|  |
| --- |
| 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 | | | | | |

**Results**

**Table 2 Results from microbenchmark (seconds): LHS does not take longer than MC**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Scenario** | **min** | **lq** | **mean** | **median** | **uq** | **max** | **eval** |
| MVN\_TRUE\_MC\_NOPAR | 6.75 | 7.65 | 8.71 | 8.52 | 9.79 | 12.90 | 100 |
| MVN\_TRUE\_LHS\_NOPAR | 6.60 | 7.46 | 8.71 | 8.34 | 9.74 | 15.01 | 100 |

**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.