|  |
| --- |
| Advanced Statistical Computing Course - Fall 2015  **Improved Sampling and Parallelization of a Sampling Importance Resampling (SIR) Procedure**  Project by Anne-Gaëlle Dosne |

**Aim**

Increase the efficiency of SIR (Sampling Importance Resampling), a method to obtain parameter uncertainty, by improving sampling and implement parallelization.

**Background**

This method was originally developed in a Bayesian context [ref] and implemented to estimate parameter uncertainty for nonlinear mixed effects models [ref]. The idea behind SIR is to sample *M* p-dimensional parameter vectors *θ* from a p-dimensional proposal distribution *hprior(θ)*, compute weights (also called importance ratios *IR*) based on the vectors’ adequacy to the data relative to their likelihood in *h(θ)*, and resample *N* vectors based on their *IR*. The resampled vectors form the posterior density *hpost(θ).* This procedure can be iterated a number of times, using *hpost(θ)* of one iteration as the as the input proposal for the next.

The idea is to investigate different sampling strategies for generating *M* samplesfrom the proposal distribution *hprior(θ)* and implement computation of the *IR* in parallel.

**Methods**

All work was done in R. A very simple simulated example was be used, where *hpost(θ)* was assumed to be known. Samples were generated from different proposals and the number of iterations (at fixed *M* and *m*) and time for *hpost(θ)* to be reached was compared to evaluate the performance of the sampling strategies. The speed increase when using parallelization for IR computation was quantified.

Investigated sampling strategies will be random sampling (Monte-Carlo, MC) and Latin Hypercube Sampling (LHS). Parallelization was be investigated locally using the parallel library. Parallelization using systems like Hadoop was not considered here as the data to handle was very limited in size and thus the additional benefit of such approaches was deemed too little.

The example assumed a 5-dimensional multivariate distribution. The 5 variables were assumed to have a mean of 0, a variance of 1 and no correlations between them (“true” distribution). *M*=5000 samples were then generated using MC and LHS from 8 different proposal distributions. The proposal distributions were the true distribution, inflations thereof (variances multiplied by 2 and 10), deflations thereof (variances divided by 2 and 10), shifts thereof (means shifted by 1 and 2) and a shifted inflation (means shifted by 1 and variances multiplied by 2). *IR* were computed as the ratio between the density of the parameter vector in the true distribution divided by its density in the proposal distribution. *N*=1000 vectors were then resampled based on their *IR*. The empirical covariance matrix of these vectors was computed as used as the proposal distribution for the next SIR iteration. This process was repeated 5 times. The means, variances and CI95% of each variable were computed for each iteration to investigate whether LHS was able to find the true distribution faster than MC. Additional investigations were done for selected options by introducing correlations in the true distribution or reducing the number of samples/resamples with and without changing the *M/N* ratio. Table 1 presents the summary of the investigated options.

**Table1** Investigated options for the SIR procedure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Options** | | | | |
| **True distribution** | **Proposal distribution** | | **Sampling** | **Parallelization** | **M/N** |
| MVN (0,I) | *true* | MVN (0,I) | MC | yes | 5000/1000 |
| MVN (0,corrI)\* | *inflation 1* | MVN (0,2\*I) | LHS | no | 2000/1000\* |
|  | *inflation 2* | MVN (0,10\*I) |  |  | 100/20\* |
|  | *deflation 1* | MVN (0,0.5\*I) |  |  |  |
|  | *deflation 2* | MVN (0,0.1\*I) |  |  |  |
|  | *bias 1* | MVN (1, I) |  |  |  |
|  | *bias 2* | MVN (2,I) |  |  |  |
|  | *infl+bias* | MVN (1,2\*I) |  |  |  |
| \*additional investigations, for selected options only. Correlations of 0.5 or -0.5 were assumed between parameters 1 and 2 and parameters 4 and 5. | | | | | |

**Results**

MC sampling was implemented in R using the *sample* function. LHS using the *randomLHS* function from the *lhs* package.

The benefit of as sampling method was judged based on the number of iterations needed to reach the true value for the mean and variances. From the 15 tested scenarios, LHS outperformed MC in 2 cases and performed similarly in the remaining 13 (Figure 1). LHS was better than MC in cases where the proposal was shifted by 2 compared to the true (BIAS2: true center at 0 at 4th iteration for LHS vs. 0.65 at 5th iteration for MC) and where the number of samples/resamples was low (DEFL2a: true variance at 1 at 5th iteration for LHS vs. 0.55 at 5th iteration for MC). LHS and MC performed similarly under deflations and inflations of the proposal when *M/N*= 5000/1000 (and 2000/1000, DEFL2b) with 3-5 and 1-2 iterations needed until convergence of means and variances, respectively. In the presence of correlations of ±0.5, they performed similarly in terms of mean and variance convergence but MC outperformed LHS in terms of covariance convergence (Figure 2). Note that none of the methods was able to find the true correlation value after 5 iterations. Regarding execution time, using LHS instead of MC did not add computing times (Table 2).

**Table 2 Results from microbenchmark between sampling strategies (milliseconds): LHS does not take longer than MC.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Scenario** | **min** | **lq** | **mean** | **median** | **uq** | **max** | **eval** |
| MVN\_TRUE\_MC\_NOPAR | 26.8 | 27.8 | 29.3 | 28.7 | 30.6 | 40.0 | 100 |
| MVN\_TRUE\_LHS\_NOPAR | 26.8 | 27.8 | 29.2 | 28.3 | 29.7 | 38.7 | 100 |
| Runtimes are compared for 1 SIR iteration, M=100 and N=20. | | | | | | | |

It was then investigated whether runtime could be optimized. Code profile using *lineprof* was tried in order to determine the bottlenecks in the code (*sir\_function.R*). However, I did not manage to get it to work.

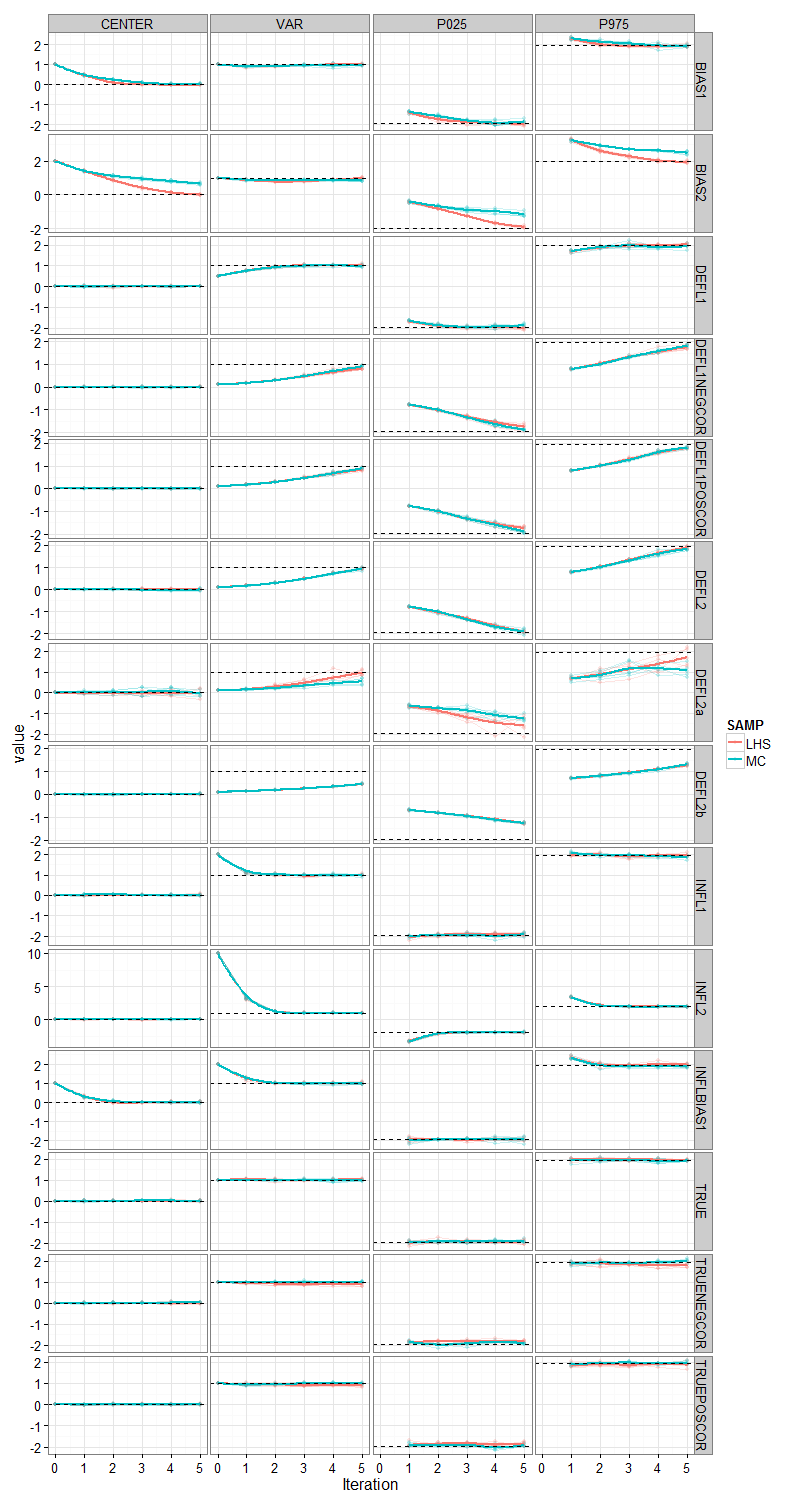
To speed up the code, 2 options were considered: byte compiling using *cmpfun* and parallelization using the *parallel* library. For the parallelization, it was thought that the most computer-intensive task was the computation of the IR, so this part was parallelized.

**Table 3 Results from microbenchmark between implementations (milliseconds): parallel XX than original function, no advantage of byte compiling.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Scenario** | **min** | **lq** | **mean** | **median** | **uq** | **max** | **eval** |
| MVN\_TRUE\_MC\_NOPAR | 27.4 | 28.0 | 29.2 | 28.4 | 30.9 | 34.6 | 100 |
| Cmpfun(MVN\_TRUE\_LHS\_NOPAR) | 27.1 | 28.0 | 28.8 | 28.3 | 28.9 | 33.2 | 100 |
| MVN\_TRUE\_MC\_PAR |  |  |  |  |  |  |  |
| Runtimes are compared for 1 SIR iteration, M=100 and N=20. | | | | | | | |

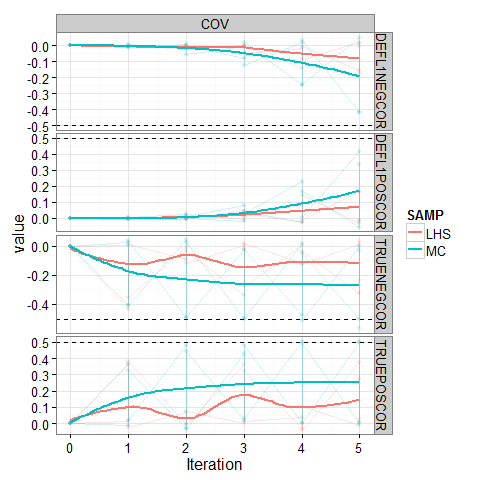
**Figure 1 Convergence of estimated means, variances and CI95% bounds for investigated scenarios: LHS outperforms MC in 2 out of 15 scenarios only**

Full thick lines are smooths of means, variances and CI95% bounds of the 5 parameters over SIR iterations using MC sampling (blue) and LHS (red). Full thin lines and points correspond to the data for each parameter. Dotted horizontal lines correspond to the true values.



**Figure 2 Convergence of estimated covariances for selected scenarios: MC closer to true value than LHS**

Full lines are smooths of covariances over SIR iterations using MC sampling (blue) and LHS (red) for the 5 scenarios in which the true distribution displayed correlations. Full thin lines and points correspond to the data for each parameter pair. Dotted horizontal lines correspond to the true values.



**Discussion and conclusions**

Using LHS did not prove beneficial over MC at the *M*=5000 and 2000 sample sizes. These sample sizes were chosen as they were in the range of what had been previously implemented for obtaining parameter uncertainty in NLMEM. It appears however that sample size and thus runtime could be reduced by reducing the number of samples while using LHS, as convergence at *M*=100 and *N*=20 is almost as good as that with *M*=5000 and *N*=1000 with LHS but worse with MC. LHS performed however less well than MC at identifying the correlation structure when it had been misspecified (only underestimation was considered here), which should be considered.

**Sampling multivariate distributions**

Generic methods for generating [independent](https://en.wikipedia.org/wiki/Statistical_independence) samples:

* [Rejection sampling](https://en.wikipedia.org/wiki/Rejection_sampling) for arbitrary density functions
* [Inverse transform sampling](https://en.wikipedia.org/wiki/Inverse_transform_sampling) for distributions whose CDF is known
* [Slice sampling](https://en.wikipedia.org/wiki/Slice_sampling)
* [Ziggurat algorithm](https://en.wikipedia.org/wiki/Ziggurat_algorithm), for monotonously decreasing density functions as well as symmetric unimodal distributions
* [Convolution random number generator](https://en.wikipedia.org/wiki/Convolution_random_number_generator), not a sampling method in itself: it describes the use of arithmetics on top of one or more existing sampling methods to generate more involved distributions.

Generic methods for generating [correlated](https://en.wikipedia.org/wiki/Correlated) samples (often necessary for unusually-shaped or high-dimensional distributions):

* [Markov chain Monte Carlo](https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo), the general principle
* [Metropolis–Hastings algorithm](https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm)
* [Gibbs sampling](https://en.wikipedia.org/wiki/Gibbs_sampling)
* [Slice sampling](https://en.wikipedia.org/wiki/Slice_sampling)
* [Reversible-jump Markov chain Monte Carlo](https://en.wikipedia.org/wiki/Reversible-jump_Markov_chain_Monte_Carlo), when the number of dimensions is not fixed (e.g. when estimating a [mixture model](https://en.wikipedia.org/wiki/Mixture_model) and simultaneously estimating the number of mixture components)
* [Particle filters](https://en.wikipedia.org/wiki/Particle_filter), when the obse
* The maximum number of combinations for a Latin Hypercube of M divisions and N variables (i.e., dimensions) can be computed with the following formula:
* \left(\prod_{n=0}^{M-1} (M-n)\right)^{N-1} = (M!)^{N-1}
* In **random sampling** new sample points are generated without taking into account the

previously generated sample points. One does not necessarily need to know beforehand how many sample points are needed.

* In **Latin Hypercube sampling** one must first decide how many sample points to use and for each sample point remember in which row and column the sample point was taken.
* In **Orthogonal sampling**, the sample space is divided into equally probable subspaces. All sample points are then chosen simultaneously making sure that the total ensemble of sample points is a Latin Hypercube sample and that each subspace is sampled with the same density.

Thus, orthogonal sampling ensures that the ensemble of random numbers is a very good representative of the real variability, LHS ensures that the ensemble of random numbers is representative of the real variability whereas traditional random sampling (sometimes called brute force) is just an ensemble of random numbers without any guarantees.