Introduction to SamplerCompare

Madeleine B. Thompson
University of Toronto

Abstract

SamplerCompare is an R package for comparing the performance of Markov Chain Monte Carlo samplers. It samples from a collection of distributions with a collection of MCMC methods over a range of tuning parameters. Then, using log density evaluations per independent observation as a figure of merit, it generates a grid of plots showing the results of the simulation. It comes with a collection of predefined distributions and samplers and provides R and C interfaces for defining additional ones. This document demonstrates the basics of running simulations, visualizing results, and defining distributions and samplers in R.

Keywords: MCMC, visualization.

1. Purpose of package

SamplerCompare is an R package that allows for automated comparison of MCMC methods. It samples from collections of probability distributions with collections of Markov Chain Monte Carlo samplers with a range of tuning parameters and presents the results of such simulations graphically. These comparisons allow researchers to better understand which MCMC methods perform best in which circumstances.

This document introduces the mechanics of using the **SamplerCompare** package. For the mathematical background of the comparisons and analysis of the resulting graphics, see Thompson (2010). Other sources of information on **SamplerCompare** are the R online help and the vignette "R/C Glue in SamplerCompare." A list of online help topics and vignettes can be found by typing:

```
> library(help='SamplerCompare')
```

Vignettes can be read with the `vignette` command. For example:

```
> vignette('glue')
```

PDF copies can be found in the `doc` directory of the installed package.

2. Running MCMC simulations

The three central types of objects in **SamplerCompare** are distributions (which have the class `dist`), sampler functions, and simulation results. The function `compare.samplers` runs a

R function	Sampler
`multivariate.metropolis.sample`	Metropolis–Hastings with spherically symmetric Gaussian proposals
`univar.metropolis.sample`	Metropolis–Hastings with single-coordinate updates
`adaptive.metropolis.sample`	Adaptive Metropolis–Hastings (Roberts and Rosenthal 2009)
`arms.sample`	Adaptive Rejection Metropolis (Gilks, Best, and Tan 1995)
`stepout.slice.sample`	slice sampler with stepping out (Neal 2003, §4)
`interval.slice.sample`	slice sampler without stepping out (Neal 2003, §4)
`hyperrectangle.sample`	slice sampler with hypercube for initial slice approximation, shrinkage using gradient (Neal 2003, §5.1)
`nograd.hyperrectangle.sample`	slice sampler with hypercube for initial slice approximation, shrinkage in all dimensions (Neal 2003, §5.1)
`nonadaptive.crumb.sample`	slice sampler with Gaussian crumbs (Neal 2003, §5.2)
`cov.match.sample`	covariance-matching slice sampler (Thompson and Neal 2010, §4)
`shrinking.rank.sample`	shrinking rank slice sampler (Thompson and Neal 2010, §5)

Figure 1: Predefined samplers; see the R help for the sampler's R function for more information on an individual method.

R symbol	Distribution
`N2weakcor.dist`	weakly correlated two-dimensional Gaussian
`N4poscor.dist`	strongly positively correlated four-dimensional Gaussian
`N4negcor.dist`	strongly negatively correlated four-dimensional Gaussian
`schools.dist`	ten-dimensional multilevel model (Gelman, Carlin, Stern, and Rubin 2004, pp. 138–145)
`funnel.dist`	ten-dimensional distribution with funnel-shaped marginals (Neal 2003, p. 732)

R function	Distributions generated
`make.gaussian`	multivariate Gaussians
`make.cone.dist`	distributions with cone-shaped log density (Roberts and Rosenthal 2002)
`make.multimodal.dist`	mixtures of standard Gaussians
`make.mv.gamma.dist`	distributions with uncorrelated gamma marginals

Figure 2: Predefined distributions and functions that generate distributions; see the R help for a symbol for more information on an individual distribution or generator.

list of samplers on a list of distributions with a set of tuning parameters and returns a data frame containing simulation results. Sampler functions are assumed to have a single scalar tuning parameter. If they have more, wrapper functions can be used to represent a single sampler with a varying tuning parameter as multiple samplers. **SamplerCompare** comes with a collection of predefined samplers (listed in figure 1) and distributions (listed in figure 2).

Suppose we would like to compare Adaptive Metropolis (`adaptive.metropolis.sample`) and Adaptive Rejection Metropolis (`arms.sample`) with the tuning parameters 1, 20, and 400 on two-dimensional Gaussian (`make.gaussian`) and Gamma (`make.mv.gamma.dist`) distributions. We can do this with `compare.samplers` using the code:

```
> library(SamplerCompare)
> gauss.cor7 <- make.gaussian(mean = c(1, 2), rho = 0.7)
> gamma.shape23 <- make.mv.gamma.dist(shape = c(2, 3))
> sim.results <- compare.samplers(sample.size = 1000, dists = list(gauss.cor7,
+     gamma.shape23), samplers = list(adaptive.metropolis.sample,
+     arms.sample), tuning = c(1, 20, 400))
```

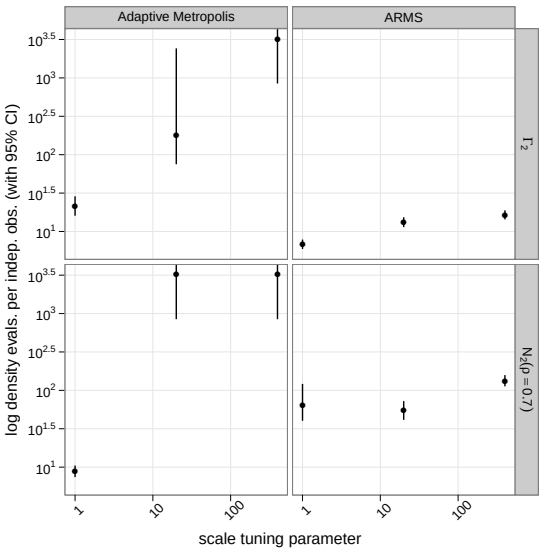
```
N2,rho=0.7 Adaptive Metropolis: 8.77 (7.42,10.5) evals tuning=1
N2,rho=0.7 Adaptive Metropolis: 3.2e+03 (845,Inf) evals tuning=20
N2,rho=0.7 Adaptive Metropolis: 3.2e+03 (845,Inf) evals tuning=400
N2,rho=0.7 ARMS: 64.4 (40.1,121) evals tuning=1
N2,rho=0.7 ARMS: 54.7 (41.3,72.4) evals tuning=20
N2,rho=0.7 ARMS: 133 (113,158) evals tuning=400
Gamma2 Adaptive Metropolis: 21.1 (16,28.7) evals tuning=1
Gamma2 Adaptive Metropolis: 178 (75.1,2.42e+03) evals tuning=20
Gamma2 Adaptive Metropolis: 3.2e+03 (845,Inf) evals tuning=400
Gamma2 ARMS: 6.77 (5.92,7.86) evals tuning=1
Gamma2 ARMS: 13.1 (11.4,15.3) evals tuning=20
Gamma2 ARMS: 16.4 (14.4,18.9) evals tuning=400
```

Each line in the trace output has the distribution name, the sampler name, the number of evaluations per independent observation with 95% confidence interval in parentheses, and the tuning parameter.

3. Visualizing results

To visualize the results from a simulation, one can use the `comparison.plot` function. It has a single required argument, a data frame containing results from `compare.samplers`, and returns a **ggplot2** plot object. One can call `print` on this object to view the plot; it can also be edited with the **grid** package. To plot the results from the previous example, one would type:

```
> print(comparison.plot(sim.results))
```



In this graphic, the columns of plots represent the samplers and the rows of plots represent the distributions. The vertical axis in each plot is the number of log density evaluations per independent observation; see the help for `ar.act` for more information on how this is computed. The horizontal axis is the scalar tuning parameter. The vertical bars are approximate 95% confidence intervals for the figure of merit.

4. Defining additional samplers

MCMC samplers are specified by functions that have the signature:

```
sampler(target.dist, x0, sample.size, tuning)
```

They must also have a `name` attribute, a length-one character vector. The `target.dist` parameter specifies the target distribution; see the R help for `make.dist` for details on its structure. `x0` specifies the start state for the simulation, `sample.size` specifies the sample size, and `tuning` specifies a scalar tuning parameter.

A sampler function should return a list with two elements: `X`, a matrix of rows of observations, and `evals`, a count of the number of times it evaluated the log density (with `target.dist$log.density`). If the sampler evaluates the gradient of the log density (with `target.dist$grad.log.density`), the list should contain a `grads` element, indicating the number of times it did this.

The following code specifies a Metropolis sampler with multivariate proposals:

```
> metropolis.sample <- function(target.dist, x0, sample.size, tuning) {
+     X <- matrix(nrow = sample.size, ncol = target.dist$ndim)
+     state <- x0
+     evals <- 1
+     state.log.dens <- target.dist$log.density(state)
+     for (obs in 1:sample.size) {
+         proposal <- rnorm(target.dist$ndim, state, tuning)
```

```
+         evals <- evals + 1
+         proposal.log.dens <- target.dist$log.density(proposal)
+         if (runif(1) < exp(proposal.log.dens - state.log.dens)) {
+             state <- proposal
+             state.log.dens <- proposal.log.dens
+         }
+         X[obs, ] <- state
+     }
+     return(list(X = X, evals = evals))
+ }
> attr(metropolis.sample, "name") <- "Metropolis"
```

See the R help for `compare.samplers` for more information on writing samplers in R. See the R help for `wrap.c.sampler` and the vignette "R/C Glue in SamplerCompare" for more information on writing samplers in C.

5. Defining additional distributions

`make.dist` can be used to specify a distribution whose log density is expressed in R. (See the R help for `make.c.dist` and the vignette "R/C Glue in SamplerCompare" for more information on specifying distributions in C.) Its most important arguments are `ndim`, `name`, and `log.density`. `ndim` specifies the dimension of the distribution and `name` names the distribution. `log.density` is a function of one vector argument of length `ndim` that returns the log density at that point; it should return `-Inf` if the point is outside the support of the distribution. The log density does not need to be normalized.

The following code defines a Beta(2,3) distribution:

```
> beta23.log.dens <- function(x) ifelse(x < 0 | x > 1, -Inf, log(x) +
+     2 * log(1 - x))
> beta23.dist <- make.dist(ndim = 1, name = "Beta(2,3)", log.density = beta23.log.dens,
+     mean = 2/(2 + 3))
```

The optional `mean` argument to `make.dist` makes the autocorrelation time computation in `compare.samplers` more accurate, so it is advisable to specify it when the mean is known.

6. A final example

Samplers and distributions defined as above can be used directly:

```
> sim <- metropolis.sample(beta23.dist, x0 = 0.5, sample.size = 100,
+     tuning = 1)
> summary(sim$X)

       V1
 Min.   :0.09057
 1st Qu.:0.25804
```

```
 Median :0.44070
 Mean   :0.43395
 3rd Qu.:0.60327
 Max.   :0.77242
```

Or, they can be passed to `compare.samplers`:

```
> sim.results <- compare.samplers(sample.size = 100, dists = list(beta23.dist),
+     samplers = list(metropolis.sample), tuning = c(0.1, 1))

Beta(2,3) Metropolis: 17.8 (8.71,115) evals tuning=0.1
Beta(2,3) Metropolis: 7.96 (4.5,19.3) evals tuning=1
```

Note the large confidence intervals for the evaluations per independent observation; this is a sign that a larger `sample.size` should be used.

7. Limitations

SamplerCompare was created to support my own research; I am publishing it with the hope that others find it useful. Some current limitations include:

- Distributions are assumed to be continuous and to be of a constant dimension.
- All simulations start at a random point on the unit hypercube.
- Samplers are assumed to have exactly one scalar tuning parameter.
- All samplers in a given invocation of `compare.samplers` are run with the same simulation length and set of tuning parameters.
- There is no explicit support for multithreading. (This, and the previous limitation, can be worked around manually by using `rbind` on the results of multiple calls to `compare.samplers`.)
- Distributions are defined entirely in terms of their log density; there is no way to specify that a distribution is unimodal or that a particular parameter is always positive.

References

Gelman A, Carlin JB, Stern HS, Rubin DB (2004). *Bayesian Data Analysis, Second Edition*. Chapman and Hall/CRC. URL http://www.stat.columbia.edu/~gelman/book/.

Gilks WR, Best NG, Tan KKC (1995). "Adaptive Rejection Metropolis Sampling within Gibbs Sampling." *Applied Statistics*, **44**(4), 455–472. URL http://www.jstor.org/stable/2986138.

Neal RM (2003). "Slice sampling." *Annals of Statistics*, **31**, 705–767. URL http://projecteuclid.org/getRecord?id=euclid.aos/1056562461.

Roberts GO, Rosenthal JS (2002). "The Polar Slice Sampler." *Stochastic Models*, **18**(2), 257–280. URL http://www.informaworld.com/openurl?genre=article&issn=1532%2d6349&volume=18&issue=2&spage=257.

Roberts GO, Rosenthal JS (2009). "Examples of Adaptive MCMC." *Journal of Computational and Graphical Statistics*, **18**(2), 349–367. URL http://pubs.amstat.org/doi/abs/10.1198/jcgs.2009.06134.

Thompson MB (2010). "Graphical Comparison of MCMC Performance." In preparation.

Thompson MB, Neal RM (2010). "Covariance-Adaptive Slice Sampling." *Technical Report TR-1002*, Dept. of Statistics, University of Toronto. URL http://www.cs.toronto.edu/~radford/cass.abstract.html.

Affiliation:

Madeleine B. Thompson
Dept. of Statistics, University of Toronto
100 St. George Street Room 6022
Toronto, Ontario, M5S 3G3, Canada
E-mail: mthompson@utstat.toronto.edu
URL: http://www.utstat.toronto.edu/mthompson