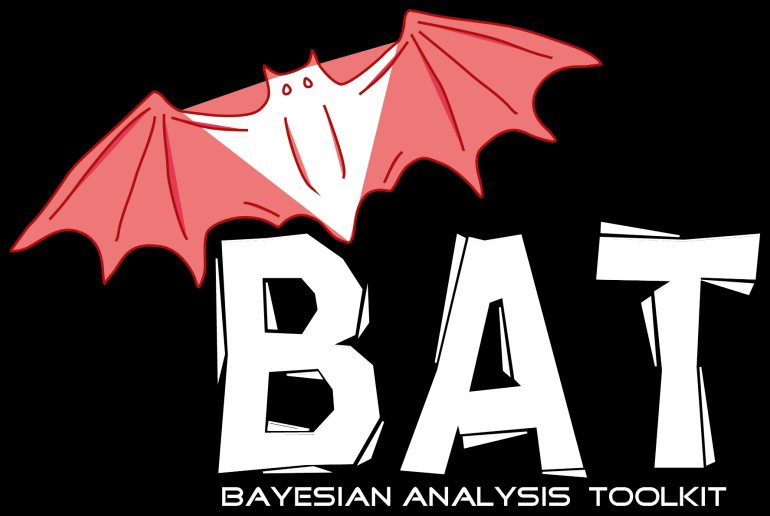


The Bayesian Analysis Toolkit

Allen Caldwell (MPI Munich), Daniel Kollár (CERN), Kevin Kröninger (Uni. Göttingen)

26.03.2008

CHEP'09 | Prague | Czech Republic | 21-27 March 2009

BAT → Software package for solving of statistical problems using Bayesian approach

Bayes' formula for parameter estimation $$p(\vec{\lambda}\,|\,\vec{D}) = \frac{p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})}{\int p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})\; d\vec{\lambda}}$$

Motivation:

- many of us have done Bayesian analyses in HEP always having to implement the numerical algorithms and tools by ourselves → generally non-trivial
- create a package/toolkit to take care of that

The idea behind BAT

- Merge common parts of every Bayesian analysis into a software package
- Provide flexible environment to phrase arbitrary problems
- Provide a set of well tested/tuned numerical algorithms and tools
- C++ based framework (flexible, modular)
- Interfaces to ROOT, Cuba, Minuit, user defined, ...

The idea

$$p(\vec{\lambda}\,|\,\vec{D}) = \frac{p(\vec{D}\,|\,\vec{\lambda})\,p_0(\vec{\lambda})}{\int p(\vec{D}\,|\,\vec{\lambda})\,p_0(\vec{\lambda})\,d\vec{\lambda}}$$

USER DEFINED

- create model
- read-in data

MODEL INDEPENDENT
(common tools)

- normalize
- find mode / fit
- test the fit
- marginalize wrt. one or two parameters
- compare models
- provide nice output

Define MODEL

- define parameters $\vec{\lambda}$
- define likelihood $p(\vec{D}\,|\,\vec{\lambda})$
- define priors $p_0(\vec{\lambda})$

Read DATA

- from text file, ROOT tree, user defined

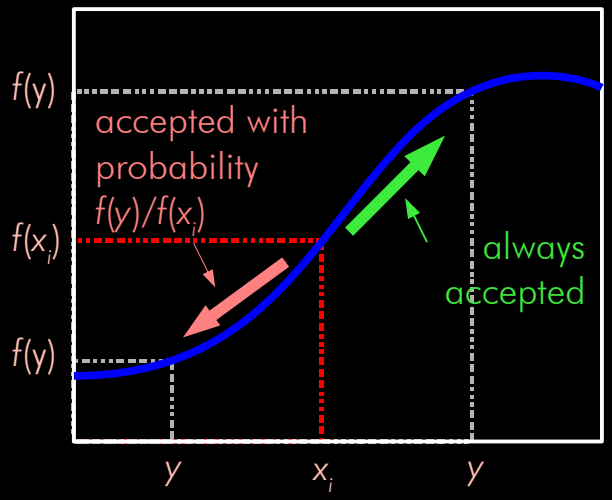
Common tools

- Posterior mapping → Marginalization
  - Markov Chain Monte Carlo (MCMC)
    - key tool in the package
    - lot of emphasis put on efficiency, performance and validation

- Integration
  - Monte Carlo (sampled mean), Cuba (Vegas, ...)
- Maximization
  - Monte Carlo, MCMC, Minuit, Simulated Annealing
- Model testing
  - Posterior comparison, K-factors, p-value calculation
- User interface
  - simple model definition
  - standard output: text output, plots, ROOT histograms and trees, ...

MCMC — Metropolis algorithm

- In BAT implemented Metropolis algorithm
- Map positive function **f(x)** by random walk towards higher probabilities
- Algorithm:
  - Start at some randomly chosen x_i
  - Randomly generate y
  - If $f(y) \geq f(x_i)$, set $x_{i+1} = y$
  - If $f(y) < f(x_i)$, set $x_{i+1} = y$ with probability $p = \frac{f(y)}{f(x_i)}$
  - If y not accepted, stay where you are, i.e. set $x_{i+1} = x_i$
  - Start over

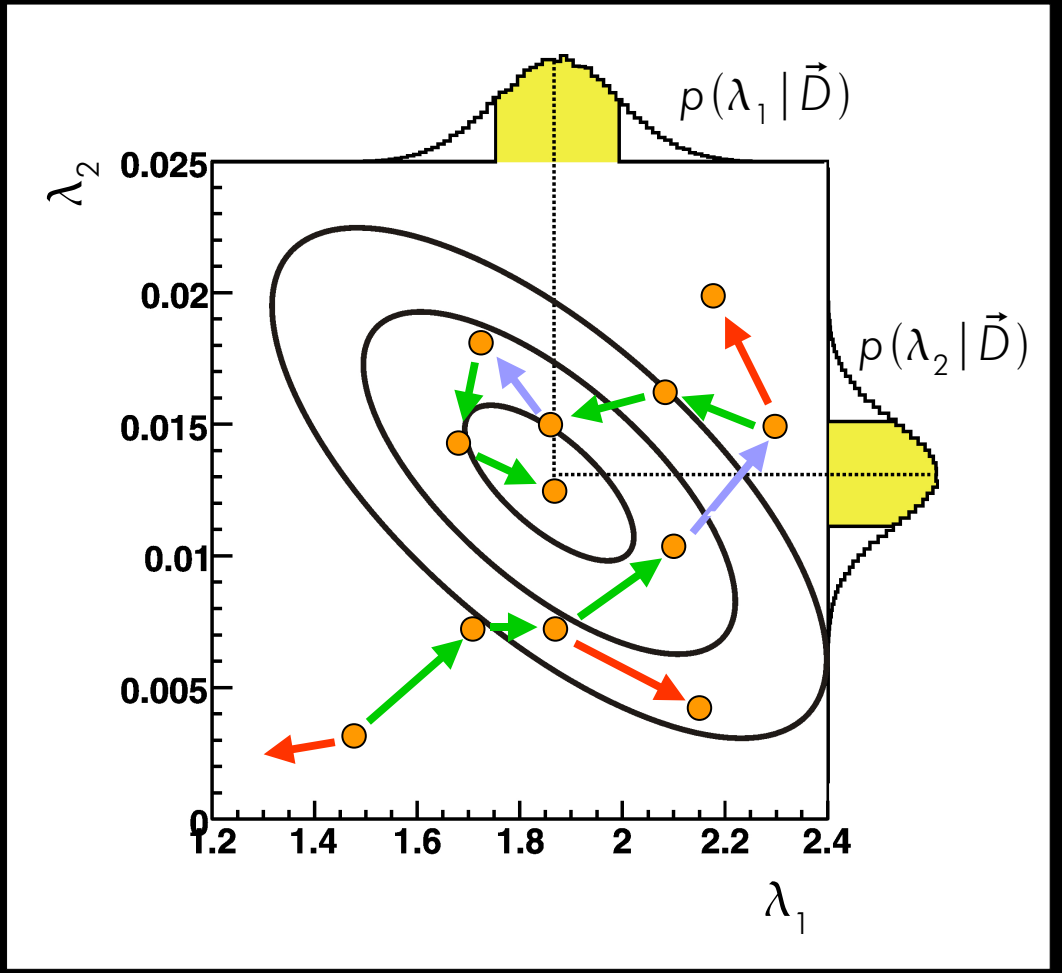


- Sampling is enhanced in regions with higher values of **f(x)**

Scanning parameter space with MCMC

- In BAT, use MCMC to scan parameter space
  $f(\vec{\lambda}) = p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})$
- MCMC converges towards underlying distribution
  - Determining of the overall probability distribution of the parameters $p(\vec{\lambda}\,|\,\vec{D})$
- Marginalize wrt. individual parameters while walking → obtain
  $p(\lambda_i\,|\,\vec{D}) = \int p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})\, d\vec{\lambda}_{j\neq i}$
- Find maximum (mode)
- Uncertainty propagation
  - calculate and store value of any function of parameters λ

$$p(\vec{\lambda}\,|\,\vec{D}) = \frac{p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})}{\int p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})\, d\vec{\lambda}}$$



Details of MCMC implementation

Running several chains in parallel (default is 5)

- Start at random locations in allowed parameter space
- Iniitialize chains by doing a pre-run to achieve convergence
  - Convergence defined using *r*-value (*Gelman & Rubin, StatSci 7, 1992*)
    - Ratio of the mean of the RMS values of the probability and the RMS of the mean values
    - Convergence criterion $|r\text{-}1| < 0.1$
- Steps in parameter space done consecutively for each parameter and chain
- Proposal function for new steps is chosen flat with varying ranges
- The efficiency for accepting new point is evaluated for each parameter and chain over last (npar+1)*1000 iterations and the step size is adjusted to increase the performance
  - If efficiency > 50%, increase the step size
  - If efficiency < 15%, decrease the step size
- use MCMC only after pre-run has ended, convergence was reached and all parameter step-sizes have been adjusted

Most parameters can be set by the user

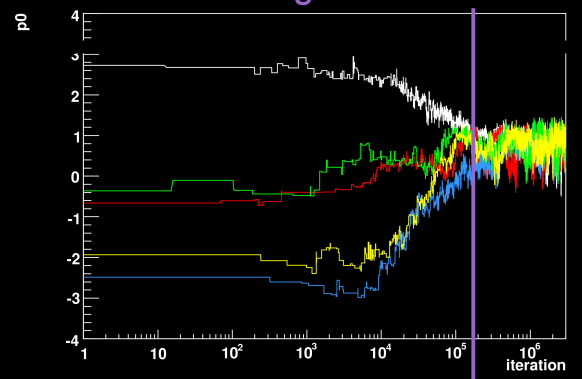
Analysis of Markov Chain

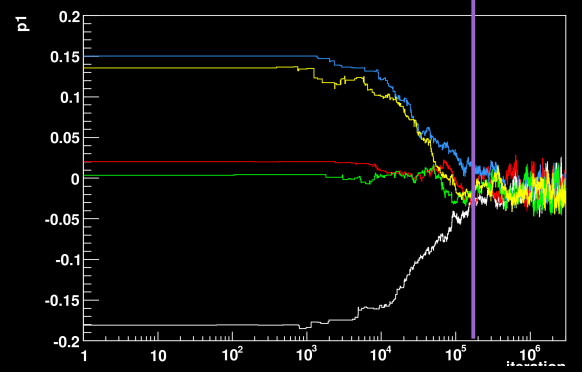
- Danger of non-convergence still remains
- the full chain(s) can be stored for further analysis and parameter tuning
- available formats: text file, ROOT Tree
  - allows direct usage of standard ROOT tools for analysis
  - e.g. fast access to marginalized distributions

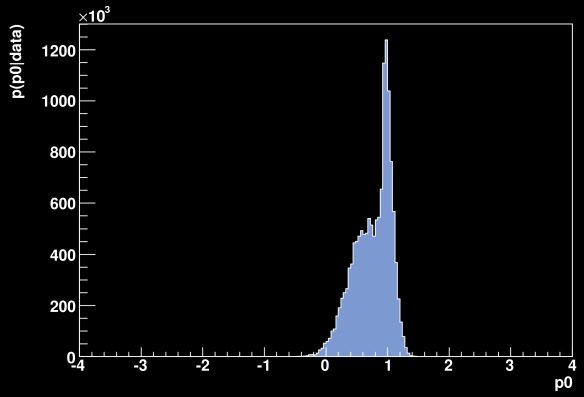
```
root[1] chain0 -> Draw("p0")
root[2] chain0 -> Draw("p1:p2")
```

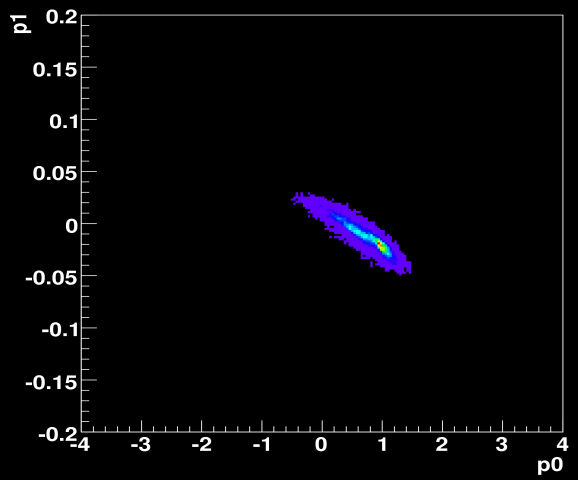
- Markov Chain contains the complete information about the posterior probability (except for the normalization)

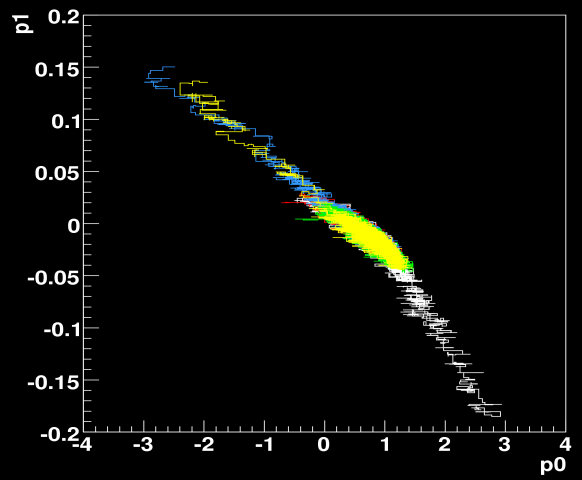
convergence reached





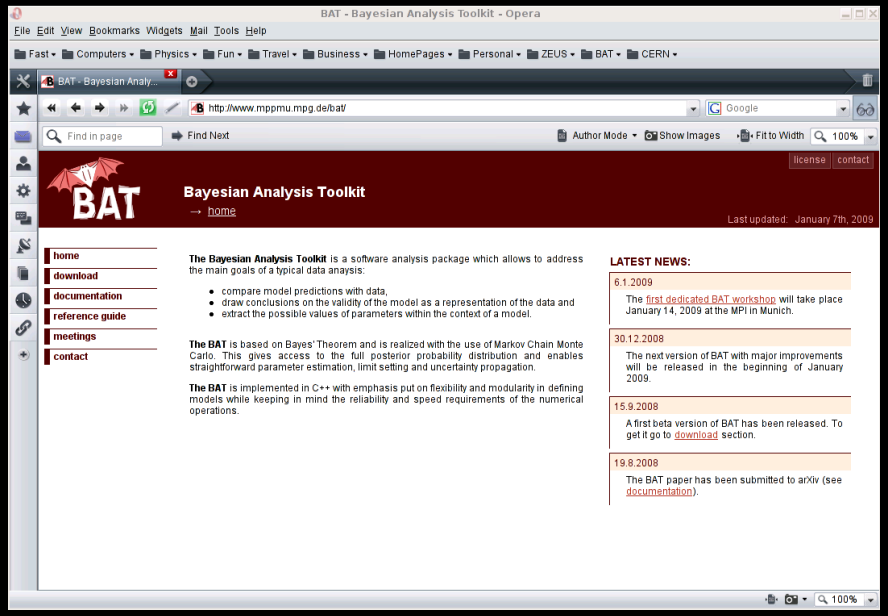






BAT distribution

- download from http://www.mppmu.mpg.de/bat
- comes in the form of shared library
  (plus a .rootmap file for interactive ROOT session)
- depends on ROOT I/O and drawing functionality
  - can in principle be removed if there's need
- can be compiled with Cuba support
- BAT contains 15 classes at the moment which provide:
  - main infrastructure
  - algorithms
  - output and logging
  - extension classes to solve specific (frequent) fitting problems



BAT user code

Define a model class inheriting from a base model class **BCModel**:

- add parameters:
- implement likelihood:
- implement prior:

```
AddParameter("p0",-4,4);
double LogLikelihood(vector<double> params)
double LogAPrioriProbability(vector<double> params)
```

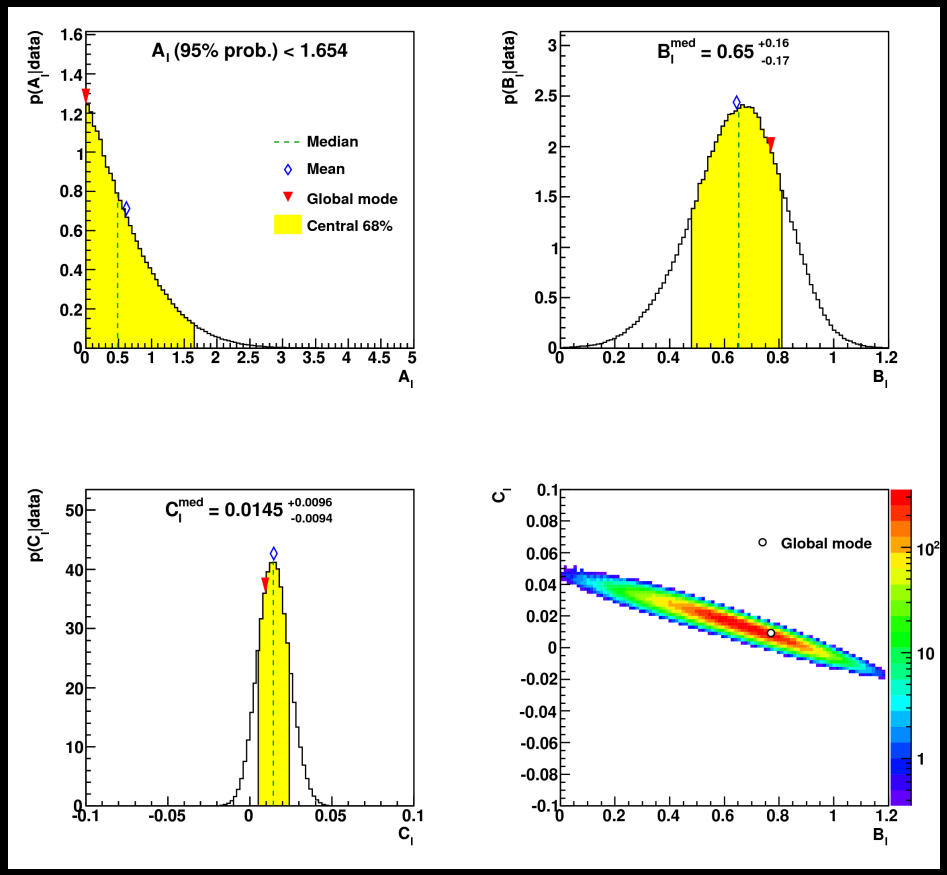
Main program/macro:

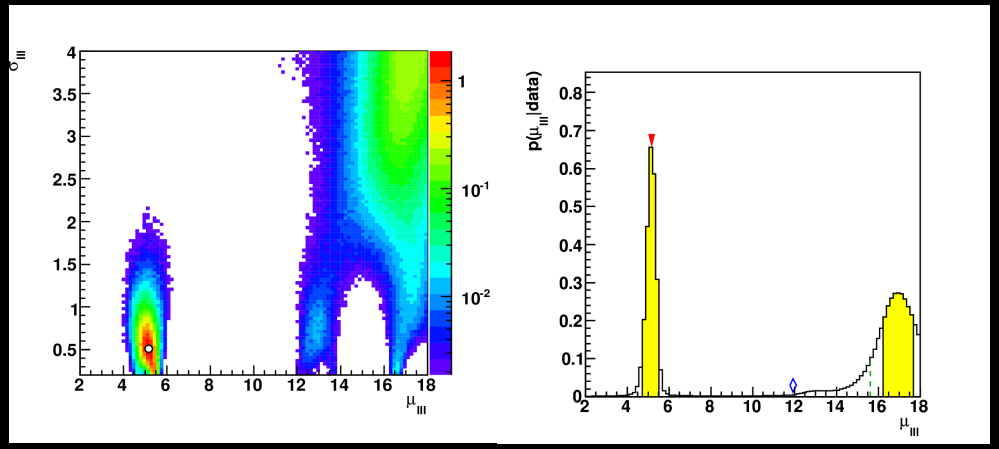
- create instance of the model:
- read data:
- assign data to the model:
- do the analysis:

```
MyModel * mm = new MyModel("Model 0");

BCDataSet * data = new BCDataSet();
data->ReadFromFileTxt("file.txt",4);

mm->SetDataSet(data);

mm->Normalize();
mm->MarginalizeAll();
mm->FindModeMinuit( mm->GetBestFitParameters() );
mm->PrintAllMarginalized("distributions.ps");
etc.
```

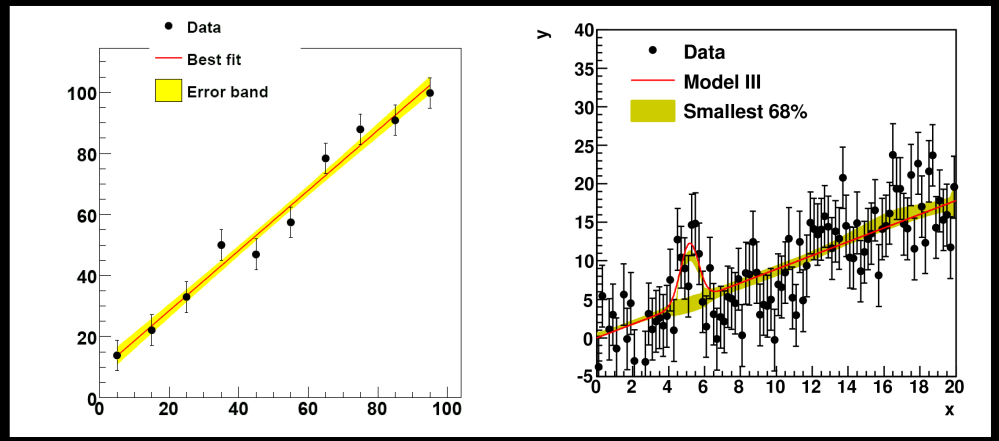
Output

- marginalized distributions 1D and 2D
  - probability intervals, contours
  - probability limits
  - mode, mean, median, rms





- uncertainty bands



- MCMC itself
- information about local maxima and uncertainty intervals/probability limits in text file or ROOT file

Simple fitter classes

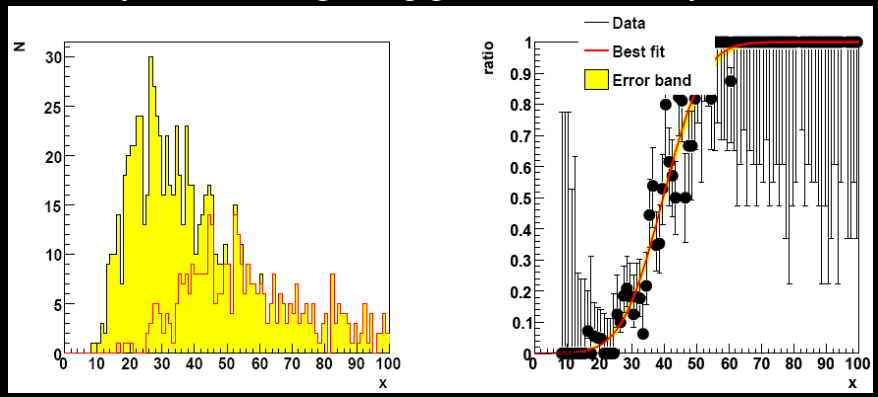
Three **BCModel** extensions for standard fitting problems: (using flat priors)

- **BCGraphFitter** – fits graph (TGraphErrors) assuming Gaussian uncertainties
- **BCHistogramFitter** – fits histogram (TH1D) assuming poissonian uncertainties
- **BCEfficiencyFitter**
  - fits efficiency – a ratio of two histograms (TH1D) where numerator is a subset of denominator – assuming binomial uncertainties

Simple use:

```
TH1D * hfull = ...;
TH1D * hpass = ...;
TF1 * f = ...;
BCEfficiencyFitter * beff =
      new BCEfficiencyFitter(hfull,hsub,f);
beff -> Fit();
beff -> DrawFit("",true);
```

Example: fitting trigger efficiency



Since all three fitters inherit from **BCModel** all other standard class methods can be used, e.g normalization, model comparison, pretty printing etc.

BAT status and plans

- development version 0.2 of BAT was released in January
  - received a lot of important feedback from first users
  - **next release with few small updates coming soon**
- since no detailed manual available yet, mostly oriented to experts at the moment
- lot of things to be added
  - standard predefined priors
  - MCMC diagnostics and checks
  - performance tuning
  - other algorithms for integration, MCMC, maximization, ...
  - library of other standard problems
- interfacing BAT with RooStats
  - `BCRooStatsInterface` class implemented by RooStats team
  - validation ongoing
- current BAT code structure has historical origin – far from being optimal, hard to extend
  - **restructuring is on the way** while having in mind the interactions with RooStats
  - looking into possibility to include BAT into ROOT

Conclusions

- BAT allows to solve simple statistical problems like function fitting as well as complex Data vs. Theory comparisons and parameter estimation
- BAT collects common tools used in Bayesian analyses in a single modular framework
- Markov Chain Monte Carlo is the key tool
- Even though Bayesian, not rejecting non-Bayes algorithms and tools
  - e.g. p-value
  - ready to add more useful tools
- Still, the main responsibility for meaningful results is on users, since they have to define the model, we only provide the tools (and some suggestions)

- BAT attempts to fill the gap on the market with statistical tools
  - clearly a challenge to convince the users
- Also the manpower situation is not optimal but should improve in near future
- we have started discussions with other teams doing Bayesian analyses and got positive feedback and eventually they would also like to contribute