DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

title: 'compareMCMCs: An R package for studying MCMC efficiency'
tags:
- R
- statistics
- Markov chain Monte Carlo
- nimble
- JAGS
authors:
- name: Perry de Valpine
affiliation: 1
- name: Sally Paganin
affiliation: 1
- name: Daniel Turek
affiliation: 2
affiliations:
- name: University of California, Berkeley
- index: 1
- name: Williams College
- index: 2
date: 15 Oct 2021
bibliography: paper.bib

Summary

Markov chain Monte Carlo (MCMC) algorithms are used to stochastically simulate from complicated probability distributions. MCMC is very widely used to implement Bayesian statistical analysis, where the distribution of interest (the "target distribution") is a posterior distribution of parameters given data. In this context, the posterior is known only up to a constant, so only relative probabilities (or densities) can be easily calculated, which is sufficient for MCMC to work. In applied Bayesian statistics, MCMC algorithms for large or complex statistical models and data sets are sometimes run for minutes, hours or days, making them an analysis bottleneck, so there is a premium on efficiency. MCMC efficiency includes both computational speed and algorithmic mixing, which refers to how well the algorithm explores the posterior distribution from one iteration to the next. Computational speed may comprise one or more steps such as algorithm setup, MCMC "burn-in" or "warm-up" phases, and MCMC execution or "sampling."

There are many MCMC algorithms (also called "samplers") and software packages implementing them. Because MCMC samplers can be mixed and matched, with different samplers for the same or different dimensions of a target distribution possibly combined into the same MCMC algorithm, there is an enormous space of MCMC algorithms. Invention of new methods, comparisons among methods, and theoretical study of MCMC mixing are all important areas of active research. Various software packages provide samplers such as Gibbs, adaptive random-walk Metropolis-Hastings, slice, Hamiltonian, multivariate ("block") or other variants of these, and others. Different MCMC algorithms can yield efficiency that differs by orders of magnitude for a particular problem, with these variations in efficiency being problem-dependent.

The R package `compareMCMCs` provides a highly modular system for managing performance comparisons among MCMC software packages for purposes of research on MCMC methods. MCMC runs can take a long time, so the output ("samples") and components of computation time from a run are stored regardless of whether performance metrics are computed immediately. Arbitrary MCMC packages ("MCMC engines") can be added to the system by

writing a simple plug-in or wrapper to manage inputs and outputs in a unified way. Conversions among model parameter names and/or different parameterizations can be provided to standardize across packages. Performance metrics are organized by model parameter (one result per parameter per MCMC engine), by MCMC (one result per MCMC engine), or arbitrarily (a user-defined list of metric results per MCMC engine). Built-in metrics include two methods of estimating effective sample size (ESS), posterior summaries such as mean and common quantiles, efficiency defined as ESS per computation time, rate defined as computation time per ESS, and minimum efficiency per MCMC. New metrics can be provided by a plug-in system and applied programmatically to a set of MCMC samples without re-running the MCMC engines. Finally, standardized graphical comparison pages can be generated in html. Built-in graphical outputs include figures comparing MCMC efficiency and/or rate on a per-parameter or per-MCMC basis as well as comparing posterior distributions. New graphical outputs can be provided by a plug-in system. In summary, `compareMCMCs` is modular and extensible for running new MCMC engines on comparable problems, for creating new metrics of interest (e.g., posterior summaries or effective sample size estimated in different ways), and for creating new graphical comparison outputs into a report.

Statement of need

TBD

Acknowledgements

We thank other developers who have contributed to `nimble`, in which an early version of the features of `compareMCMCs` was first drafted and from which its design needs became clearer. Daniel Turek wrote the first version of what evolved into `compareMCMCs`, and Christopher Paciorek contributed ideas along the way.

References