nimbleHMC: An R package for Hamiltonian Monte Carlo sampling in nimble

NEW DATE HERE (18 July 2023)

Summary

Markov chain Monte Carlo (MCMC) algorithms are widely used for fitting hierarchical models to data. MCMC is the predominant tool used in Bayesian analyses to generate samples from the posterior distribution of model parameters conditional on observed data. MCMC is not a single algorithm, but actually a framework which admits any assignment of sampling techniques to unobserved parameters. There exists a vast set of valid samplers to draw upon, which differ in complexity, autocorrelation of samples produced, and applicability.

Hamiltonian Monte Carlo (HMC; Brooks et al. 2011) sampling is one such technique, applicable to continuous-valued parameters, which uses the gradients to generate large transitions in parameter space. The resulting samples have low autocorrelation, and therefore have high information content, relative for example to an equal-length sequence of highly autocorrelated samples. The No-U-Turn (NUTS) variety of HMC sampling (HMC-NUTS; Hoffman and Gelman 2014) greatly increases the usability of HMC by self-adapting key sampler tuning parameters upon which the overall performance is highly dependent.

Many software packages offer implementations of MCMC, such as `nimble` (de Valpine et al. 2017), `WinBUGS` (Lunn et al. 2000), `jags` (Plummer 2003), `pyMC` (Fonnesbeck et al. 2015), and `Stan` (Carpenter et al. 2017). These packages differ, however, in their approaches to sampler assignments. As sampling techniques vary in computation and quality of the samples, the effectiveness of the MCMC algorithms will vary depending on the software and model.

Among MCMC software packages, only `nimble` opens the hood of the sampler assignment process. Users may select any valid assignment of samplers to each parameter, selecting from the suite of samplers provided with `nimble`. These include random walk Metropolis-Hastings sampling (Robert and Casella 1999), slice sampling (Neal 2003), elliptical slice sampling (Murray, Adams, and MacKay 2010), automated factor slice sampling (Tibbits et al. 2014), conjugate sampling (George, Makov, and Smith 1993), and others.

The `nimbleHMC` package provides two implementations of HMC-NUTS sampling for use within `nimble`. Specifically, `nimbleHMC` provides an implementation of the original ("classic") HMC-NUTS algorithm as developed in Hoffman and Gelman (2014), and a more current version of HMC-NUTS sampling identical to that offered in version 2.32.2 of `Stan` (Stan Development Team 2023). The samplers provided in `nimbleHMC` can be assigned to any continuous-valued parameters, and may be used in combination with other samplers provided with `nimble`.

Example

The following example demonstrates fitting a hierarchical model to data using `nimbleHMC`. We use the European Dipper *Cinclus cinclus)* dataset drawn from ecological capture-recapture (*e.g.*, Lebreton et al. 1992; Turek, de Valpine, and Paciorek 2016). Modelling includes both continuous parameters to undergo HMC sampling and discrete parameters which cannot be sampled via HMC. This combination is not supported by software other than `nimbleHMC`.

Individual birds are captured, tagged, and potentially recaptured on subsequent sighting occasions. Data is a 294×7 binary-valued array of capture histories of 294 uniquely tagged birds over 7 years. Model parameters

are detection probability p, and annual survival rates on non-flood years ϕ_1 and flood years ϕ_2. Data is provided in the R package `mra` (McDonald 2018).

```
library(mra)
data(dipper.data)
y <- dipper.data[,1:7]
```

We specify the hierarchical model using uniform priors on the interval $[0, 1]$ for all parameters. Binary-valued latent states $x_{i,t}$ represent the true alive (1) or dead (0) state of individual i on year t. Doing so allows the survival process to be modelled as $x_{i,t+1} \sim \text{Bernoulli}(\phi_{f_t} \cdot x_{i,t})$ where f_t indicates the flood/non-flood history of year t, and observations are modelled as $y_{i,t} \sim \text{Bernoulli}(p \cdot x_{i,t})$.

```
library(nimbleHMC)

code <- nimbleCode({
    phi[1] ~ dunif(0, 1)
    phi[2] ~ dunif(0, 1)
    p ~ dunif(0, 1)
    for(i in 1:N) {
        for(t in (first[i]+1):T) {
            x[i,t] ~ dbern(phi[f[t]] * x[i,t-1])
            y[i,t] ~ dbern(p * x[i,t])
        }
    }
})
```

A `nimble` model object is now built. The argument `buildDerivs = TRUE` affects derivatives of likelihood calculations to be built into the model object to support derivative-based algorithms – here, HMC sampling.

```
Rmodel <- nimbleModel(
    code,
    constants = list(N = nrow(y), T = ncol(y), f = c(1,2,2,1,1,1,1),
                     first = apply(y, 1, which.max)),
    data = list(y = y),
    inits = list(phi = c(0.5, 0.5), p = 0.5, x = array(1, dim(y))),
    buildDerivs = TRUE)
```

Next we create an MCMC configuration object, which specifies the sampling algorithm to be applied to each parameter. By default, `configureMCMC` uses `nimble`'s default sampler assignments of adaptive random walk Metropolis-Hastings (`RW` sampler; Robert and Casella 1999) for each parameter, and a `binary` Gibbs sampler for each $x_{i,t}$ latent state.

```
conf <- configureMCMC(Rmodel)

## RW sampler (3)
##   - phi[]  (2 elements)
##   - p
## binary sampler (848)
##   - x[]  (848 elements)
```

Now we customize the MCMC configuration object to use HMC sampling for the model parameters. `replaceSamplers` replaces the samplers operating on ϕ_1, ϕ_2 and p with the state-of-the-art HMC-NUTS sampler (called the `NUTS` sampler) provided in `nimbleHMC`.

```
conf$replaceSamplers(target = c("phi", "p"), type = "NUTS")
conf$printSamplers(byType = TRUE)

## NUTS sampler (1)
```

```
##   - phi, p
## binary sampler (848)
##   - x[]  (848 elements)
```

Alternatively, the convenience function `configureHMC(Rmodel)` may be used to create an identical MCMC configuration, applying HMC-NUTS sampling to ϕ_1, ϕ_2 and p, and default binary samplers for discrete parameters.

Now we build and compile the MCMC algorithm.

```
Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
Cmcmc <- compileNimble(Rmcmc, project = Rmodel)
```

We execute the MCMC for 20,000 iterations, and discard the initial 10,000 samples as burn-in.

```
set.seed(0)
samples <- runMCMC(Cmcmc, niter = 20000, nburnin = 10000)
```

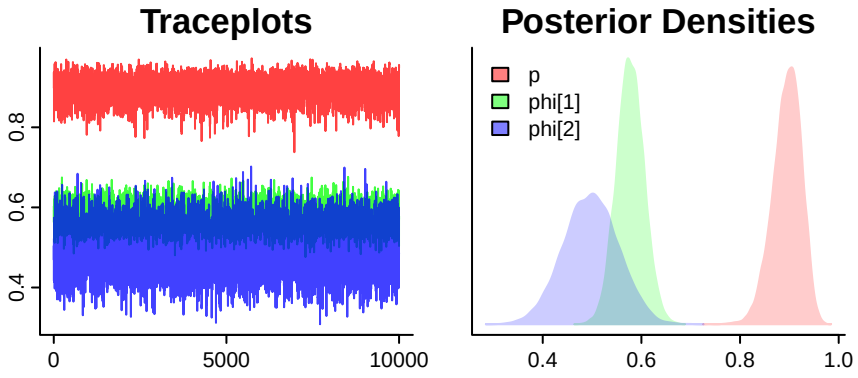
Finally, posterior summary statistics are calculated for the model parameters.

```
samplesSummary(samples, round = 2)
```

```
##        Mean Median St.Dev. 95%CI_low 95%CI_upp
## p      0.89   0.90    0.03      0.83      0.94
## phi[1] 0.58   0.58    0.03      0.52      0.63
## phi[2] 0.50   0.50    0.06      0.39      0.60
```

Traceplots and posterior density plots are generated using the `samplesPlot` function from the `basicMCMCplots` package.

```
basicMCMCplots::samplesPlot(samples, legend.location = "topleft")
```



Statement of need

HMC is recognized as a state-of-the-art MCMC strategy. A testimony to this, software packages such as `Stan` have been built exclusively around HMC sampling. As a result, however, such software cannot operate on models with discrete parameters where HMC cannot operate. Models with discrete parameters arise in a range of statistical motifs including hidden Markov models, finite mixture models, and generally in the presence of unobserved categorical data (Bartolucci, Pandolfi, and Pennoni 2022). In contrast, other mainstream MCMC packages *`WinBUGS`, `OpenBUGS` and `jags`) can sample discrete parameters, but provide no facilities for HMC sampling. This leaves a gap, as there is no support for applying HMC sampling to continuous-valued parameters of hierarchical models which also contain discrete parameters.

`nimbleHMC` fills this gap, by providing an HMC sampler which operates inside `nimble`'s MCMC engine. `nimble` provides a host of MCMC sampling algorithms which are suitable for either continuous or discrete

parameters, as well as the ability to customize an MCMC algorithm by specifying sampler assignments. `nimbleHMC` supplements the suite of sampling algorithms provided with `nimble` with an HMC sampler, which can be used alongside other samplers. The example presented herein demonstrates precisely this use case: HMC sampling operating alongside other discrete samplers, which is not possible without the use of `nimbleHMC`.

It is an open question of what combination of samplers will optimize MCMC efficiency. One metric of comparison is the effective sample size of the samples generated per unit runtime of the algorithm. That is, how quickly an MCMC algorithm generates information about the parameters. This metric is studied in Turek et al. (2017) and Ponisio et al. (2020), but without any conclusive result. For that reason, the ability to mix-and-match samplers from a large pool of candidates is important from both practical and theoretical standpoints. Indeed, packages such as `compareMCMCs` (de Valpine, Paganin, and Turek 2022) are designed to compare the relative performance of MCMC algorithms. The addition of HMC sampling provided by `nimbleHMC` supports new combinations of MCMC algorithms, as well as facilitates a deeper study of practical Bayesian modelling.

References

Bartolucci, Francesco, Silvia Pandolfi, and Fulvia Pennoni. 2022. "Discrete Latent Variable Models." *Annual Review of Statistics and Its Application* 9: 425–52.

Brooks, Steve, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of Markov Chain Monte Carlo*. CRC press.

Carpenter, Bob, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1). https://doi.org/10.18637/jss.v076.i01.

de Valpine, Perry, Sally Paganin, and Daniel Turek. 2022. "CompareMCMCs: An R Package for Studying Mcmc Efficiency." *Journal of Open Source Software* 7 (69): 3844. https://doi.org/10.21105/joss.03844.

de Valpine, Perry, Daniel Turek, Christopher J Paciorek, Clifford Anderson-Bergman, Duncan Temple Lang, and Rastislav Bodik. 2017. "Programming with Models: Writing Statistical Algorithms for General Model Structures with Nimble." *Journal of Computational and Graphical Statistics* 26 (2): 403–13. https://doi.org/10.1080/10618600.2016.1172487.

Fonnesbeck, Chris, Anand Patil, David Huard, and John Salvatier. 2015. "PyMC: Bayesian Stochastic Modelling in Python." *Astrophysics Source Code Library*, ascl–1506.

George, Edward I, UE Makov, and AFM Smith. 1993. "Conjugate Likelihood Distributions." *Scandinavian Journal of Statistics*, 147–56.

Hoffman, Matthew D, and Andrew Gelman. 2014. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.* 15 (1): 1593–1623.

Lebreton, Jean-Dominique, Kenneth P Burnham, Jean Clobert, and David R Anderson. 1992. "Modeling Survival and Testing Biological Hypotheses Using Marked Animals: A Unified Approach with Case Studies." *Ecological Monographs* 62 (1): 67–118. https://doi.org/10.2307/2937171.

Lunn, David J, Andrew Thomas, Nicky Best, and David Spiegelhalter. 2000. "WinBUGS-a Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing* 10: 325–37.

McDonald, Trent. 2018. *Mra: Mark-Recapture Analysis*. https://CRAN.R-project.org/package=mra.

Murray, Iain, Ryan Adams, and David MacKay. 2010. "Elliptical Slice Sampling." In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 541–48. JMLR Workshop; Conference Proceedings.

Neal, Radford M. 2003. "Slice Sampling." *The Annals of Statistics* 31 (3): 705–67. https://doi.org/10.1214/aos/1056562461.

Plummer, Martyn. 2003. "JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, 124:1–10. 125.10. Vienna, Austria.

Ponisio, Lauren C, Perry de Valpine, Nicholas Michaud, and Daniel Turek. 2020. "One Size Does Not Fit All: Customizing Mcmc Methods for Hierarchical Models Using Nimble." *Ecology and Evolution* 10 (5): 2385–2416. https://doi.org/10.1002/ece3.6053.

Robert, Christian P, and George Casella. 1999. "The Metropolis—Hastings Algorithm." In *Monte Carlo Statistical Methods*, 231–83. Springer.

Stan Development Team. 2023. "Stan Modeling Language Users Guide and Reference Manual, Version 2.32.2." https://mc-stan.org.

Tibbits, Matthew M, Chris Groendyke, Murali Haran, and John C Liechty. 2014. "Automated Factor Slice Sampling." *Journal of Computational and Graphical Statistics* 23 (2): 543–63. https://doi.org/10.1080/10618600.2013.791193.

Turek, Daniel, Perry de Valpine, and Christopher J Paciorek. 2016. "Efficient Markov Chain Monte Carlo Sampling for Hierarchical Hidden Markov Models." *Environmental and Ecological Statistics* 23: 549–64. https://doi.org/10.1007/s10651-016-0353-z.

Turek, Daniel, Perry de Valpine, Christopher J Paciorek, and Clifford Anderson-Bergman. 2017. "Automated Parameter Blocking for Efficient Markov Chain Monte Carlo Sampling." *Bayesian Analysis* 12 (2): 465–90. https://doi.org/10.1214/16-BA1008.