Bioinformatics, 31(21), 2015, 3549–3551
doi: 10.1093/bioinformatics/btv393
Advance Access Publication Date: 2 July 2015
Applications Note

Genetics and population analysis

al3c: high-performance software for parameter inference using Approximate Bayesian Computation

Alexander H. Stram[1,]*, Paul Marjoram[2] and Gary K. Chen[2]

[1]Cancer Center - Research, USC, Los Angeles, CA 90089, USA and [2]Division of Biostatistics, Department of Preventive Medicine, USC, Los Angeles, CA 90033, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on February 6, 2015; revised on May 21, 2015; accepted on June 24, 2015

Abstract

Motivation: The development of Approximate Bayesian Computation (ABC) algorithms for parameter inference which are both computationally efficient and scalable in parallel computing environments is an important area of research. Monte Carlo rejection sampling, a fundamental component of ABC algorithms, is trivial to distribute over multiple processors but is inherently inefficient. While development of algorithms such as ABC Sequential Monte Carlo (ABC-SMC) help address the inherent inefficiencies of rejection sampling, such approaches are not as easily scaled on multiple processors. As a result, current Bayesian inference software offerings that use ABC-SMC lack the ability to scale in parallel computing environments.
Results: We present *al3c*, a C++ framework for implementing ABC-SMC in parallel. By requiring only that users define essential functions such as the simulation model and prior distribution function, *al3c* abstracts the user from both the complexities of parallel programming and the details of the ABC-SMC algorithm. By using the *al3c* framework, the user is able to scale the ABC-SMC algorithm in parallel computing environments for his or her specific application, with minimal programming overhead.
Availability and implementation: *al3c* is offered as a static binary for Linux and OS-X computing environments. The user completes an XML configuration file and C++ plug-in template for the specific application, which are used by *al3c* to obtain the desired results. Users can download the static binaries, source code, reference documentation and examples (including those in this article) by visiting https://github.com/ahstram/al3c.
Contact: astram@usc.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

When modeling biological phenomena, the presence of interacting factors can make the identification of the model's parameter distribution intractably complicated, creating an incentive to simplify the model. Rather than risk oversimplifying such a model, a numerical estimate of parameter distributions can be obtained using Monte Carlo rejection sampling methods. By repeatedly simulating data with randomly chosen parameters and considering the distribution of only those parameters responsible for simulations most similar to the observed data, one can make inferences about the model's parameter distribution given the observed data. When the proposed parameters for rejection sampling are generated from a prior distribution, the process can be classified as an Approximate Bayesian Computation (ABC) algorithm, as described in Beaumont (2010). Due to their reliance on rejection sampling, ABC algorithms are inherently inefficient.

While basic ABC algorithms can trivially be accelerated by parallelizing simulations, more efficient methods of ABC are harder to parallelize. ABC Sequential Monte Carlo (ABC-SMC) improves the efficiency of ABC with an iterative method of rejection sampling, ultimately requiring less simulations and therefore less computation time (Sisson *et al.*, 2009). Due to its sequential nature, ABC-SMC is less trivial to parallelize and as a result, existing software implementations by Csilléry *et al.* (2012), Jabot *et al.* (2013) and Wegmann *et al.* (2010) cannot effectively use large computing clusters in parallel. Software by Liepe *et al.* (2014) allows for the massively parallel execution of code on graphics processing units (GPUs), although this requires in-depth parallel programming knowledge. Researchers are therefore unable to scale an efficient method of ABC on multiprocessor computers or computing clusters without some parallel programming knowledge. To address this, we present *al3c*, an object oriented framework for implementing ABC-SMC in parallel without knowledge of parallel programming.

2 Software

2.1 ABC-SMC in Parallel Algorithm

al3c implements a Sequential Importance Sampling derived ABC-SMC algorithm (Toni, 2009) for parameter inference using user-defined functions which characterize the simulation model and the parameters to infer. If D is the observed data of interest and θ the parameter(s) used to simulate data, we use ABC-SMC to estimate $P(\theta|D)$, the posterior distribution of θ given D. Estimates are found by sampling θ from a prior distribution $\pi(\theta)$, then simulating data with each sampled θ and accepting values of θ that simulated data most similar to D. In subsequent generations we perform weighted sampling from the set of θs that were accepted in the previous generation, perturb each sampled parameter slightly to yield θ^*, and simulate data with θ^*. After a specified number of perturbed θ^*s have been accepted, we weight each and repeat the process iteratively.

al3c's parallel implementation of ABC-SMC uses a 'single program multiple data' parallelization strategy via Message Passing Interface (MPI) libraries to decrease the computation time of the ABC-SMC algorithm. Multiple processors concurrently sample, perturb and evaluate parameters for goodness of fit independently of each other, using a common rejection threshold, until each processor has found its allocated number of acceptances. When each processor has completed its assignment, parameters are collected from each processor, weighted and re-distributed to all processors for the next generation of ABC-SMC. Since the simulation component of rejection sampling constitutes the vast majority of computation time in ABC-SMC, we expect our algorithm will scale linearly with respect to the number of processors used, despite the time spent synchronizing processors to accommodate for the sequential nature of ABC-SMC.

3 Examples

3.1 Estimation of demographic model parameters

To demonstrate the scalability of *al3c*, we configured the software to incorporate *MaCS*, a program designed to simulate large genetic sequences based on arbitrary demographic models (Chen *et al.*, 2009). Following the work of Gravel (2011), we defined a demographic model with seven parameters and compared summary statistics between simulations and 1000 Genomes data to test for goodness of fit (McVean, 2012), as detailed in the Supplementary Information.

For benchmark purposes, we first ran *al3c* on one processor, and found that after approximately 24 h its worst accepted simulation was of Euclidean distance 66.0 from our observed dataset. We repeated this analysis with *al3c* configured to run on 2–128 processors and quit once each of the 1000 accepted simulations was of distance less than or equal to 66.0 from our observed dataset, and recorded the respective run time. Each compute node was equipped with two Intel Xeon 2.33 GHZ 4-core processors and 12 GB of memory.

3.1.1 al3c scales linearly with respect to processors available

Each time the number of processors available to *al3c* was doubled, the run time was reduced by approximately one-half, while each run saw a similar convergence in distribution for each of the seven examined parameters (Table 1).

3.2 Estimation of behavioral patterns in Drosophila fruit-fly

abctoolbox, a prominent ABC-SMC software offering requires users to call an external simulation program with system calls (Wegmann *et al.*, 2010). In contrast, *al3c* supports both system and native function calls to existing C/C++ code. The use of native functions as opposed to system calls can significantly reduce the amount of overhead involved in initializing and running each simulation, particularly in cases where running a single simulation requires a relatively large amount of initialization steps, but the marginal cost of further simulations is minimal. To assess the advantage of using native calls, we configured *al3c* to simulate data using *flysim*, a Markov chain Monte Carlo simulator of Drosophila fly behavior (Foley *et al.*, 2015) using both configurations.

3.2.1 al3c computation times can be further improved through the use of shared libraries

Running *al3c* by natively invoking the simulation via a function call to a shared library requires only one initialization per processor whereas calling *flysim* via system calls requires a fresh initialization for each simulation. We found that when configured with 128 processors and an acceptance distance criterion of 2.5, *al3c* completed

Table 1. Total runtime and parameter MLEs for the *MaCS* example

Processors	1	2	4	8	16	32	64	128
Time (HH:MM)	24:55	12:51	6:40	3:18	2:02	0:55	0:36	0:23
N_{AFR}	1.51	1.50	1.50	1.52	1.50	1.50	1.51	1.51
$m_{AFR->EUR}$	2.90	2.88	2.85	2.88	2.90	2.98	2.95	2.98
$m_{EUR->ASN}$	3.15	3.15	3.15	3.20	3.23	3.25	3.23	3.23
$m_{AFR->ASN}$	0.88	0.88	0.88	0.88	0.90	0.90	0.90	0.90
r_{EUR}	0.49	0.48	0.49	0.48	0.48	0.50	0.50	0.50
r_{ASN}	0.58	0.59	0.58	0.59	0.58	0.59	0.60	0.59
$P_{AFR-EUR}$	17.1	17.1	17.2	17.1	17.3	17.6	17.6	17.6

N_{AFR} is effective African population size, $m_{AFR->EUR}$, $m_{EUR->ASN}$, $m_{AFR->ASN}$ are migration rates from Africa to Europe, Europe to Asia, Africa to Asia, respectively, r_{EUR} and r_{ASN} are growth rates in Europe in Asia and $P_{AFR-EUR}$ is the time since Africa to Europe migration event.

in 16 h using system calls, versus 319 s when using native calls—a 180-fold speedup. (A more stringent criterion exceeded our cluster's time allotment).

4 Discussion

We have given two examples in which parameter inference of biological models was accelerated with the *al3c* framework. The literature on methods and programs for ABC inference is broad, and there are problems where parallel implementations are optimized to perform well for a narrower class of problems and may be a more appropriate choice, such as in systems biology (Liepe *et al.*, 2014). *al3c*, however, provides the research community with the first available general-purpose ABC application that allows users to employ parallelized ABC-SMC in any context without knowledge of parallel programming. Future features for al3c may include automatic optimizing of rejection threshold schedules to prevent parameter estimates from converging to local maxima (Silk, 2013), and GPU support to further parallelize individual simulations.

Funding

The authors gratefully acknowledge funding support from NIH (MH100879), U01 (GM10380) and NSF (DMS 1101060).

Conflict of Interest: none declared.

References

Beaumont,M.A. (2010) Approximate Bayesian computation in evolution and ecology. *Annu. Rev. Ecol. Evol. Syst.*, **41**, 379–406.

Chen,G.K. *et al.* (2009) Fast and flexible simulation of DNA sequence data. *Genome Res.*, **19**, 136–142.

Csilléry,K. *et al.* (2012) abc: an R package for approximate Bayesian computation (ABC). *Methods Ecol. Evol.*, **3**, 475–479.

Foley,B. *et al.* (2015) A novel Bayesian approach to social structure uncovers cryptic regulation of group dynamics. *Am. Nat.*, **185**, 797–808.

Gravel,M. (2011) Demographic history and rare allele sharing among human populations. *Proc. Natl Acad. Sci.*, **108**, 11983–11988.

Jabot,F. *et al.* (2013) EasyABC: performing efficient approximate Bayesian computation sampling schemes using r. *Methods Ecol. Evol.*, **4**, 684–687.

Liepe,J. *et al.* (2014) A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation. *Nat. Protocols*, **9**, 439–456.

McVean,G.A. (2012) An integrated map of genetic variation from 1 092 human genomes. *Nature*, **491**, 56–65.

Silk,D. *et al.* (2013) Optimizing threshold-schedules for sequential approximate Bayesian computation: applications to molecular systems. *Stat. Appl. Genet. Mol. Biol.*, **12**, 603–618.

Sisson,S. *et al.* (2009) Correction for sisson et al., sequential Monte Carlo without likelihoods. *Proc. Natl Acad. Sci.*, **106**, 16889.

Toni,T. *et al.* (2009) Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *J. R. Soc. Interface*, **6**, 187–202.

Wegmann,D. *et al.* (2010) ABCtoolbox: a versatile toolkit for approximate Bayesian computations. *BMC Bioinformatics*, **11**, 116+.