Pakman: a modular, efficient and portable tool for approximate Bayesian inference

Thomas F. Pak[1], **Ruth E. Baker**[1], **and Joe M. Pitt-Francis**[2]

1 Mathematical Institute, University of Oxford **2** Department of Computer Science, University of Oxford

DOI: 10.21105/joss.01716

Software

- Review
- Repository
- Archive

Editor: Jed Brown

Reviewers:

- @jmlarson1
- @gonsie

Submitted: 01 August 2019
Published: 05 March 2020

Summary

The development of high-throughput techniques in the biological sciences has resulted in an abundance of experimental data. At the same time, mathematical models are becoming increasingly popular in the biological sciences. Combined, these parallel advances have the potential to greatly expand our understanding of biological processes through mathematical modelling and data-driven parameter inference. When the data are noisy or the mathematical model is inherently stochastic, we can apply Bayesian methods for parameter inference and model selection. Moreover, even when the likelihood function is unknown or intractable, it is still possible to make progress by applying a method known as **approximate Bayesian computation (ABC)** (Marjoram, Molitor, Plagnol, & Tavare, 2003; Toni, Welch, Strelkowa, Ipsen, & Stumpf, 2009).

The drawback of ABC methods is that they require many model simulations, which quickly becomes a bottleneck when simulations are computationally expensive. Fortunately, some ABC algorithms have a simulation workload that is embarrassingly parallel, or proceed through a sequence of embarrassingly parallel simulation workloads. These algorithms, ABC rejection and ABC sequential Monte Carlo (SMC) specifically, can be parallelised to dramatically reduce computation times (Toni et al., 2009). Therefore, parallel ABC methods are a natural choice for leveraging "big data" and "big compute" for parameter inference.

There are a number of software tools for ABC already available. However, some were designed for particular classes of domain-specific models. For instance, `DIYABC` (Cornuet et al., 2014) is a comprehensive software package for inference in population genetics, and includes a graphical user interface. Another example is `ABC-SysBio` (Liepe et al., 2010), a Python package for inferring the parameters of dynamical models of biochemical systems based on ordinary or stochastic differential equations. Other software tools are modular and can be used with any type of model, but do not scale well with computational resources. For instance, the R package `abc` (Csilléry, François, & Blum, 2012) can handle arbitrary models, but does not parallelise simulations. `ABCtoolbox` (Wegmann, Leuenberger, Neuenschwander, & Excoffier, 2010) is a command-line tool for running ABC algorithms on models that are supplied as command-line programs, but it does not support explicit parallelisation. For an overview of general-purpose ABC software, see Kousathanas, Duchen, & Wegmann (2018).

More recently, parallel ABC has been implemented successfully to accelerate inference in a computationally intensive problem (Jagiella, Rickert, Theis, & Hasenauer, 2017). However, the implementation was problem- and architecture-specific and thus could not be applied more generally. In response, the Python package `pyABC` (Klinger, Rickert, & Hasenauer, 2018) was developed as a more flexible implementation of parallel ABC, supporting shared- and distributed-memory computing. In principle, `pyABC` can be used with any type of model, but its computational implementation must be expressed as a Python function. This poses a

problem when the user develops their model in a programming language other than Python or is using an external software package that cannot easily be adapted to Python.

Pakman is a parallel ABC manager that is designed to be modular at the systems-level, as opposed to the application-level. Furthermore, Pakman is also designed to be portable and efficient. Pakman is written in C++11 (The C++ Standards Committee, 2011), and relies on the Message Passing Interface (MPI) library, standard MPI-3.1 (Message Passing Interface Forum, 2015), for parallelisation. We chose C++11 because of its native support for MPI and POSIX system calls, high-level programming language features, and efficiency. Moreover, we chose MPI as the platform for parallelisation because it is a well-established standard for distributed computing that has been implemented on a wide variety of systems, ranging from multi-core machines to large computational clusters.

In summary, Pakman was made for performing likelihood-free inference when model simulations are computationally expensive. The lack of an analytical likelihood requires the application of ABC methods, and the computational cost of individual simulations merits a parallel approach to decrease the time to solution. Moreover, in order to be as modular as possible, models are specified as black-box programs. The target audience consists of researchers who want to parameterise a computationally demanding stochastic model based on experimental data.

Modularity

Pakman is designed around a modular framework where problem-specific tasks are performed by **user executables**. These are executables (binaries or scripts) that are supplied by the user, which are then invoked by Pakman using the system calls `fork()` and `exec()`. Furthermore, Pakman communicates with user executables through their standard input and standard output.

Conceptually, user executables serve the same role as *functions* in programming languages; they are called with some input, perform a computational task and return an output. The difference is that they are treated as black boxes that are not constrained to any programming language. The user can therefore use any language or software package they desire to implement user executables. As long as the model simulation can be executed in the shell, or wrapped in a shell script, it can be used with Pakman.

Efficiency

The parallel architecture of Pakman is based on the Master–Worker model, where a Master process dispatches jobs to a pool of Worker processes, ensuring that Workers are busy at all times. In our case, each job is a single simulation and a Worker is an instance of the `simulator` user executable. The choice of MPI for parallelisation prevents us from directly applying the Master–Worker model however; a Worker cannot simultaneously be an MPI process and an instance of a user executable. If the Worker were an MPI process, the simulation would have to be compiled into Pakman, breaking the modular framework.

Our solution is to implement a layer of "Managers" between the Master and Workers, resulting in a **Master–Manager–Worker** architecture. These Managers are parallel MPI processes that communicate with the Master through MPI and fork Workers as user executables to perform the actual work. Hence, the Master and Manager processes collectively form a parallel infrastructure from which processes are spawned to perform simulations.

Further efficiency improvements are achieved by implementing the Master and Managers with event-based loops. This means that the Master and Manager processes are asleep most of the time and only take action when an event requires it (e.g. a Worker has finished its simulation and wants to report its results), leaving the CPU to focus on running simulations.

Portability

As highlighted earlier, Pakman is written in C++11 and uses MPI for parallelisation. Moreover, we use CMake, a cross-platform tool, to manage the build process. As a result, Pakman can be built and used on wide variety of systems with few dependencies, making it a very portable tool. Travis CI is used to continuously test Pakman on Linux (Ubuntu Xenial distribution), as well as macOS (version 10.13). Details on the Travis CI configuration can be found in the code repository.

Illustrative examples

Pakman comes with three examples that demonstrate its capabilities. In the first example, a biased coin with an unknown probability to land on heads, q, is flipped 20 times and lands on heads five times. This example is simple enough that the likelihood for the parameter of interest (q) and thus the posterior distribution, can be written down analytically, as the number of heads obeys a binomial distribution. Therefore, numerical results can be compared with the analytical solution to verify correctness of the ABC methods. In Figure 1, we show the results of applying the ABC rejection method and the ABC SMC method.

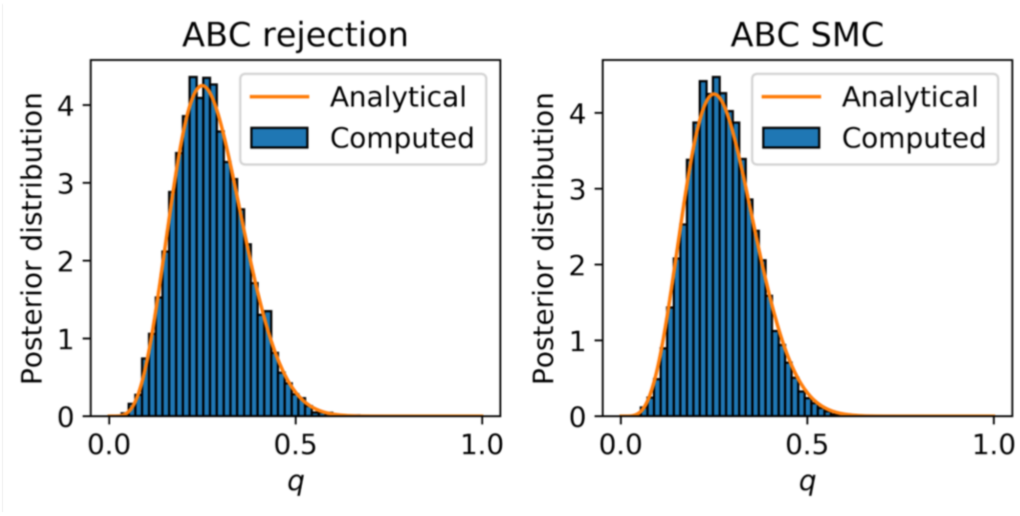

Figure 1: The computed posterior distributions for q obtained using the ABC rejection method (left) and the ABC SMC method (right). We chose a standard uniform distribution as the prior distribution. For the ABC rejection method, we used a tolerance of zero and obtained 10,000 accepted samples. For the ABC SMC method, we used a normal distribution with a standard deviation of 0.3 as the perturbation kernel. The tolerance series was $\langle 2, 1, 0 \rangle$ and the population size was 10,000. For both methods, the computed distributions closely fit the analytical solution, validating our implementation.

The second example involves a dynamic model for the spread of an infectious disease that confers no immunity to the host after recovery. Representing susceptible individuals by S and infected individuals by I, the SIS model we consider is given by

$$S + I \xrightarrow{\beta} 2I,$$

$$I \xrightarrow{\gamma} S.$$

The first reaction represents infection of a susceptible individual by an infected individual and the second reaction describes the recovery of an infected individual. We observe the number of susceptible and infected individuals at a series of fixed time points and our goal is to infer the parameters β and γ. The exact likelihood is feasible to compute when the total number of individuals is relatively small, so here too we can compare numerical and analytical results. The results of running the ABC SMC method are given in Figure 2.

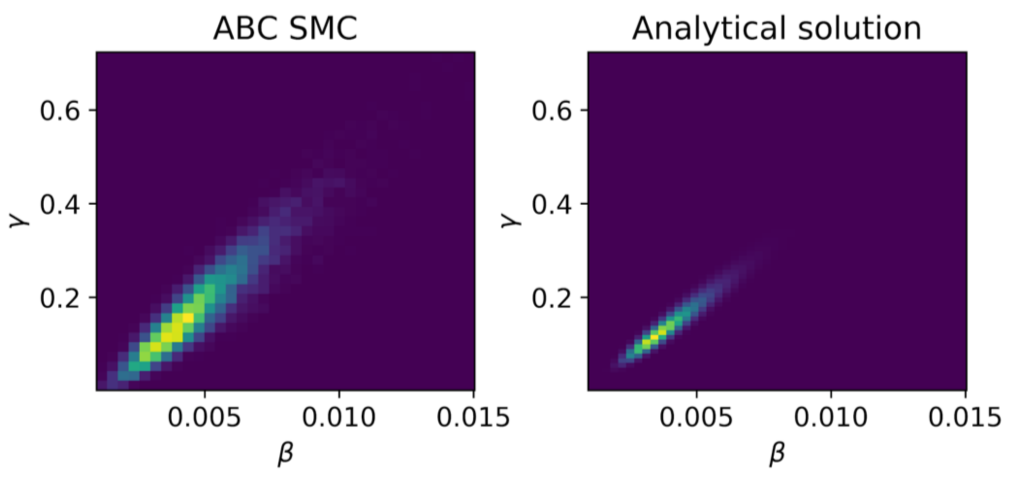

Figure 2: The computed (left) and analytical (right) bivariate posterior distribution for β and γ. We obtained the computed posterior using the ABC SMC method. We used a uniform prior for both parameters and used normally distributed perturbation kernels to perturb them independently, with standard deviations 0.006 and 0.2 for β and γ, respectively. The tolerance series was $\langle 75, 70, \ldots, 35\rangle$ and the population size was 10,000. We obtained the analytical posterior by numerically computing the transition probabilities of the corresponding discrete-state continuous-time Markov chain. The computed and analytical posteriors do not correspond exactly because we used a nonzero tolerance. See the GitHub wiki for more details.

In the final example, we use a cell-based model of epithelial tissue growth to infer the average cell cycle time, t_{cycle}, from the number of cells that we observe after letting an initial population of four cells grow and divide for a fixed period of time. The model was implemented in Chaste, a software library for computational physiology and biology (Mirams et al., 2013) that includes a range of cell-based models (Osborne, Fletcher, Pitt-Francis, Maini, & Gavaghan, 2017). This example demonstrates that Pakman is flexible enough to run simulations implemented using established scientific software packages and libraries. We used Pakman to run the ABC SMC method on this example and obtained the posterior histogram shown in Figure 3.

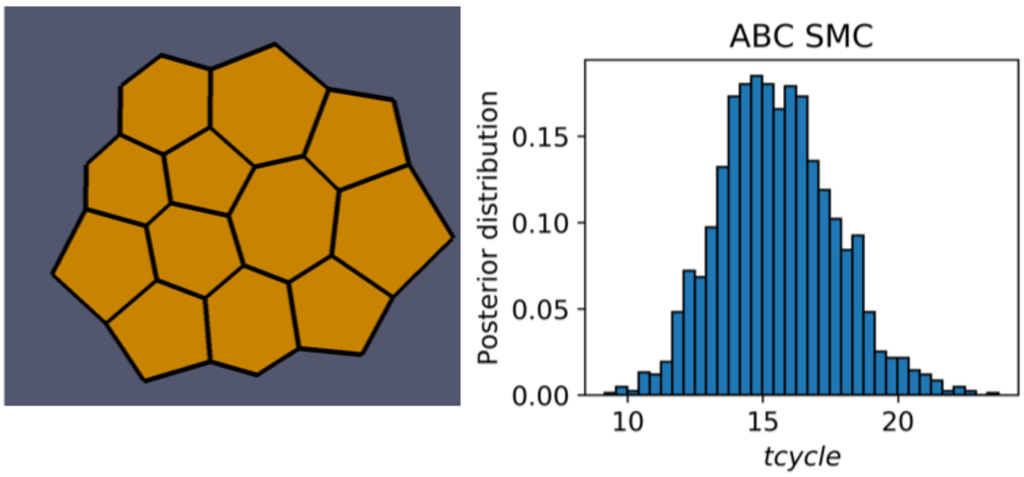

Figure 3: Left: an example simulation of the cell-based model. Right: the approximate posterior distribution for t_{cycle}, computed using the ABC SMC method. We used a uniform prior and a normally distributed perturbation kernel with a standard deviation of 1. The tolerance series was $\langle 4, 3, 2, 1, 0 \rangle$ and the population size was 2,000. See the GitHub wiki for more details.

Future development

At the current time, the ABC rejection method and the ABC SMC method have been implemented in Pakman. These are the most widely used ABC algorithms, as well as the easiest to parallelise. Nevertheless, the software architecture of Pakman was designed with object-oriented programming to facilitate extending Pakman to include other ABC algorithms. These include variations of ABC SMC that employ adaptive proposal kernels to improve acceptance rates, such as ABC population Monte Carlo (Beaumont, Cornuet, Marin, & Robert, 2009), or further generalisations thereof (Filippi, Barnes, Cornebise, & Stumpf, 2013).

There are also ABC algorithms that are not directly related to ABC SMC, such as ABC Markov Chain Monte Carlo (MCMC) (Marjoram et al., 2003), and ABC parallel tempering (Baragatti, Grimaud, & Pommeret, 2013). The latter method is inspired by a Monte Carlo method for simulating physical systems. We aim to implement these advanced ABC algorithms in the near future. Furthermore, we plan to update Pakman with new parallel ABC algorithms as they are introduced to the field of ABC.

The design of Pakman also lends itself well to systematically exploring the parameter space of models in a non-Bayesian context. In this case, Pakman functions as an engine to parallelise simulation workloads where the model needs to be evaluated for a large number of parameter sets and/or initial conditions. We have implemented a parameter sweep method that leverages Pakman's modular framework and parallel architecture to achieve this. Using Pakman to parallelise parameter sweeps has the potential to greatly accelerate the parameter sensitivity analysis of models where a large range of parameter sets needs to be considered, such as models of cardiac electrophysiology (Britton et al., 2013; Lawson et al., 2018).

Acknowledgements

Thomas Pak was supported by funding from the Biotechnology and Biological Sciences Research Council (BBSRC) via grant number BB/M011224/1.

Ruth E. Baker is a Royal Society Wolfson Research Merit Award holder and a Leverhulme Research Fellow, and also acknowledges the BBSRC for funding via grant number BB/R000816/1.

References

Baragatti, M., Grimaud, A., & Pommeret, D. (2013). Likelihood-free parallel tempering. *Statistics and Computing*, *23*(4), 535–549. doi:10.1007/s11222-012-9328-6

Beaumont, M. A., Cornuet, J. M., Marin, J. M., & Robert, C. P. (2009). Adaptive approximate Bayesian computation. *Biometrika*, *96*(4), 983–990. doi:10.1093/biomet/asp052

Britton, O. J., Bueno-Orovio, A., Van Ammel, K., Lu, H. R., Towart, R., Gallacher, D. J., & Rodriguez, B. (2013). Experimentally calibrated population of models predicts and explains intersubject variability in cardiac cellular electrophysiology. *Proceedings of the National Academy of Sciences*, *110*(23), E2098–E2105. doi:10.1073/pnas.1304382110

Cornuet, J. M., Pudlo, P., Veyssier, J., Dehne-Garcia, A., Gautier, M., Leblois, R., Marin, J. M., et al. (2014). DIYABC v2.0: a software to make approximate Bayesian computation inferences about population history using single nucleotide polymorphism, DNA sequence and microsatellite data. *Bioinformatics*, *30*(8), 1187–1189. doi:10.1093/bioinformatics/btt763

Csilléry, K., François, O., & Blum, M. G. B. (2012). abc: an R package for approximate Bayesian computation (ABC). *Methods in Ecology and Evolution*, *3*(3), 475–479. doi:10.1111/j.2041-210X.2011.00179.x

Filippi, S., Barnes, C. P., Cornebise, J., & Stumpf, M. P. H. (2013). On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. *Statistical Applications in Genetics and Molecular Biology*, *12*(1), 87–107. doi:10.1515/sagmb-2012-0069

Jagiella, N., Rickert, D., Theis, F. J., & Hasenauer, J. (2017). Parallelization and high-performance computing enables automated statistical inference of multi-scale models. *Cell Systems*, *4*(2), 194–206. doi:10.1016/j.cels.2016.12.002

Klinger, E., Rickert, D., & Hasenauer, J. (2018). pyABC: distributed, likelihood-free inference. *Bioinformatics*, *34*(20), 3591–3593. doi:10.1093/bioinformatics/bty361

Kousathanas, A., Duchen, P., & Wegmann, D. (2018). A guide to general-purpose approximate Bayesian computation software. In *Handbook of approximate Bayesian computation* (pp. 369–414). Chapman and Hall/CRC. Retrieved from http://arxiv.org/abs/1806.08320

Lawson, B. A. J., Drovandi, C. C., Cusimano, N., Burrage, P., Rodriguez, B., & Burrage, K. (2018). Unlocking data sets by calibrating populations of models to data density: A study in atrial electrophysiology. *Science Advances*, *4*(1). doi:10.1126/sciadv.1701676

Liepe, J., Barnes, C., Cule, E., Erguler, K., Kirk, P., Toni, T., & Stumpf, M. P. H. (2010). ABC-SysBio–approximate Bayesian computation in Python with GPU support. *Bioinformatics*, *26*(14), 1797–1799. doi:10.1093/bioinformatics/btq278

Marjoram, P., Molitor, J., Plagnol, V., & Tavare, S. (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, *100*(26), 15324–15328. doi:10.1073/pnas.0306899100

Message Passing Interface Forum. (2015). *MPI : A Message-Passing Interface Standard, Version 3.1*. Knoxville: University of Tennessee.

Mirams, G. R., Arthurs, C. J., Bernabeu, M. O., Bordas, R., Cooper, J., Corrias, A., Davit, Y., et al. (2013). Chaste: an open source C++ library for computational physiology and biology. *PLoS Computational Biology*, *9*(3), e1002970. doi:10.1371/journal.pcbi.1002970

Osborne, J. M., Fletcher, A. G., Pitt-Francis, J. M., Maini, P. K., & Gavaghan, D. J. (2017). Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLoS Computational Biology*, *13*(2), e1005387. doi:10.1371/journal.pcbi.1005387

The C++ Standards Committee. (2011). ISO/IEC 14882:2011, standard for programming language C++. Geneva: International Organization for Standardization. Retrieved from https://www.iso.org/standard/50372.html

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., & Stumpf, M. P. H. (2009). Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, *6*(31), 187–202. doi:10.1098/rsif.2008.0172

Wegmann, D., Leuenberger, C., Neuenschwander, S., & Excoffier, L. (2010). ABCtoolbox: a versatile toolkit for approximate Bayesian computations. *BMC Bioinformatics*, *11*. doi:10.1186/1471-2105-11-116