Bayesian Analysis Toolkit: 1.0 and beyond

Frederik Beaujean\textsuperscript{1}, Allen Caldwell\textsuperscript{2}, D. Greenwald\textsuperscript{3}, S. Kluth\textsuperscript{2}, Kevin Kröninger\textsuperscript{4}, O. Schulz\textsuperscript{2}

\textsuperscript{1} Ludwig-Maximilians-Universität München, Germany
\textsuperscript{2} Max-Planck-Institut für Physik, München, Germany
\textsuperscript{3} Technische Universität München, Germany
\textsuperscript{4} Technische Universität Dortmund, Germany
E-mail: Frederik.Beaujean@lmu.de

Abstract. The Bayesian Analysis Toolkit is a C++ package centered around Markov-chain Monte Carlo sampling. It is used in high-energy physics analyses by experimentalists and theorists alike. The software has matured over the last few years. We present new features to enter version 1.0, then summarize some of the software-engineering lessons learned and give an outlook on future versions.

1. Introduction
The main goals of a typical data analysis are to compare model predictions with data, to draw conclusions on the validity of the model as a representation of the data, and to extract the possible values of parameters within the context of a model. The Bayesian Analysis Toolkit \cite{BAT} (BAT) is a software package that helps users meet these goals. BAT is free software and can be obtained from \url{http://mpp.mpg.de/bat}. Its foundation is Bayes’ theorem which for a single model, $M$, has the form

$$P(\theta|D, M) = \frac{P(D|\theta, M)P_0(\theta|M)}{\int d\theta P(D|\theta, M)P_0(\theta|M)};$$

i.e., the probability of a set of parameters $\theta$ given data $D$, the posterior probability, is proportional to the probability of the data given the parameters, also known as the likelihood, times the prior probability for the parameters. The denominator on the right-hand side of the equation, the evidence, is the integral of the numerator over $\theta$ and ensures that the posterior probability is normalized.

Application of Bayes’ theorem generally requires integration, for example, to compute a 1D marginal distribution

$$P(\theta_i|D, M) = \int \prod_{j \neq i} d\theta_j P(\theta|D, M)$$

or the evidence

$$P(D|M) = \int d\theta P(D|\theta, M)P_0(\theta|M).$$

Quadrature methods are the best numerical algorithms for low-dimension problems but are inapplicable to high-dimension problems. The general strategy to overcome the curse of
dimensionality is to use random numbers; i.e., Monte Carlo methods. The goal of the Bayesian Analysis Toolkit is to supply the necessary algorithms and some generic models that are widely applicable to typical analyses in high-energy physics (HEP). This enables users to focus on their particular data analyses, and relieves them from reimplementing standard algorithms.

BAT is implemented in C++ as two libraries: a core library and a model library. It relies on ROOT [2] functionality for input, output, and drawing. The core library is a set of classes that provide a general infrastructure, algorithms, output, and logging. The model library bundles together extension classes to solve specific (fitting) problems. BAT’s functionality covers the mapping of the posterior probability in multidimensional parameter space and the extraction of quantities of interest such as estimated values of parameters, uncertainty intervals, correlations between different parameters, and parameter limits. It also facilitates goodness-of-fit testing and model comparison. At its heart, BAT uses a Metropolis algorithm [3] to create a Markov chain of posterior samples.

Bayesian inference is increasingly used in the HEP community as witnessed by the rising number of citations and downloads of BAT. At the time of writing, there were 112 citations of the original BAT publication [1] on INSPIRE HEP and 194 downloads of the latest version 0.9.4.1 released on Jan. 19, 2015. BAT was recently cited in HEP analyses by the experimental collaborations ATLAS [4], Belle [5], CMS [6], and GERDA [7]; and also by the theory collaboration UTFIT [8].

This note is an update to our presentations of BAT at past CHEP conferences in 2009 [9] and 2010 [10]. For introductory material on how to use BAT, refer to the documentation available online. In the rest of this paper, we present the lessons we learned in software engineering while developing and maintaining the code in Section 2. We present new features in current development in Section 3 and give an outlook on future developments in Section 4.

2. Lessons learned

Initial development on BAT began in 2007 and the first public release was in 2008. The early work was done by only two developers, one of whom left physics since. For version control, we used a centralized approach with a privately hosted subversion server.

Later on, a new generation of developers took over the project. For all developers, BAT can only consume a small fraction of our total time. We therefore migrated the code from a privately hosted subversion repository to github (https://github.com/bat/bat) to facilitate community involvement. We saw three main benefits from the migration.

2.1. Cleaner code with git

Git is a decentralized version control system so every clone of the repository contains the complete history. Compared to subversion, branching is much easier, so we actually use it to develop isolated features or bug fixes. Being able to easily share the current status of a branch with a coworker makes it much easier to provide code review before changes are merged into the master branch. In total, this leads to better code.

2.2. Better interaction through github

Moving from a private to a publicly visible repository made it much easier for users to add features by sending pull requests, triggering code reviews as explained above. Github includes a basic issue tracker that offers everything we have needed so far. It makes it easier for us to manage a “to do” list. Users only need github accounts to quickly report issues and bugs. The whole discussion is public, so it is easy to track, contribute to, and find via search engines like google.
2.3. **Safer code with unit tests**

BAT was initially developed without unit tests. There exists a comprehensive suite of tests to validate the MCMC sampling but it is only run once immediately before each new release. Most new features were documented with examples. However, from the examples it is not immediately clear if a feature still works as originally intended, especially if it was written by another developer.

This situation limited maintenance as it became harder to change something when we didn’t know if it would break functionality in an unexpected place. The solution was to add unit tests and to integrate them into the build process such that a successful run of `make check` after modifications to the core of BAT gives the comforting assurance that things are still working. We must emphasize here that we do not have 100% test coverage; but we add unit tests for modules that we change, and usually there is some interplay with existing code. In this way, we continuously increase the test coverage.

3. **New features**

The main goal of current developments is to prepare for a release of version 1.0 that will be maintained long term but probably not see too many new features. Therefore, we invest time to improve the current code and add a number of useful features.

We streamline option setting and strive for a more uniform user experience. We want to be mostly backwards compatible but change the API in the few places with the biggest inconsistencies. Backwards-incompatible changes will be avoided once version 1 is out. Other efforts go towards making the build procedure more robust on multiple platforms and supporting ROOT versions 5 and 6.

3.1. **Factorized priors**

Prior setting, an essential part of the Bayesian approach, should be both easy and come with lots of options in BAT. For many problems the prior probability can be factorized into one-dimensional probabilities:

\[
P_0(\theta|M) = \prod_i P_0(\theta_i|M).
\]  

(4)

These one-dimensional prior probabilities are represented in BAT version 1 by classes that inherit from a prior base class. BAT version 1 will contain prior classes for the Gaussian, split Gaussian, Cauchy, and uniform distributions; and classes to implement prior distributions based on ROOT’s function and histogram objects. The prior base class handles all necessary interaction between a prior and the core of BAT, allowing users to implement new prior probability distributions quickly, with thought only to the mathematics of the distribution. In this way, we invite users to submit their priors for inclusion in the BAT library, so that others may benefit from them.

3.2. **Observables**

Previous versions of BAT have given users access to the Markov-chain samples at every iteration, so that they may generate marginalizations of functions of the model parameters—so-called observables. This required users to create, fill, store, and output histograms for every observable, and access the multiple Markov chains on an intimate level. And more robust information, such as correlations with parameters, is also complicated to generate. However, BAT currently handles all such functionality for parameters. So BAT version 1 implements a system for the marginalization of such observables that is as easy and intuitive as the current system for parameters.
3.3. File output
BAT has always allowed output of Markov-chain samples to a ROOT file using ROOT’s TTree class. Version 1 will streamline the turning on of such output for the user, and add the much-needed feature of loading samples back into BAT. This is implemented such that users can share samples produced by BAT with each other without having to share the computer code used to generate them—that is, the model implementation. In this way, for example, a member of an experimental collaboration may share posterior samples with a theorist without giving him access to the experimental data used to produce the samples—which are usually not shared outside collaborations.

3.4. Multivariate proposal
Presently, the proposal in the Metropolis step updates one parameter at a time and the proposal for a single parameter is a Cauchy distribution centered on the current point of the Markov chain with width tuned to reach an acceptance rate in a user-defined range whose default is \([0.15, 0.5]\). A step must be proposed in each dimension of the model, and the likelihood evaluated for each proposal, to complete one iteration of the Metropolis-Hastings algorithm.

For version 1.0, we include an adaptive scheme [11] that is based on learning the full posterior covariance from past samples. Thus we can update all parameters at once with a single evaluation of the likelihood. Initial tests indicate that convergence as quantified by the \(R\) value [12] takes somewhat more all-parameter updates but comes at the price of a significantly reduced number of calls to the posterior. For most problems, especially tough problems in high dimensions in which a single call can take seconds, the new multivariate proposal provides orders-of-magnitude speed ups.

3.5. Evidence from MCMC
Bayesian model comparison requires evaluating the evidence [3]. In the context of BAT, we developed a new algorithm [13] to compute the evidence as a post-processing step from MCMC samples. We define a small volume \(V\) around the posterior mode with fairly large posterior probability,

\[
A \equiv \int_V P(\theta|D, M),
\]

and sample-mean integrate the unnormalized posterior in \(V\),

\[
B \equiv \int_V P(D|\theta, M)P_0(\theta|M).
\]

Assuming the MCMC samples are distributed according to the posterior, \(A\) can be estimated by the fraction of samples that end up in \(V\) and the evidence then follows as

\[
P(D|M) = \frac{B}{A} \approx \frac{B \times \text{total number of samples}}{\text{number of samples in } V}.
\]

Implementation of this algorithm as post-processing hinges on the file output described in Section 3.3.

4. Outlook
Our plan for the future is to make BAT ready for very demanding analysis problems in which the posterior evaluation requires seconds to minutes and memory consumption could exceed that available on a single compute node. An essential requirement is that support for parallel architectures comes at multiple levels; i.e., we want to provide sampling algorithms that are
We intrinsically parallel by evaluating the posterior at different parameter values but we also want to help the user with parallelizing a single posterior call. The most efficient way to solve a problem strongly depends on the problem of the user, which we cannot know in advance. But we intend to provide enough means to allow the user to easily find the best combination of parallel algorithms and posterior evaluations.

For single-node application, we have to support threading. And for users with access to high-performance clusters, we need to rely on the message-passing interface (MPI). The HEP user base of BAT predominantly uses C++, so this will continue to be the main language in which we implement BAT. Support of MPI is rather invasive and has to be considered from the ground up. We therefore plan a complete rewrite of the software for version 2.

This rewrite allows us to achieve a number of goals. While HEP seems to be one of the last strongholds of C++ mainly because of ROOT [2], many users outside of the big collaborations use scripting languages such as python. We intend to provide a lean core version of BAT 2 that does not require ROOT and contains an API to interface with scripting languages.

The last major design decision for version 2 we wish to discuss are the algorithms. We have come to learn that there is no one “do it all” algorithm for every problem and there is no perfect black box. From a user perspective, important questions are

(i) Is it feasible to compute the gradient of the posterior? Do I rely on external libraries that inhibit this calculation?
(ii) How many parameters are relevant to my problem?
(iii) Is the posterior multimodal?
(iv) Do I want to do parameter inference only or do I want to compare models; i.e., is the evidence needed?
(v) How can I get a maximum number of effective samples per time from the computing resources available to me?

We will continue to offer random-walk MCMC but in addition we want to support Hamiltonian Monte Carlo [14, 15, 16], nested sampling [17, 18], importance sampling with population Monte Carlo or variational Bayes [19, 20] and possibly more. Each of these algorithms corresponds to a particular set of answers to the above questions.

Acknowledgments
F. B. is grateful for the support by the computational center for particle and astrophysics (C2PAP) at the Universe Cluster, Munich.

References
[1] Caldwell A, Kollár D and Kröninger K 2009 Comp. Phys. Comm. 180 2197–2209 (Preprint 0808.2552)
[2] Brun R and Rademakers F 1997 Nuc. Inst. Meth. Phys. Research A 389 81–86 URL http://root.cern.ch
[3] Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H and Teller E 1953 J. Chem. Phys. 21 1087–1092
[4] Aad G et al. (ATLAS) 2014 Phys. Rev. D 90 052005 (Preprint 1405.4123)
[5] Jaegle I et al. (Belle) 2015 (Preprint 1502.00084)
[6] Khachatryan V et al. (CMS) 2015 Phys. Rev. B 91 052009 (Preprint 1501.04198)
[7] Agostini M et al. (GERDA) 2013 Phys. Rev. Lett. 111 122503 (Preprint 1307.4720)
[8] Bevan A et al. (UTfit) 2014 JHEP 1403 123 (Preprint 1402.1664)
[9] Caldwell A C, Kollár D and Kröninger K 2010 J. Phys.: Conf. Series 219 032013
[10] Beaujean F, Caldwell A, Kollár D and Kröninger K 2011 J. Phys.: Conf. Series 331 072040
[11] Haario H, Saksman E and Tamminen J 2001 Bernoulli 7 223–242
[12] Brooks S P and Gelman A 1998 J. Comp. Graph. Stat. 7 434–455
[13] Caldwell A and Liu C 2014 (Preprint 1410.7149)
[14] Duane S, Kennedy A D, Pendleton B J and Roweth D 1987 Phys. Lett. B 195 216–222
[15] Neal R M 1994 J. Comp. Phys. 111 194–203
[16] Hoffman M D and Gelman A 2014 J. Machine Learning Research 15 1593–1623

5
[17] Skilling J 2004 *Bayesian inference and maximum entropy methods in science and engineering* 735 395–405
[18] Feroz F, Hobson M and Bridges M 2009 *Mon. Not. Roy. Astr. Soc.* 398 1601–1614
[19] Beaujean F 2012 *A Bayesian analysis of rare B decays with advanced Monte Carlo methods* Dissertation Technische Universität München
[20] Beaujean F and Caldwell A 2013 *(Preprint 1304.7808)*