Reducing computation time in the R-package 'BayLum'

Frederik Baumgarten, Anne Philippe, Guérin Guillaume, Sebastian Kreutzer

Ancient TL

Reducing computation time in the R-package 'BayLum'

Frederik Baumgarten[1*], Anne Philippe[2], Guillaume Guérin[3], Sebastian Kreutzer[4,5]

[1] Department of Physics, Technical University of Denmark, DTU Risø Campus, Denmark
[2] Laboratoire de Mathématiques Jean Leray, Université de Nantes, 2 rue de la Houssinière, BP 92208 44322 Nantes Cedex 3, France
[3] Université de Rennes, CNRS, Géosciences Rennes, UMR 6118, 35000 Rennes, France
[4] Institute of Geography, Ruprecht-Karl University of Heidelberg, 69120 Heidelberg, Germany
[5] Archéosciences Bordeaux, UMR 6034, CNRS-Université Bordeaux Montaigne, Pessac, France

*Corresponding Author: fhaba@dtu.dk

Received: April 21, 2023; in final form: June 13, 2023

Abstract

'BayLum' is an R-package that facilitates the application of Bayesian models to the field of OSL dating. Here we present two recent feature updates to 'BayLum', significantly reducing computation time and improving general use. The first feature allows users to parallelize the computations involved in the MCMC sampling of values, while the second introduces the ability to extend a 'BayLum' model, which has run to completion without converging. All updates are automatically available with 'BayLum' v0.3.1.

Keywords: Age model, Chronology, MCMC algorithm, Luminescence dating, OSL

1. Introduction

'BayLum' is an R – package (R Core Team, 2022) that gives users the tools to easily apply the Bayesian models presented in Combès et al. (2015) and Combès & Philippe (2017) to luminescence dating data. See, for example, the work of Heydari et al. (2020), where an OSL chronology is provided for the paleolithic site of Mirak, Iran, using 'BayLum'. In this work, they showed that the age uncertainty can be reduced significantly by imposing stratigraphic order – a feature of 'BayLum'. Since the introduction of 'BayLum' (Philippe et al., 2019), 'BayLum' has grown by drawing resources from the ever-developing R-landscape around it. The latest iteration of 'BayLum' (v0.3.1) (Christophe et al., 2023) now employs `'runjags'` (Denwood, 2016) as the R to JAGS (Plummer, 2003) facilitator, which has made possible two key features of 'runjags' to be used inside 'BayLum': (i) MCMC-sampling parallelization and (ii) the ability to extend a model (drawing additional MCMC samples after a model has already run to completion). This paper will highlight these two new features of 'BayLum' and show examples of how to use them.

2. Problem: Stationary distributions require long run times

The Bayesian models produced with 'BayLum' infer parameter estimates (such as equivalent dose and age) from marginal posterior distributions of these parameters. This is to say that 'BayLum' takes the output of the Bayesian approach, a posterior distribution, and evaluates the dimensions of individual variables. 'BayLum' constructs these distributions via Markov Chain Monte Carlo sampling. The result of the MCMC sampling is a chain of values, each link consisting of a combination of values from all parameters in the Bayesian model. A distribution can then be constructed for each parameter, given its value in each link. To let the MCMC converge on the solution, we skip a number of the first iterations (burn-in phase) and only then begin constructing the distributions. To be confident in the results, the distributions must be stationary – that is, the location and shape of each distribution must not change if we draw additional samples. 'BayLum' assesses if distributions are stationary and independent of initialization of the MCMC by construct-

ing multiple chains instead of one. If the distributions from each chain agree with each other, we can be confident that the chains have converged to a single solution. By default, 'BayLum' uses three MCMC chains – a suitable balance between the power to detect non-convergence and the computational resources required (the number of chains is fully customizable by the user). 'BayLum' formalizes the question of convergence by incorporating as output the Rubin and Gelman diagnostic (Gelman & Rubin, 1992), which compares within-chain and between-chain variance. A common rule of thumb is that the upper 95 % credible interval limit of this diagnostic value indicates convergence when below 1.05.
For many practical applications of OSL dating, the number of iterations (or links in each chain) required to reach convergence is high (>500 000) – and higher still when 'BayLum' models incorporate many OSL samples as is the case with high-resolution chronologies. Because MCMC chains are to be processed consecutively, the overall process can become very time-consuming. For example, using a computer equipped with a 11^{th} Gen i7-1185G7 clocking at 3.0 GHz (which has a relatively high single-core threading performance rating), runtimes can extend beyond several days. Furthermore, even when a model completes, not all of the model's parameters may have converged – a result which could require a complete re-run of the 'BayLum' modelling function.

3. 'BayLum' feature: MCMC parallelization

Previous versions of 'BayLum' could only process MCMC chains consecutively using a single processor core. With parallelization, it is now possible to assign n chains out onto n CPU processor cores. This allows each chain to be processed concurrently, and the runtime will (ideally) approach 1/n when compared to the time for running n chains using a single core. We tested this using 'BayLum' models where OSL example sets GDB3 and GDB5 were used (both included with the 'BayLum' package) to produce 2-sample models. Figure 1A shows that when running 4 000 total iterations per chain, we see a significant runtime reduction when running the model using parallelization (`jags_method = "rjparallel"`) as compared to using only a single CPU core (`jags_method = "rjags"`). Reduction increases with the number of MCMC chains constructed in the model, which is what we expect. We observed a reduction of 65 % for a 3-chain setup and 72 % for a 4-chain setup. The minor differences we see from the theoretical 1/n-rule most likely arise from runtime inside the 'BayLum' model functions, which is not due to the iteration of MCMC sampling. We also see from Figure 1B that this reduction is consistent with increasing numbers of iterations. Example 1 (Sec. 3.1) shows how to apply parallelization in 'BayLum' v0.3.1. Note that our model testing was carried out using the High-Performance Computing Cluster "Sophia" (Technical University of Denmark, 2019). The same code run on a desktop PC will show the same relative reductions but may show poorer runtimes, not only because of lower overall computation power but also - and more likely - due to advanced power throttling measures of modern CPU architectures implemented to prevent overheating in prolonged high-load situations.

3.1. Example 1

In the example below, which we kept as simple and user-friendly as possible, we show how to achieve parallelization. The key argument to set is `jags_method = "rjparallel"`. We use the example data included within 'BayLum' at installation.

Example 1: R Code: Achieving parallelization

```r
# MCMC parallelization example ####
# load libraries
library(BayLum)

# load example DataFiles GDB3 and GDB5
data(DATA1)
data(DATA2)

# combine DataFiles
# (we now have a 2-sample DataFile)
DF <- combine_DataFiles(DATA1, DATA2)

# construct BayLum model
BayLum_model <- AgeS_Computation(
  DATA = DF,
  SampleNames = c("GDB3", "GDB5"),
  Nb_sample = 2,
  BinPerSample = c(1, 1),
  LIN_fit = FALSE,
  Origin_fit = TRUE,
  Iter = 1e+03,
  burnin = 5e+02,
  adapt = 5e+02,
  n.chains = 3,
  jags_method = "rjparallel"
)
```

4. 'BayLum' feature: extend the 'BayLum' model

Unfortunately, 'BayLum' model chains will not always converge within the specified number of iterations. In previous versions of 'BayLum', the 'BayLum'-model would likely need to run again with a higher number of iterations. The added runtime of re-running 'BayLum' can now be avoided by extending the non-converged model instead of building it again from scratch. In this case, all non-converged model iterations are treated as burn-in. See Example 2 (Sec. 4.1) for an illustration of how to extend 'BayLum' models.

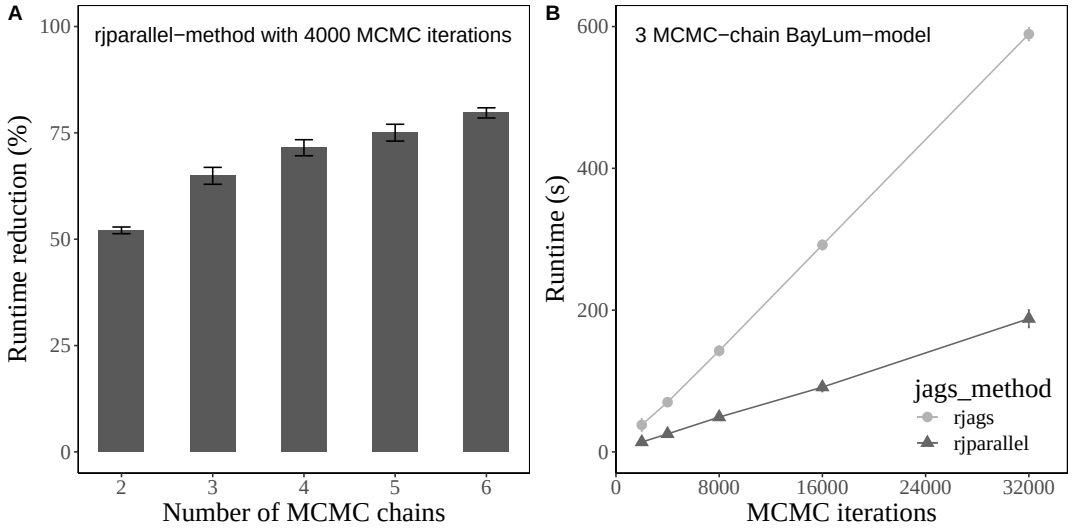

Figure 1: (A): Runtime reduction in percentage when running a 'BayLum' model with fixed iterations vs a varying number of MCMC chains using GDB3 and GDB5 example sets included within 'BayLum'. (B): Runtime in seconds vs the number of MCMC iterations for a 3-chain 'BayLum' model also using GDB3 and GDB5. All estimates show mean ± sd (n=8). To run the model, we used the High-Performance Computing cluster named "Sophia" owned by DTU. Arguments `"rjags"` and `"rjparallel"` entail whether 'BayLum' is run using a single CPU core ('rjags') or run in parallel on several cores (`'rjparallel'`).

4.1. Example 2

In Example 1 (Sec. 3.1), a model was built to show how parallelization could be achieved. The Rubin and Gelman convergence diagnostics from that model reveal evidence that not all MCMC chains converged (see `"D (Dose)"` for GDB5, Table 1).

Table 1: Rubin and Gelman convergence diagnostics for three parameters of the 'BayLum'-model in Example 1. We show only the upper 95 % credible interval limit.

Sample	A (Age)	D (Dose)	sD (Stand. deviation)
GDB3	1.006	1.022	1.004
GDB5	1.007	1.065	1.000

However, we can now add iterations to the 'BayLum'-model in order to achieve convergence:

Example 2: R Code: Extending model

```
# extend MCMC sampling of BayLum-model
BayLum_model_extended <- AgeS_Computation(
  DATA = BayLum_model,
  SampleNames = c("GDB3", "GDB5"),
  Nb_sample = 2,
  BinPerSample = c(1, 1),
  LIN_fit = FALSE,
  Origin_fit = TRUE,
  Iter = 1e+04,
  burnin = 0,
  adapt = 5e02,
  jags_method = "rjparallel"
)
```

Rubin and Gelman's convergence diagnostics now show we can be confident about all the parameters (Table 2).

Table 2: Rubin and Gelman convergence diagnostics for three parameters of the 'BayLum' model from example 1 (Sec. 3.1). We show only the upper 95 % credible interval limit.

Sample	A (Age)	D (Dose)	sD (Stand. deviation)
GDB3	1.002	1.007	1.000
GDB5	1.001	1.010	1.004

5. Conclusions

In this report, we introduced two feature updates to the R-package 'BayLum'. Together, they allow users to parallelize MCMC sampling and extend BayLum-models - both features significantly reduce the time needed to build a viable 'BayLum'-model.

Acknowledgments

We thank Geoff Duller for his thorough and supportive comments. We also gratefully acknowledge the computational and data resources the Sophia HPC Cluster provided at the Technical University of Denmark, DOI: https://doi.org/10.57940/FAFC-6M81.

References

Christophe, C., Philippe, A., Kreutzer, S., Guérin, G., and Baumgarten, F. *BayLum: Chronological Bayesian Models Integrating Optically Stimulated Luminescence and Radiocarbon Age Dating. https://cran.r-project.org/package=BayLum*, 2023. URL https://cran.r-project.org/package=BayLum. R package version 0.3.1.

Combès, B. and Philippe, A. *Bayesian analysis of individual and systematic multiplicative errors for estimating ages with stratigraphic constraints in optically stimulated luminescence dating*. Quaternary Geochronology, 39: 24–34, 2017. ISSN 1871-1014. doi: 10.1016/j.quageo.2017.02.003. URL https://www.sciencedirect.com/science/article/pii/S1871101416300838.

Combès, B., Philippe, A., Lanos, P., Mercier, N., Tribolo, C., Guérin, G., Guibert, P., and Lahaye, C. *A Bayesian central equivalent dose model for optically stimulated luminescence dating*. Quaternary Geochronology, 28: 62–70, 2015. doi: 10.1016/j.quageo.2015.04.001.

Denwood, M. J. *runjags: An R Package Providing Interface Utilities, Model Templates, Parallel Computing Methods and Additional Distributions for MCMC Models in JAGS*. Journal of Statistical Software, 71: 1–25, 2016. doi: 10.18637/jss.v071.i09.

Gelman, A. and Rubin, D. B. *Inference from Iterative Simulation Using Multiple Sequences*. Statistical Science, 7: 457–472, 1992. ISSN 08834237. URL http://www.jstor.org/stable/2246093.

Heydari, M., Guérin, G., Kreutzer, S., Jamet, G., Kharazian, M. A., Hashemi, M., Nasab, H. V., and Berillon, G. *Do Bayesian methods lead to more precise chronologies? 'BayLum' and a first OSL-based chronology for the Palaeolithic open-air site of Mirak (Iran)*. Quaternary Geochronology, 59: 101082, 2020. ISSN 1871-1014. doi: 10.1016/j.quageo.2020.101082. URL https://www.sciencedirect.com/science/article/pii/S1871101420300315.

Philippe, A., Guérin, G., and Kreutzer, S. *BayLum - An R package for Bayesian analysis of OSL ages: An introduction*. Quaternary Geochronology, 49: 16–24, 2019. doi: 10.1016/j.quageo.2018.05.009.

Plummer, M. *JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling*. Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, pp. 1–10, 2003.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/.

Technical University of Denmark. *Sophia HPC Cluster*. Research Computing at DTU, 2019. doi: 10.57940/fafc-6m81.

Reviewer

Geoff Duller