Algorithms for population-based samplers

Lawrence Middleton Simon Lyddon
Mathias C. Cronjäger

February 26, 2015

Abstract

This vingnette gives a brief overview of the algoirithms implemented in the package `$PACKAGENAME`. This covers a brief descriprion of how the algoritm works, a test-case, as well as a brief overview of how some of the functions provided by the package can be used together.

1 Test Case – A multimodal Mixture

Multimodal distributions of the form

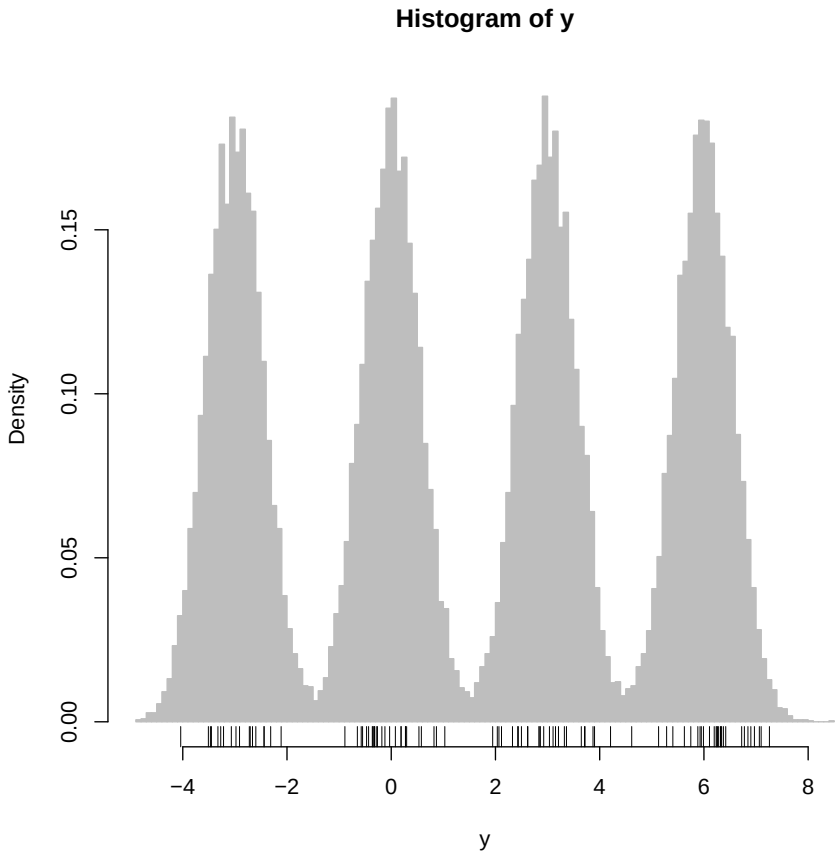
$$f(x) \sim \sum_{i=1}^{N} \omega_i \mathcal{N}(x; \mu_i, \sigma_i) \tag{1}$$

are a simple class of probability-distributions for which a lot of basic MCMC algoritms fail. Any algorithm relying on local optimization will tend to converge erroneously on a single mode. Since these distributions are still rather easy to work with, they are commonly used as test-cases for methods that are designed to capture global information (if calibrated correctly), such as multimodality. For the samplers implemented and benchmarked, $N = 4$ has been chosen and $\omega_i = \frac{1}{4}$, $\sigma_i = 0.55$ is fixed. The observarions upon which we attempt to infer the modes are sampled from $\mu = (-3, 0, 3, 6)$.

The function `sampleMM`[1] returns samples from the distribution given by (1).

```
sampleMM(10)

##  [1]  1.59466815  6.96537740  3.20651156 -3.81165034  5.80876568
##  [6] -0.07062868 -3.25535168  2.31018138  3.17764683  0.45135217

y <- sampleMM(2^15)
hist(y,breaks = 100,freq = F,
     col="grey",border="grey",bty="n")
rug(y[1:100])
```

[1] defined in `mixtureModel_SMC.R`



2 Sequential Importance Sampling – Generalized Sequential monte carlo samplers

The classical use-case for Sequential Monte Carlo samplers is to sample from the marginal distribution(s) of a stochastic process, by propagaring a set of wheighted particles forwards in time according to some transition-kernel and reweighing/resampling particles based on their importance-weights. Such a case is for instance given by the stochastic volatility modell outlined in (Lee et al., 2010, section 4.2).

As outlined in (Lee et al., 2010, section 3.3) and Del Moral et al. (2006), we can however use Sequential Monte Carlo methods in a clever way to sample from a wide variety of distributions that do not fit the above mold. This more general approach often referred to as Sequential Importance Sampling involves sampling from some measure of interest π by constructing a sequence of measures that $(\pi^{(n)})_{n=1\ldots N}$ such that $\pi^{(N)} = \pi$. For the SMC-approach to work $\pi^{(1)}$ should be

easy to sample from and need to be able to transform a sample from π^i into a sample from $\pi^{(i+1)}$ for $i = 1 < N$, as well as a mechanism for updating weights of samples.

An example where this approach might be fruitfull would be the sampleing from a highly localized (potentially multimodal) posterior $p(\theta|y_{1:M})$ of some non-localised prior by lettting $\pi^{(i)} := p(\theta|y_{1:n_i})$ for some sequence $0 < n_1 < \ldots < n_N = M$ chosen so that two consecutive distributions are always "sufficiently similar".

Another example is *simulated anealling*, whereby we consider smoothed versions of some target-density π is given by $\pi^{(i)} \propto \pi^{\beta_i}$ for some sequence $0 \leq \beta_1 < \ldots < \beta_N = 1$. The algoritms implemented in `mixtureModel_SMC.R` and `SMC_sampler.c` such an approach to the test case of inferring the modes of a mixture of 4 otherwise equal normal distributions. In the algoritms, $\beta_n := \left(\frac{n}{M}\right)^2$ has been chosen for $1 = \ldots = M$. Particles are propagated by taking 10 Metropolis hastings steps with a gaussian proposal, and weights are updated as follows. A particle x in generation n, is assigned the unnormalzed weight

$$W(x_n) := w(\tilde{x}_n) * \frac{\pi^{(n)}(\tilde{x}_n)}{\pi^{(n-1)}(\tilde{x}_n)}$$

where $\tilde{x}_n$ denotes the ancestor in generation $n-1$ of the particle x_n (after resampling has ocurred).

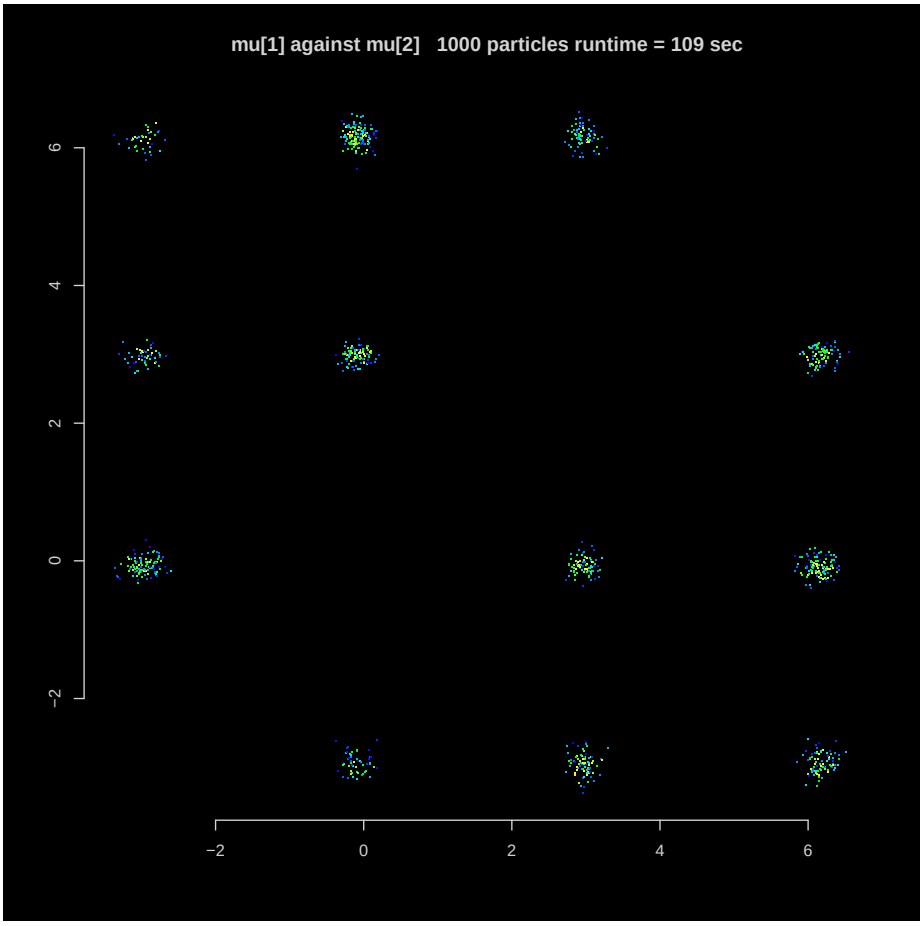
The sheme outlined above has been implemented for infering the modes of a distribution as in (1). The scheme has been implemented in both R and C. Both implementations are made available to the end-user via R functions. If we were willing to wait for a while (the R code is a bit slow), we could compare the speed of the rwo implementations as follows:

```
y <- sampleMM(100)
N_particles <- 1000
out_slow <- SMC_sampler(y,N_particles)
out_FAST <- SMC_sampler_FAST(y,N_particles)
print(paste("t_slow =",out_slow$runTime,"sec"))
print(paste("t_fast =",out_FAST$runTime,"sec"))
```

Since `SMC_sampler_FAST` is just a wrapper for c code, a brief glance at the source-code in `/src/SMC_samplder.c` is worth it for seeing what goes on under the hood.

We can get an overview of how the samples generated are distributed using the following modified plotting command which accepts the output of any of the samplers above as valid input.

```
y <- sampleMM(100)
N_particles <- 1000
out_FAST <- SMC_sampler_FAST(y,N_particles)
generatePlot(SMC_Data = out_FAST)
```

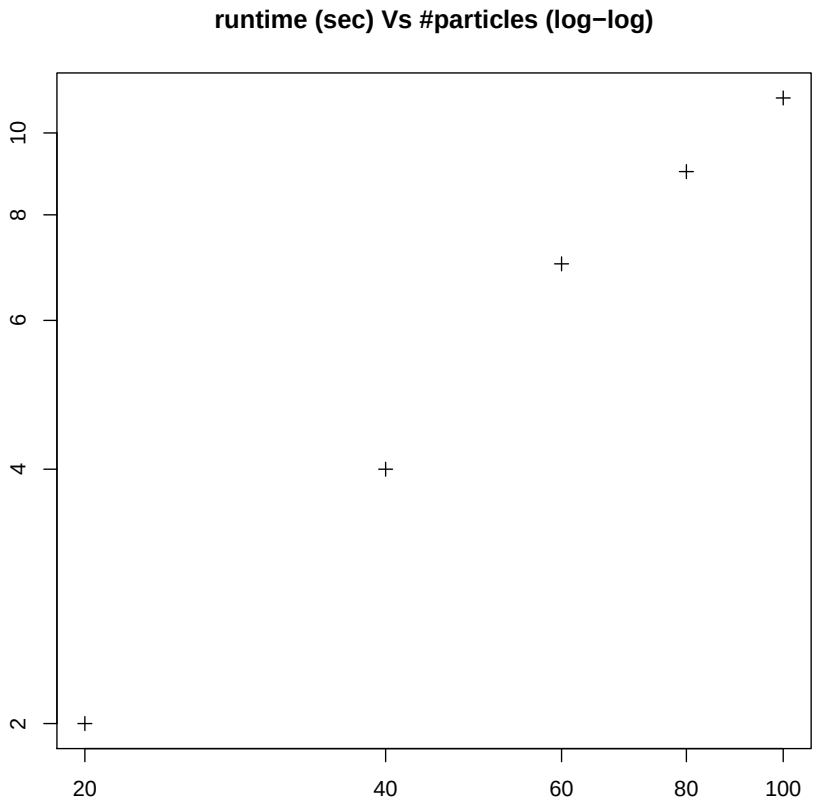


As the above plot makes evident, the SMC sampler samples from all modes of the posterior $p(\mu|y)$. Since the weights and variances are presumed to be equal, the posterior distribution on modes is exchangeable i.e. invariant w.r.t permutation.

Further benchmarking functions are availiable.

```
#prints time in secconds for each N in N_seq. stores runtimes as a vector
times <- benchmark_SMC(N_seq = 20*1:5)

## [1] "N = 20 t = 2"
## [1] "N = 40 t = 5"
## [1] "N = 60 t = 6"
## [1] "N = 80 t = 9"
## [1] "N = 100 t = 11"

times

## [1]  2  5  6  9 11
```

```
#runs the above and prints a log-log-plot of runtime vs number of particles
generateRuntimePlot(N_seq = 20*1:5)

## [1] "N = 20 t = 2"
## [1] "N = 40 t = 4"
## [1] "N = 60 t = 7"
## [1] "N = 80 t = 9"
## [1] "N = 100 t = 11"
```



References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, June 2010. ISSN 13697412. doi: 10.1111/j.1467-9868.2009.00736.x.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, June 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x.

Anthony Lee, Christopher Yau, Michael B. Giles, Arnaud Doucet, and Christopher Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, (February):769–789, 2010. ISSN 1061-8600. doi: 10.1198/jcgs.2010.10039.

Marc A Suchard, Quanli Wang, Cliburn Chan, Jacob Frelinger, Andrew Cron, and Mike West. Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures. *Journal of Computational and Graphical Statistics*, 19(February):419–438, 2010. ISSN 1061-8600. doi: 10.1198/jcgs.2010.10016.