the random-walk Metropolis algorithm. *The Annals of Statistics*, 3050–3076. https://doi.org/10.1214/12-aos1048

Livingstone, S., & Zanella, G. (2022). The Barker proposal: Combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *84*(2), 496–523. https://doi.org/10.1111/rssb.12482

Martin, A. D., Quinn, K. M., & Park, J. H. (2011). MCMCpack: Markov chain Monte Carlo in R. *Journal of Statistical Software*, *42*(9), 22. https://doi.org/10.18637/jss.v042.i09

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1007/s12045-022-1419-x

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In *Handbook of Markov chain Monte Carlo* (pp. 113–162). Chapman; Hall/CRC. https://doi.org/10.1201/b10905-6

Nesterov, Y. (2009). Primal-dual subgradient methods for convex problems. *Mathematical Programming*, *120*(1), 221–259. https://doi.org/10.2139/ssrn.912637

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for MCMC. *R News*, *6*(1), 7–11. https://journal.r-project.org/archive/

R Core Team. (2024). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://doi.org/10.32614/r.manuals

Roberts, G. O., & Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(1), 255–268. https://doi.org/10.1111/1467-9868.00123

Roberts, G. O., & Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, *2*(4), 341–363. https://doi.org/10.2307/3318418

Rossky, P. J., Doll, J. D., & Friedman, H. L. (1978). Brownian dynamics as smart Monte Carlo simulation. *The Journal of Chemical Physics*, *69*(10), 4628–4633. https://doi.org/10.1063/1.436415

Roualdes, E. A., Ward, B., Carpenter, B., Seyboldt, A., & Axen, S. D. (2023). BridgeStan: Efficient in-memory access to the methods of a Stan model. *Journal of Open Source Software*, *8*(87), 5236. https://doi.org/10.21105/joss.05236

Stan Development Team. (2024). *RStan: The R interface to Stan*. https://mc-stan.org/

Turek, D., Valpine, P. de, & Paciorek, C. J. (2024). nimbleHMC: An r package for Hamiltonian Monte Carlo sampling in nimble. *Journal of Open Source Software*, *9*(99), 6745. https:

//doi.org/10.21105/joss.06745

Vihola, M. (2012). Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, *22*(5), 997–1008. https://doi.org/10.1007/s11222-011-9269-5

Vogrinc, J., Livingstone, S., & Zanella, G. (2023). Optimal design of the Barker proposal and other locally balanced Metropolis–Hastings algorithms. *Biometrika*, *110*(3), 579–595. https://doi.org/10.1093/biomet/asac056

Yon, G. G. V., & Marjoram, P. (2019). fmcmc: A friendly MCMC framework. *Journal of Open Source Software*, *4*(39), 1427. https://doi.org/10.21105/joss.01427

Zanella, G. (2019). *Code for the paper Livingstone and Zanella (2019) "On the robustness of gradient-based MCMC"*. https://github.com/gzanella/barker