BAT - The Bayesian Analysis Toolkit

2010 J. Phys.: Conf. Ser. 219 032013

(http://iopscience.iop.org/1742-6596/219/3/032013)

Journal of Physics: Conference Series **219** (2010) 032013 doi:10.1088/1742-6596/219/3/032013

BAT – The Bayesian Analysis Toolkit

Allen C Caldwell[1], Daniel Kollár[2], Kevin Kröninger[3]

[1] Max-Planck-Institut für Physik, München, Germany
[2] CERN, Geneva, Switzerland
[3] II. Physikalisches Institut, Universität Göttingen, Germany

E-mail: daniel.kollar@cern.ch

Abstract.
Bayesian Analysis Toolkit (BAT) is a new toolkit for data analysis based on Bayes' Theorem. BAT is realized with the use of Markov Chain Monte Carlo which gives access to the full posterior probability distribution. Parameter estimation, limit setting and uncertainty propagation are implemented in a straightforward manner. The tool allows to compare models and to estimate the goodness-of-fit using well-established methods.

1. Introduction

The main goals of a typical data anaysis 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 (BAT) [1] is a new toolkit which aims to help the user to meet these goals. It is based on Bayes' Theorem which for a single model has the form

$$P(\vec{\lambda}|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})}{\int P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})\,d\vec{\lambda}}\ , \tag{1}$$

i.e., the probability of parameter set $\vec{\lambda}$ given data $\vec{D}$, the *posterior* probability, is equal to the probability of data given the parameters, also known as the *likelihood*, times the initial probability for the parameters, the *prior* probability[1]. The denominator on the right hand side of the equation is just the integral of the numerator over the allowed region of $\vec{\lambda}$ and it ensures that the posterior probability is normalized. The formula can also be interpreted as a learning rule: The knowledge about the parameters of the model (or the model itself) before the experiment, the prior, is updated using the probability of the new data for different values of the parameters, resulting in posterior knowledge.

Usage of Bayesian inference for the data analysis is well established in the high-energy physics (HEP) community. Even though the algorithms for performing calculations typical for Bayesian analyses are well known, there is no widely accepted and used computer toolkit for handling the numerical calculations. Therefore replication of the non-trivial task of implementing standard algorithms is very common.

[1] Throughout the paper the term probability is used for both probability and probability density.

BAT attempts to collect the standard algorithms and tools used in Bayesian analyses into one coherent framework. The technical implementation has been carried out to fulfill two main requirements:

i) provide a flexible framework which allows formulation of arbitrary models,
ii) provide a reliable and fast code for numerical operations.

BAT is implemented in C++ and comes in a form of a library. It relies on ROOT [2] functionality for input/output and drawing. It consists of a set of classes which provide a general infrastructure, algorithms, output and logging, and a set of extension classes to solve specific (fitting) problems. The 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 or parameter limits. It also facilitates the goodness-of-fit testing and model comparison. Emphasis is placed on Markov Chain Monte Carlo, which is the key tool of BAT.

2. Markov Chain Monte Carlo

Obtaining the posterior probability given by Eq. (1) is generally a difficult task, especially for models with a large number of parameters. Except for a few trivial cases a numerical approach is the only option to scan the posterior, and even then the use of standard techniques can lead to unacceptably long execution times. The practicability of Bayesian inference has been revolutionized by the use of Markov Chain Monte Carlo (MCMC) (see e.g [3, 4]).

2.1. Metropolis

The MCMC can be used to scan very complicated probability distributions in many dimensions by performing a random walk in the allowed parameter space. The original and most popular MCMC algorithm, the Metropolis algorithm [5], is implemented in BAT. The following steps are done to map out a function $f(\vec{x})$:

1. Start at some randomly chosen $\vec{x}_i$.
2. Generate a proposal point $\vec{y}$ according to a symmetric probability density function $g(\vec{x}_i, \vec{y})$.
3. Calculate the values of the function at the current point, $\vec{x}_i$, and the proposal point, $\vec{y}$, and compare them:
   - if $f(\vec{y}) \geq f(\vec{x}_i)$ then set $\vec{x}_{i+1} = \vec{y}$,
   - otherwise generate random number, U, from a flat distribution between $[0, 1]$ and set $\vec{x}_{i+1} = \vec{y}$ if $f(\vec{y})/f(\vec{x}_i) > U$, else set $\vec{x}_{i+1} = \vec{x}_i$.
4. Start again with 1.

For a reasonable proposal function $g(\vec{x}, \vec{y})$ the limiting distribution of the generated chain is $f(\vec{x})$, i.e., the MCMC will be sampling according to $f(\vec{x})$. This allows for the generation of randomly distributed points, in our case sets of parameters of the model, according to a complicated probability density function having an unknown analytic form. All that is required is that $f(\vec{x})$ can somehow be calculated.

For a finite number of steps it is required that the chain reaches a stationary state before it starts sampling according to the $f(\vec{x})$ so that the initial steps do not bias the distribution. Determining whether the chain has converged to the stationary distribution with an acceptable accuracy can be a difficult problem in many cases.

2.2. Convergence

To achieve the convergence of MCMC and find reasonable run parameters a *pre-run* is performed in BAT before the MCMC is used for the analysis of the posterior. In the *pre-run*, several chains are run in parallel with random starting points in the allowed range. The steps in parameter space are done consecutively for each parameter and chain. The proposal function for new steps is chosen flat by default with a range initially set to the width of the allowed range of the parameter. An iteration is defined as the set of steps from an update of the first parameter of the first chain to the last parameter of the last chain. The efficiency for accepting or rejecting new points is evaluated separately for each parameter and chain over a number of iterations. The proposal function ranges are updated every $NumberOfParameters \times 1000$ iterations until an efficiency between 15% and 50% is found for each parameter.

The convergence of Markov chains is monitored via the r-value [6] which should be $r \approx 1$ once the the convergence is reached. The default convergence criterion is $(r-1) < 0.1$ and it has to be fulfilled for all parameters simultaneously as well as for the posterior. The *pre-run* is performed for a minimum number of iterations and is continued either until the chains converged and the efficiency of each parameter is found in the required range, or until a maximum number of iterations is reached. No output is produced during the pre-run. This choice of convergence criteria was found to be robust in the tests performed so far. However, it is by no means unique, and is also not guaranteed to yield the asymptotic Markov Chain. Different criteria for defining convergence will be investigated for future releases of BAT.

The analysis run is performed for a defined number of iterations with run parameters found in the *pre-run*.

2.3. MCMC in Bayesian analysis

The MCMC is used for the extraction of many quantities of interest. At each step for each parameter a 1-dimensional (1-D) histogram is filled with the value of the parameter and the so-called marginalized distribution is obtained. It represents the posterior probability density function (*pdf*) of a single parameter of a model given the data when all other parameters are integrated over and it is defined as:

$$P(\lambda_i|\vec{D}) = \int P(\vec{\lambda}|\vec{D}) d\vec{\lambda}_{j\neq i} \; . \tag{2}$$

In analogy, 2-dimensional (2-D) histograms are filled for every combination of two parameters of the model to obtain marginalized distributions with respect to two parameters. These represent the correlations between two parameters.

In principle, distributions with more than two dimensions can also be filled during the sampling, but since displaying three or more dimensions is rather difficult, this is not done by default. However, it is possible to store the full output of the MCMC for subsequent analysis, which allows to draw higher dimensional distributions if needed.

A user interface supports further operations during the sampling, such as evaluating arbitrary functions of the parameters. A typical use case is uncertainty propagation where one would fill a histogram for any function of parameters, e.g., $y = f(\vec{\lambda})$, and obtain its posterior *pdf*

$$P(y|\vec{D}) = \int P(\vec{\lambda}|\vec{D}) \, \delta(y - f(\vec{\lambda})) \, d\vec{\lambda} \, . \tag{3}$$

By default this interface is used to evaluate the probability distribution for the function being fitted to the data, if applicable, and provides the uncertainty band of the fit.

Additionally, since the walk is performed over the full parameter space, at each step the location of global maximum probability is updated.

3. Other algorithms

3.1. Maximization

As mentioned above, a by-product of the MCMC is the location of the global mode of the posterior *pdf*. However, as MCMC is not optimized for this task, it is neither effective nor accurate. On the other hand, assuming a stationary distribution and a sufficient number of iterations, the MCMC provides a safe estimate of the location of a global maximum, which can serve as a starting point for other minimization algorithms.

A second option for mode finding interfaced to BAT is the ROOT version of Minuit. The user is given full control over the Minuit setup. In the case where MCMC was ran before Minuit, the global mode obtained from MCMC is set as a starting point for Minuit.

Implementation of Simulated annealing, an algorithm similar to MCMC but optimized for mode finding, is currently ongoing.

3.2. Integration

Explicit calculation of the normalization factor in Eq. (1) is not needed if only one model is analyzed. The mode finding does not depend on it at all and the marginalized posterior distributions can easily be normalized on the level of histograms after the MCMC run.

However, the integration is necessary if one needs to perform a Bayesian model comparison. In such case the probability that the model M is true given the data $\vec{D}$ is

$$P(M|\vec{D}) = \frac{P(\vec{D}|M)P_0(M)}{\sum_M P(\vec{D}|M)P_0(M)} = \frac{\int P(\vec{D}|\vec{\lambda},M)\,P_0(\vec{\lambda}|M)d\vec{\lambda}\,P_0(M)}{\sum_M \int P(\vec{D}|\vec{\lambda},M)\,P_0(\vec{\lambda}|M)d\vec{\lambda}\,P_0(M)}\,, \tag{4}$$

i.e., the integrals over the full parameter space need to be evaluated for every model.

A numerical integration over the posterior probability in BAT can be performed using the sampled mean algorithm with and without importance sampling. Optionally it is possible to compile BAT with the CUBA library [7] which provides a set of well tuned routines for integration in many dimensions. The implementation or interfacing of other integration algorithms is planned for the future.

4. Implementation

In BAT the emphasis is put on simplicity of the code which a user needs to be implement for a particular model, and for running of the analysis.

4.1. Model definition

The main class providing the infrastructure and the links to the individual algorithms is `BCModel`. It is a parent class for all user models. The implementation of a particular user model is straightforward and it is done by implementing an extended class which inherits from `BCModel`. To define a model the user has to

1. define each parameter λ_i of the model by setting its name and allowed range using the method:
   int **AddParameter**(**const char** $*$ name, **double** min, **double** max)
2. construct the *likelihood* $P(\vec{D}|\vec{\lambda})$ by implementing the method:
   double **LogLikelihood**(std::vector<**double**> params)
3. construct the *prior* $P_0(\vec{\lambda})$ by implementing the method:
   double **LogAPrioriProbability**(std::vector<**double**> params)

As their names suggest, for the sake of numerical stability the methods `LogLikelihood` and `LogAPrioriProbability` return the natural logarithm of the probability instead of the probability itself.

Once the model has been defined and the data has been loaded (see below) the analysis is performed by calling the methods provided by `BCModel`. This construction allows an arbitrary model definition by using simple functions as well as using a complicated code or calling of external routines.

4.2. Handling of the data

The data in BAT are handled using the class `BCDataSet`. Methods are provided for reading in data from a text file, ROOT tree or from a ROOT histogram. The implementation of user defined data interface is possible.

4.3. Main program

BAT can either be used from a compiled program which is linked against the BAT library or in the ROOT interactive session, e.g., in a ROOT macro. Below is an example of code of the main programm running the analysis of the model `MyModel` which is a user defined model class inheriting from `BCModel`.

```
int main()
{
   // create instance of the model and set its name
   MyModel * model = new MyModel("StandardModel");

   BCDataSet * data = new BCDataSet();          // create new data set
   data -> ReadDataFromFileTxt("data.txt",4); // read data into the data set from
                                               //      a text file with 4 columns
   model -> SetDataSet(data);                  // assign data set to the model

   model -> Normalize();                       // calculate the normalization constant
   model -> MarginalizeAll();                  // run MCMC to get all marginalized distributions
   model -> PrintAllMarginalized("out.ps");    // print all marginalized distributions
                                               // into a PostScript file

   // run Minuit to find the global mode of the posterior,
   // starting at the mode found before (during the MCMC run)
   model -> FindModeMinuit( model -> GetBestFitParameters() );

   model -> PrintResults("results.txt");       // print results into a text file

   return 0;
}
```

A simple shell script is provided which generates a sample source code for a model class and a main program together with a makefile.

All the implemented algorithms can be tuned by the user from within the code. At the same time reasonable defaults are set for most of the parameters. The defaults allow one to quickly perform a simple analysis, but are by no means perfect for all problems and will eventually require tuning by the user.

5. Output

BAT provides several ways to output the information of interest after the analysis is done.

5.1. Text output

A file containing the summary information about each model and its parameters is created if requested by the user. It displays the results of the analysis such as the global mode, the mean and modes of the marginalized distributions, and the uncertainties on the parameters. If model

comparison was performed, the total probability for all models and the Bayes factors for every combination of two models can be summarized in another text file.

In addition, summary and debugging information is printed out on the standard output and/or a logfile during the program/macro running. The level of verbosity of the messages can be set by the user.

5.2. ROOT file

ROOT has established itself as the main computer analysis framework in HEP community. The functionality provided by ROOT class `TFile` is used for storing information from BAT run in a by now standard format. BAT uses a single `TFile` for each model to store different kinds of information if requested by the user:

- a `TTree` object with the analysis summary information,
- a `TH1D` object (1-D histogram) for every marginalized distribution wrt. one parameter and a `TH2D` object (2-D histogram) for every marginalized distribution wrt. every combination of two parameters,
- a separate `TTree` object for each Markov chain in the main run.

The `TTree`, `TH1D` and `TH2D` objects can further be easily analysed within ROOT using the ROOT functionality. Using simple ROOT commands the histograms can be renormalized, rebinned or fitted, one can or extract quantiles and do a pretty drawing, etc.

A `TTree` can directly be asked for the distribution of parameter(s) to obtain the marginalized distribution wrt. one, two or three parameters and study complicated correlations. In addition, one can calculate distributions for arbitrary functions of parameters. A stored Markov Chain allows one to perform a full analysis of the posterior "offline". This can often be dramatically faster than having to rerun the BAT analysis, especially for complicated problems with a large number of parameters. On the other hand, disadvantages include the amount of disk space required for storing the full chains, which grows with the number of parameters and number of iterations.

5.3. Plots

BAT contains a number of classes for presentation of distributions in the form of plots. The drawing is done using the ROOT functionality and therefore it is possible to export the figures in any format supported by ROOT.

Predefined drawing styles can be used to plot the 1-D and 2-D marginalized distributions and print them into separate files or into a single multi-page PostScript file. By default the figures with 1-D marginalized distributions show the central 68% probability interval and indicate the location of the mean, the median and the global mode. In addition the median $\pm$ the uncertainty calculated from the central 68% interval is shown in numbers. The 2-D distributions are drawn as correlation contour plots with the location of the global mode indicated. Other drawing styles can be chosen by the user for both 1-D and 2-D distributions. Figure 1 shows examples of standard output for the marginalized *pdf*s.

It is not always the case that the central 68% interval is the quantity of interest for a particular distribution. When requested, BAT can plot the upper or lower probability limit instead.

In case of an asymmetric or multi-modal *pdf* the smallest probability interval is usually a more meaningful uncertainty interval (set of intervals) for the parameter. It is possible to draw the smallest interval(s) for 1-D and 2-D distributions. Figure 2 shows an example for multi-modal marginalized distributions.

In the particular case of fitting a 1-D function, $f(\vec{\lambda}, x)$ of parameters $\vec{\lambda}$ to the data, during the MCMC run BAT automatically calculates the probability distribution of f at every x following the posterior *pdf* for $\vec{\lambda}$ given the data. This distribution represents the probability of the true

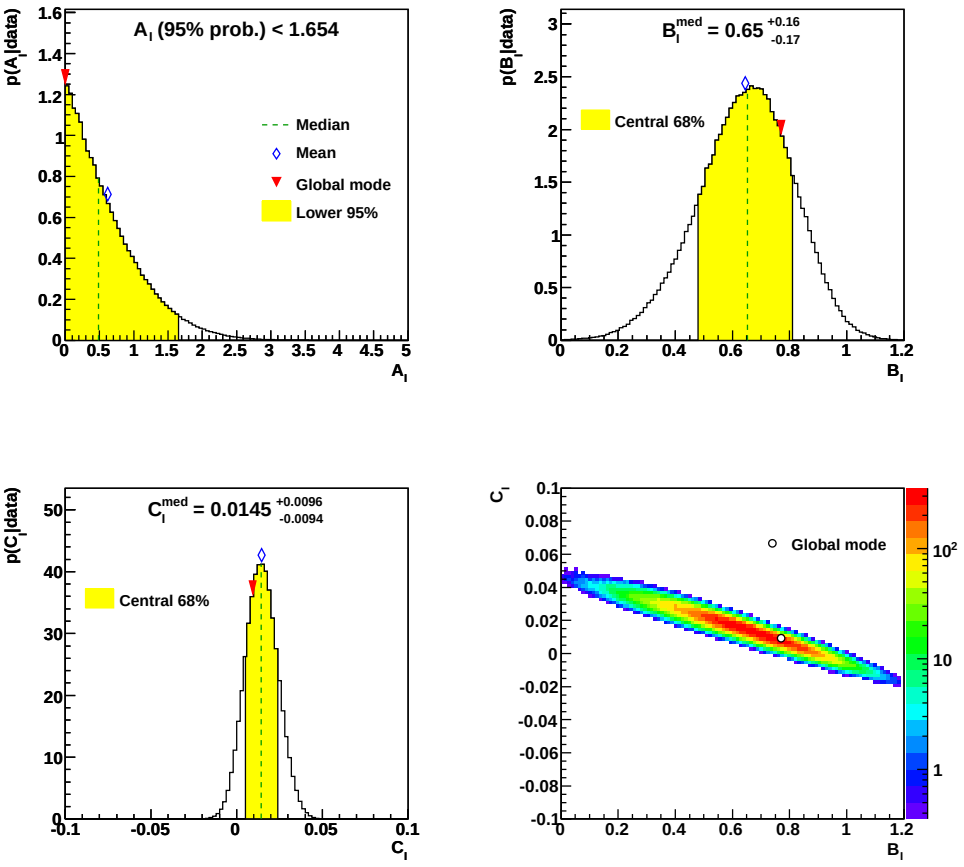

Figure 1. Examples of standard output distributions from BAT: (top-left) 1-D marginalized distribution with the lower 95% probability interval; (top-right) and (bottom-left) 1-D marginalized distributions with the central 68% interval; (lower-right) 2-D marginalized distribution (correlation) with the global mode.

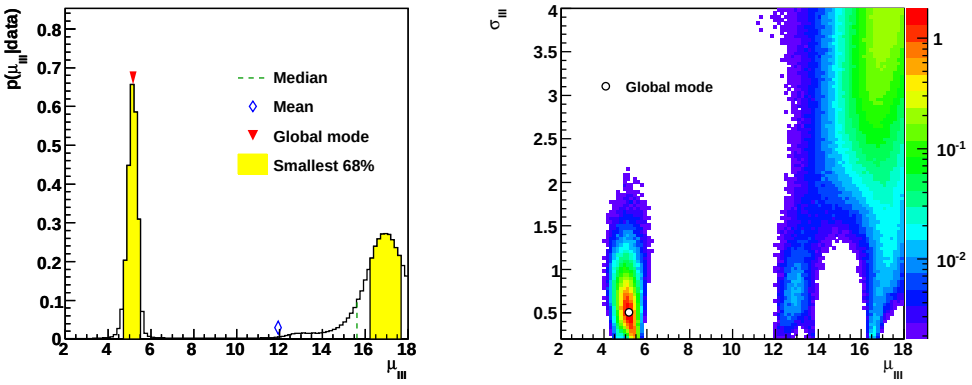

Figure 2. Examples of output for multi-modal distributions: (left) 1-D marginalized distribution with the smallest 68% probability interval; (right) 2-D marginalized distribution (correlation) with the global mode.

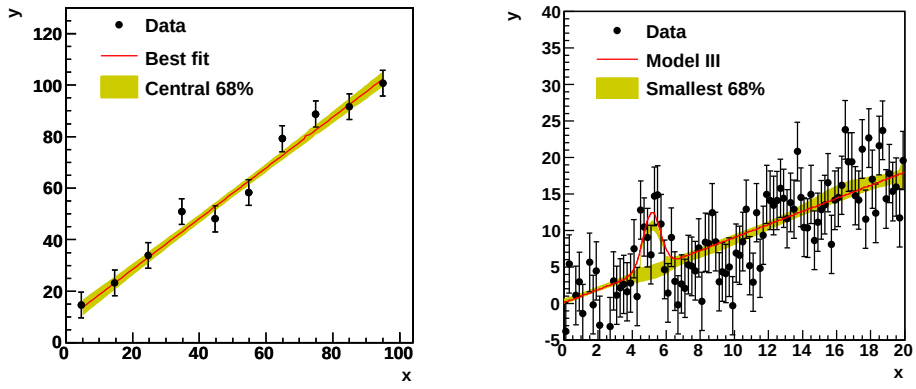

Figure 3. Example displays of different model fits to the data including the uncertainty bands: (left) central 68% band, (right) smallest 68% band.

$f(x)$ given the model and the data. From this distribution the uncertainty band for the fit can be extracted. By default the central 68% interval is used. An example of a central 68% uncertainty band is shown in Fig. 3 (left).

Multi-modal distributions can also occur for *pdf*s of the true $f(x)$. In such a case the smallest 68% interval(s) can be calculated instead. Currently, BAT can plot uncertainty multi-band with at most two bands. An example of such a multi-band is shown in Fig. 3 (right).

6. Fast fitter classes

For standard fitting problems BAT provides easy to use classes which support the common type of fits with minimal effort without the need to implement a user model class. The available fast fitter classes are:

- `BCGraphFitter`
  Fits data assuming Gaussian uncertainties (the *likelihood* is constructed as a product of Gaussians). The data are provided in form of a ROOT `TGraphErrors` object.
- `BCHistogramFitter`
  Fits a histogram assuming Poissonian uncertainties (the *likelihood* is constructed as a product of Poissonians). The data are provided in form of a `TH1D` object.
- `BCEfficiencyFitter`
  Fits an efficiency assuming binomial uncertainties (the *likelihood* is constructed as a product of binomials). The data are provided in form of two `TH1D` objects and the use of this fitter is only appropriate if one of the histograms represents a subset of the other.

In all three cases the fit function is defined as a ROOT `TF1` object. Usage of the standard ROOT classes allows a trivial use of the fast fitters. An example code for the efficiency fitter is shown below.

```
TH1D * hfull = ...                  // histogram containing the full set of entries
TH1D * hsub = ...                   // histogram containing the subset of entries
TF1 * f = ...                       // the fit function

// create an instance of the efficiency fitter
BCEfficiencyFitter * beff = new BCEfficiencyFitter(hfull,hsub,f);

beff -> Fit();                      // perform the fit : run the MCMC and the Minuit
beff -> DrawFit("",true);           // draw the result of the fit
```

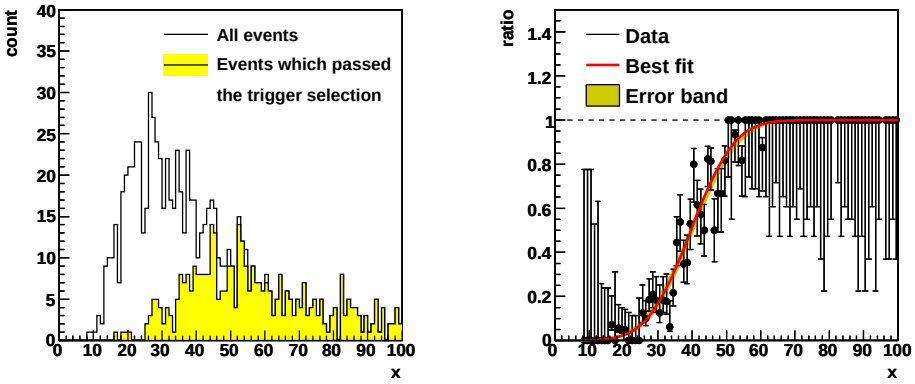

Figure 4. Example distributions of events as a function of some variable x before and after the trigger selection (left) and the fit of the trigger efficiency performed using BAT (right).

The *prior*s are assumed to be flat for all parameters. User can redefine them if needed. Since the three fitters inherit from `BCModel` all its methods can be used to extend the analysis, e.g., normalization, model comparison, pretty printing.

A typical HEP usecase for the `BCEfficiencyFitter` is a fit of the trigger efficiency. The example in Fig. 4 shows the histograms used as the input data to be fitted (left) and the result of the fit using an error function (right).

Inclusion of many more predefined model classes for other frequently encountered fitting problems is planned in the future versions of the code.

7. Conclusions

BAT is a new toolkit for Bayesian analysis which allows one to solve simple statistical problems like function fitting as well as complex data vs. theory comparison. It attempts to fill the gap in the market of statistical tools. It collects common tools and algorithms used in Bayesian analyses and provides a modular and easy to use framework to phrase arbitrary statistical problems. Still, the main responsibility for meaningful results is on users since they have to define the model.

Currently, work is ongoing on implementation of additional algorithms into BAT, such as Simulated Annealing or an interface to models defined using RooFit [8], which is part of ROOT. In addition, work has started on the restructuring of the code to make it even more modular and to allow easier implementation of new algorithms. Also, discussions have started on a possibility to include BAT into ROOT in the future.

Many improvements are planned for future releases of BAT, such as implementation of standard predefined priors, more MCMC diagnostics and checks, performance tuning, more algorithms for integration, MCMC, maximization and others. One additional planned improvement particularily interesting for users is a compilation of model definitions for frequently encountered analysis problems allowing for a quick setup of the analysis code. This effort has started with the fast fitter classes and will continue in the future by adding more complicated models.

BAT is available from `http://www.mppmu.mpg.de/bat`.

References

[1] Caldwell A, Kollár D and Kröninger K 2009 *Comp. Phys. Comm.* doi:10.1016/j.cpc.2009.06.026. (*Preprint* `0808.2552`)

[2] Brun R and Rademakers F 1997 *Nuclear Instruments and Methods in Physics Research A* **389** 81–86 (see also `http://root.cern.ch`)

[3] Karlin S and Taylor H 1975 *A First Course in Stochastic Processes* (Academic Press)
[4] Gilks W R, Richardson S and Spiegelhalter D J 1996 *Markov Chain Monte Carlo in Practice* (Chapman)
[5] Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H and Teller E 1953 *J. Chem. Phys.* **21** 1087–1092
[6] Gelman A and Rubin D B 1992 *Statistical Science* **7** 457–472
[7] Hahn T 2005 *Computer Physics Communications* **168** 78–95 (*Preprint* `hep-ph/0404043`)
[8] Verkerke W and Kirkby D 2003 (*Preprint* `physics/0306116`)