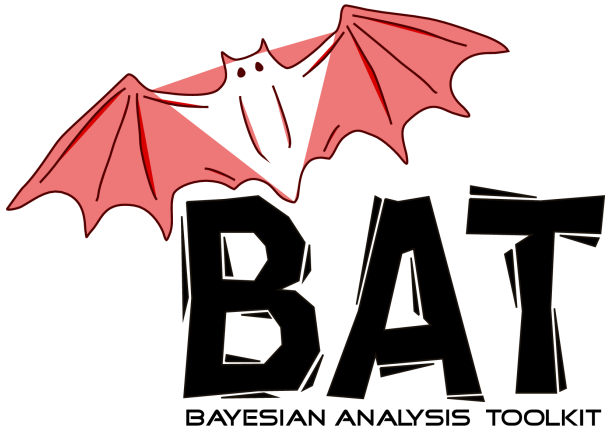


Tutorials

3 tutorials:

Day 1: introduction to Bayesian analysis and BAT, basic examples

Day 2: limit setting, including systematics and priors

Day 3: hypothesis testing, counting experiment

Basics of Bayesian approach

Procedure of learning from experiment: (for a fixed model)

$$p(\vec{\lambda} \mid \vec{D}) \propto p(\vec{D} \mid \vec{\lambda})\, p_0(\vec{\lambda})$$

$p_0(\vec{\lambda})$ - probability of the parameters $\vec{\lambda}$ — **Prior**

- *containing all the knowledge before the experiment*
- *defines allowed ranges for parameters*
- *contains preconceptions about possible values of parameters*

$p(\vec{D} \mid \vec{\lambda})$ - probability of the data $\vec{D}$ given the parameters $\vec{\lambda}$ — **Likelihood**

- *how likely is the data a representation of the model*
- *contains information from theory as well as modelling of experimental conditiona*

$p(\vec{\lambda} \mid \vec{D})$ - probability for parameters $\vec{\lambda}$ given the data $\vec{D}$ — **Posterior**

- *result of the analysis containing all the knowledge about the parameters after the experiment*

After adding a normalization

→ **Bayes formula**

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

Posterior probability

Posterior probability density contains all information

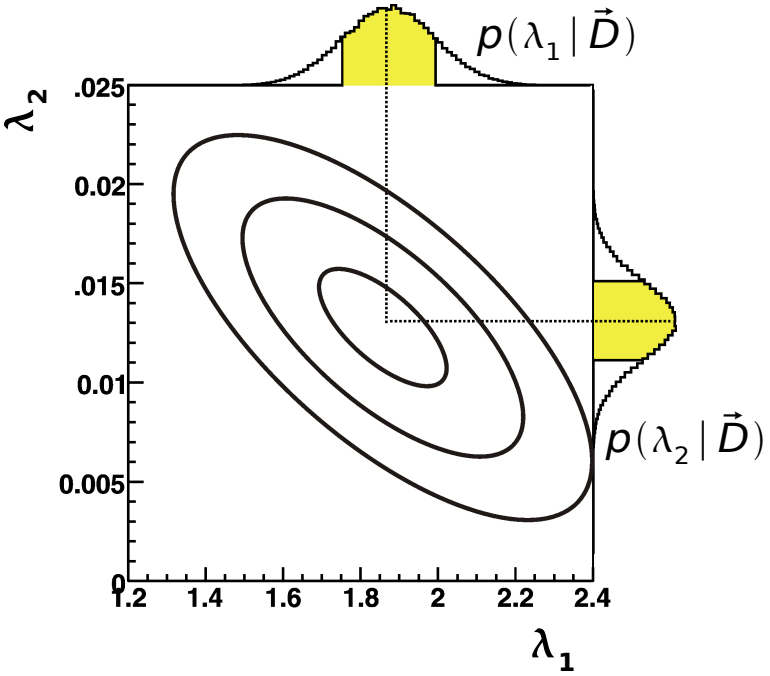
Typically, several quantities are extracted:

- mode, mean, variance, ...
- marginalized posterior distributions
  - posterior integrated over all but one (two, ...) parameter(s)

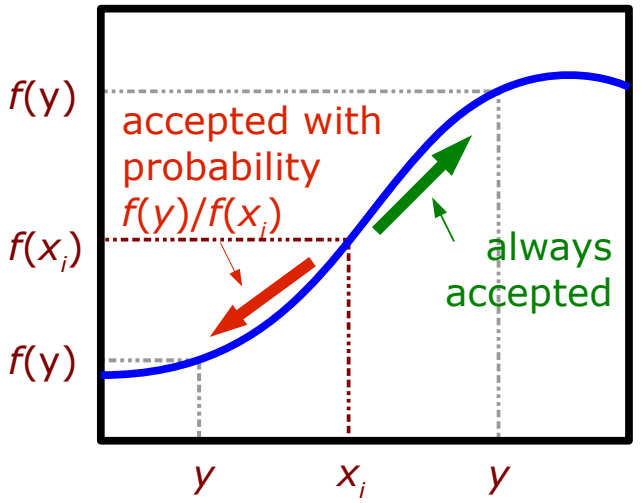
$$p(\lambda_i \mid \vec{D}) = \int p(\vec{\lambda} \mid \vec{D}) \prod_j d\lambda_{j \neq i}$$

  - generally non-trivial and in many cases practically impossible to do using standard techniques
  - key tool:

    ➡ Markov Chain Monte Carlo



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 to $x_{i+1} = y$ 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)***

MCMC: an example

- mapping an arbitrary function:
  $f(x) = x^4 \sin^2 x$

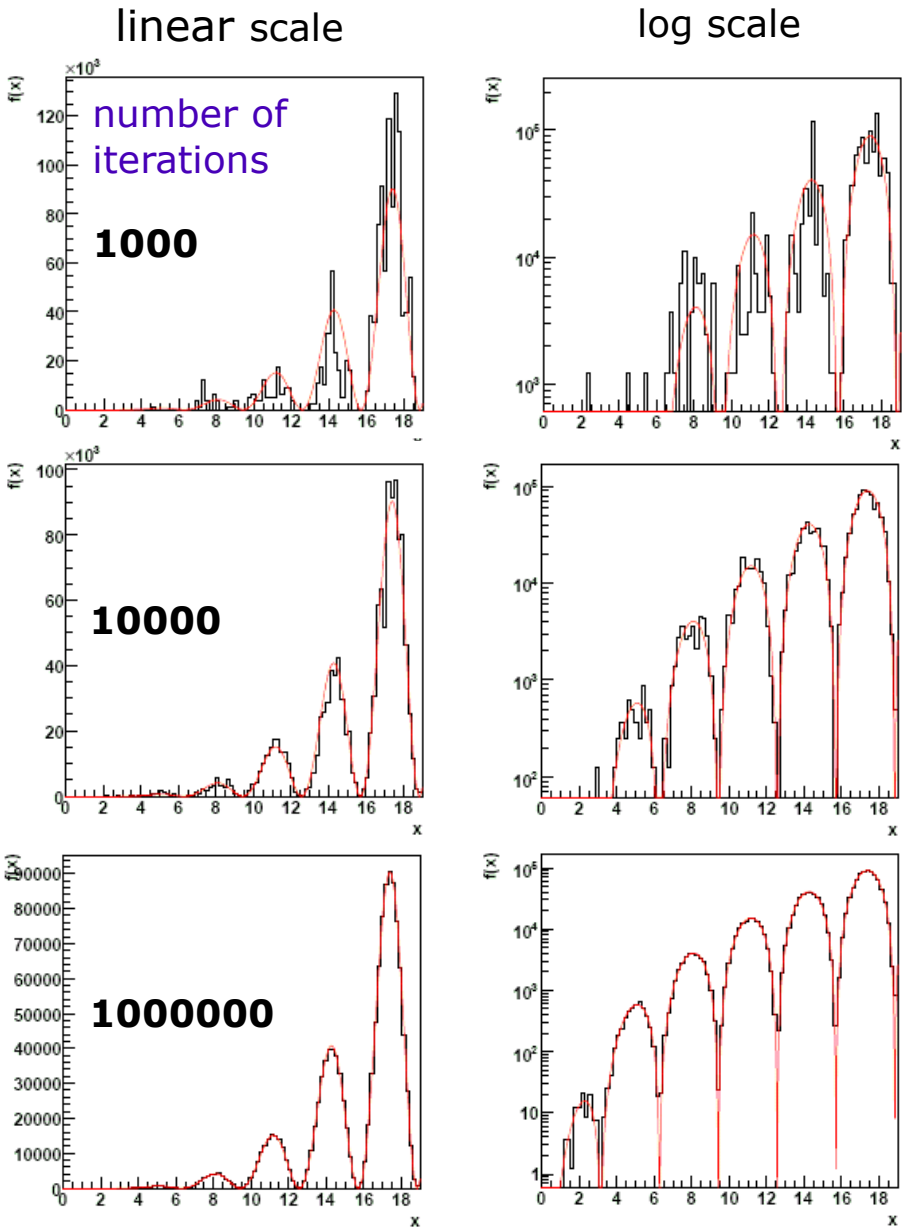
- distribution sampled by MCMC in this case quickly converges towards the underlying distribution

- **mapping of complicated shapes with multiple minima and maxima**

Note:

- MCMC has to become stationary to sample from underlying distribution

- in general the convergence is a non-trivial problem



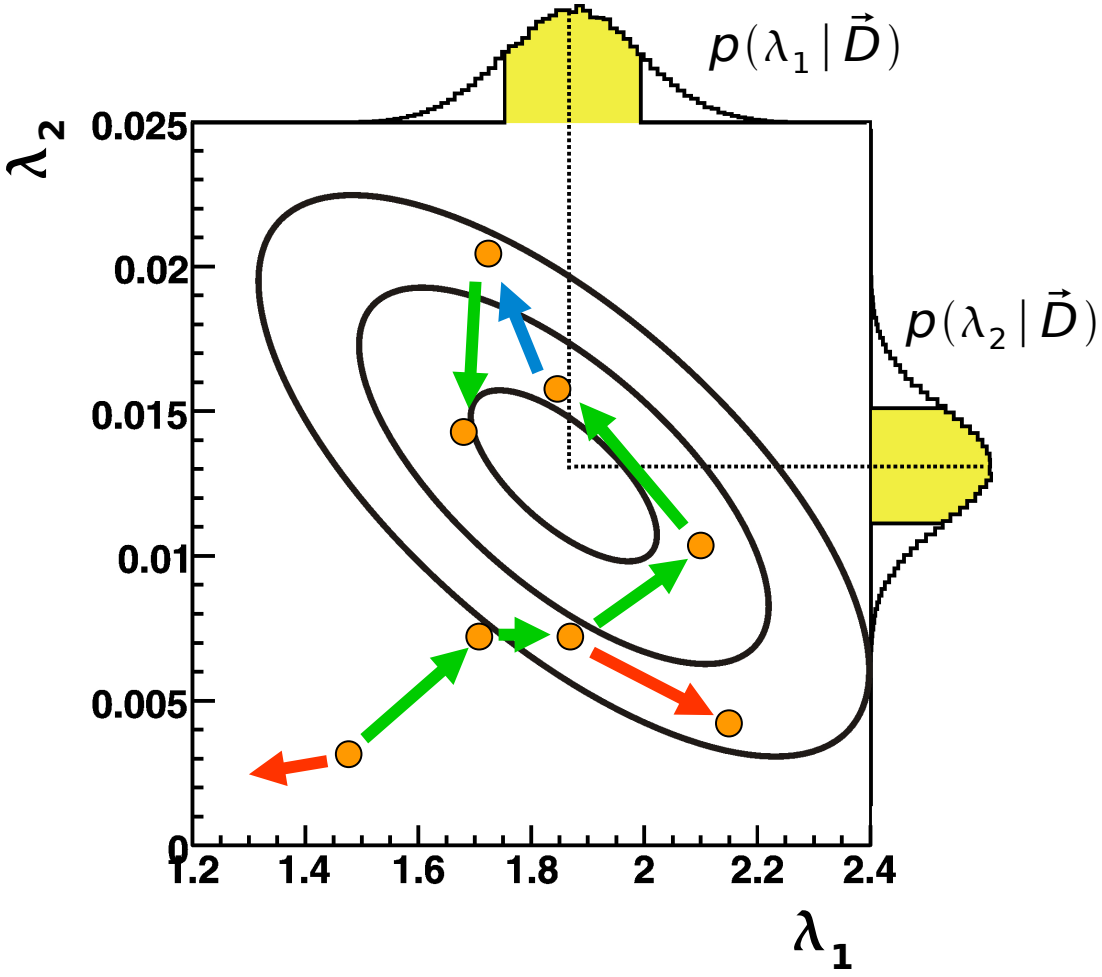
Scanning parameter space with MCMC

- Use MCMC to scan parameter space of $\vec{\lambda}$
- $f(\vec{\lambda}) = p(\vec{D}\,|\,\vec{\lambda})\; p_0(\vec{\lambda})$
- 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

$$p(\lambda_i\,|\,\vec{D}) = \int p(\vec{\lambda}\,|\,\vec{D}) \prod_j d\lambda_{j\neq i}$$

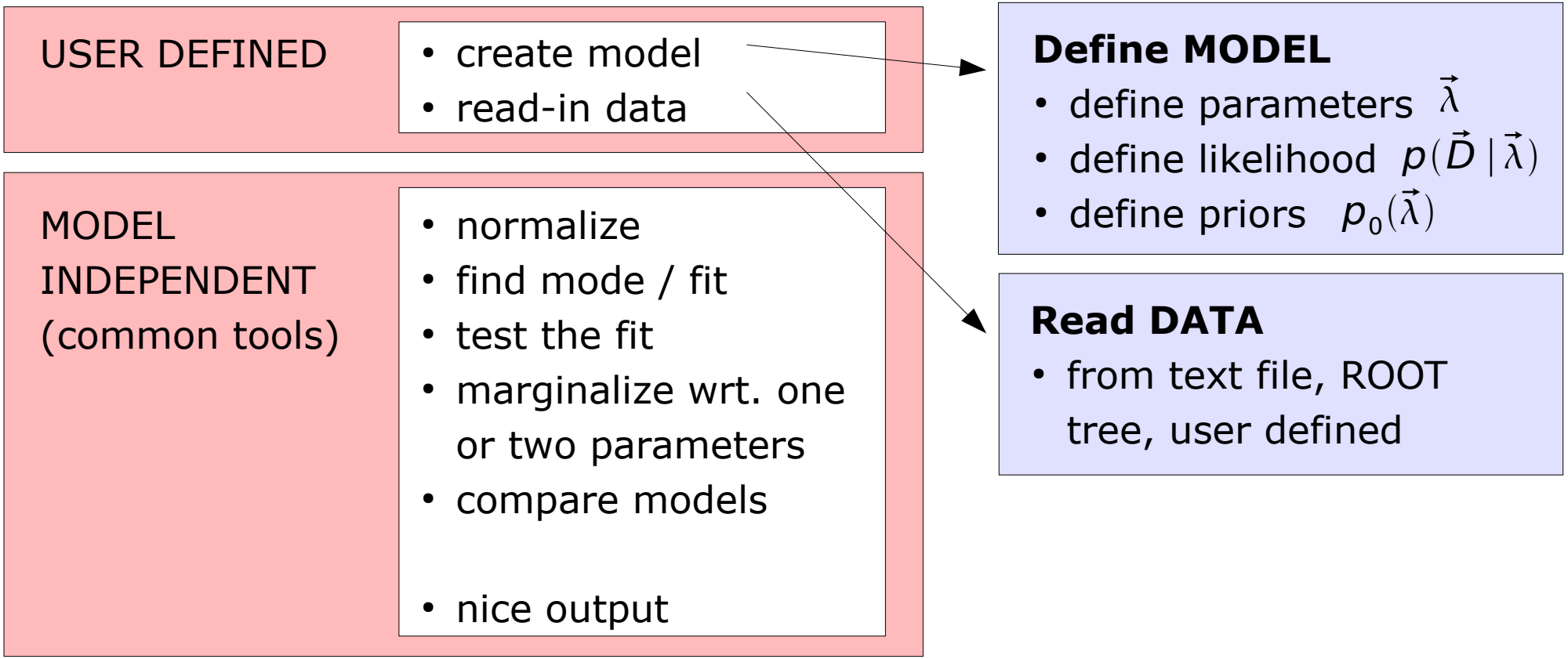
- Find maximum (mode)
- Uncertainty propagation

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



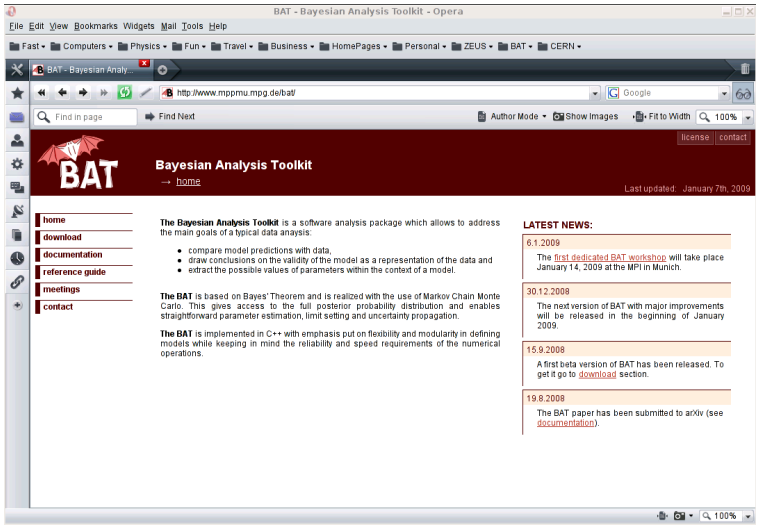
Idea behind BAT

- Provide flexible environment to phrase arbitrary problems
- Provide set of numerical tools
- C++ based framework (flexible, modular)
- Interfaces to ROOT, Cuba, Minuit, user defined, ...



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



Tutorials – DAY 1

BAT examples

- examples are part of BAT distribution
- demonstrate basic functionality and program/macro structure

ROOT macros:

01 simple GraphFitter
- function fitting assuming Gaussian uncertainties

02 HistogramFitter
- function fitting assuming Poisson uncertainties

03 advanced GraphFitter
- as 01 but for more functions at the same time
- model comparison

04 EfficiencyFitter
- function fitting assuming Binomial uncertainties

Compiled program:

05 advanced fitter
- function fitting assuming arbitrary asymmetric uncertainties

example01 - GraphFitter

Simple ROOT macro:

```
  TGraphErrors * graph = ...;



  TF1 * f1 = new TF1("f1","[0]+[1]*x", 0., 100.);
  f1->SetParLimits(0, -20., 30.);
  f1->SetParLimits(1, .5, 1.5);

// BAT fitting bellow
  BCGraphFitter * gf = new BCGraphFitter(graph, f1);

  gf->Fit();



  gf->DrawFit("", true);

  gf->PrintAllMarginalized("file.ps");
```

Data to fit are in a **TGraphErrors** object (uncertainties are necessary)

Function to fit is defined via **TF1** object

Parameter ranges have to be specified !

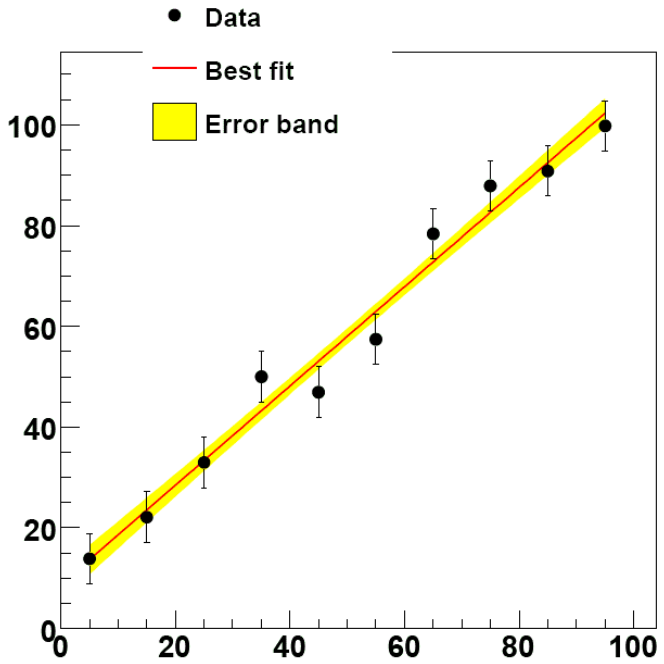
Define **BCGraphFitter**, assume gaussian uncertainties

Fit the TGraphErrors with TF1 function using BAT

Draw the data, the best fit and the error band

Print all Marginalized distributions

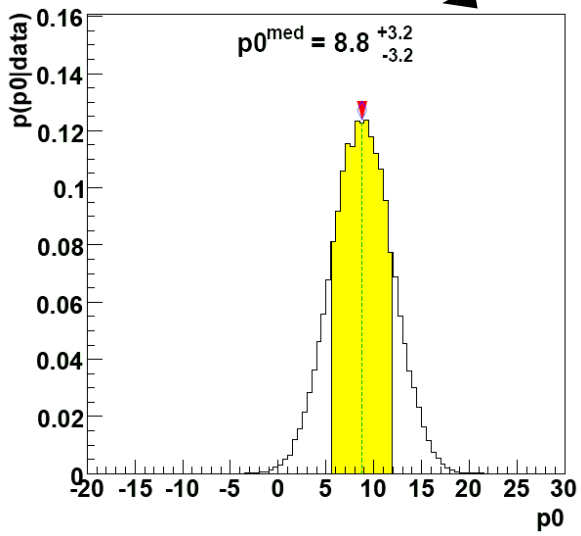
example01 - GraphFitter
Output

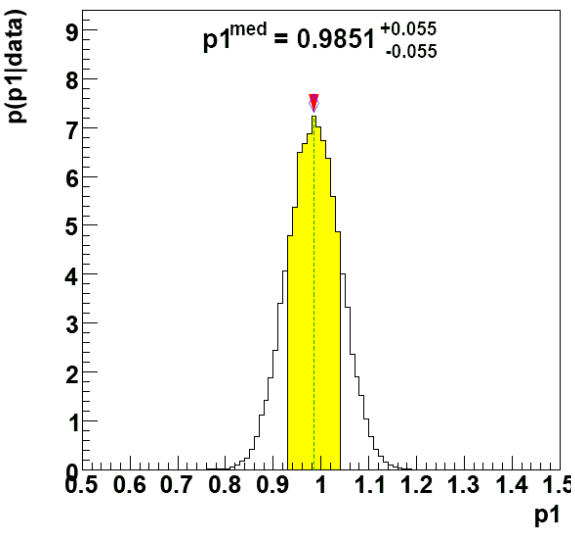


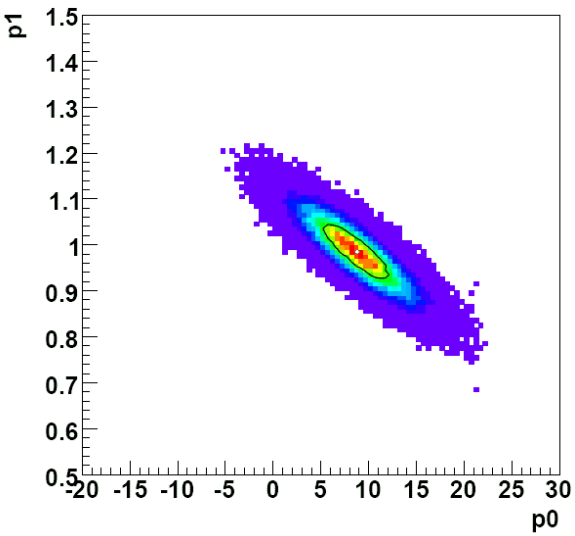
Fit and the error band representing the central 68%probability interval

Running 100000 iterations for 5 chains
- approximately 30 seconds
- ! very simple example !

Marginalized posterior probability densities







example02 - HistogramFitter

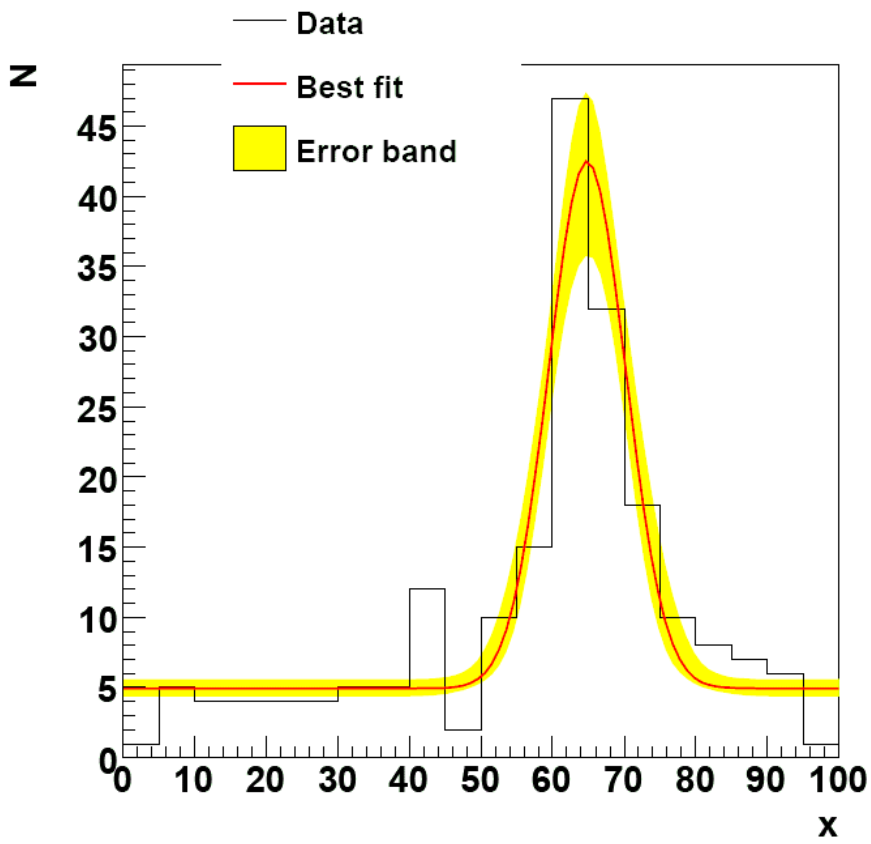
Simple ROOT macro:

- takes histogram as an input
- uses Poissonian uncertainties

```
TH1D * histo = ...;
TF1 * f1 = ...;

BCHistogramFitter * hf =
     new BCHistogramFitter(histo, f1);
hf -> Fit();
hf -> DrawFit("", true);
...
```

define data histogram

define fit function and parameter ranges

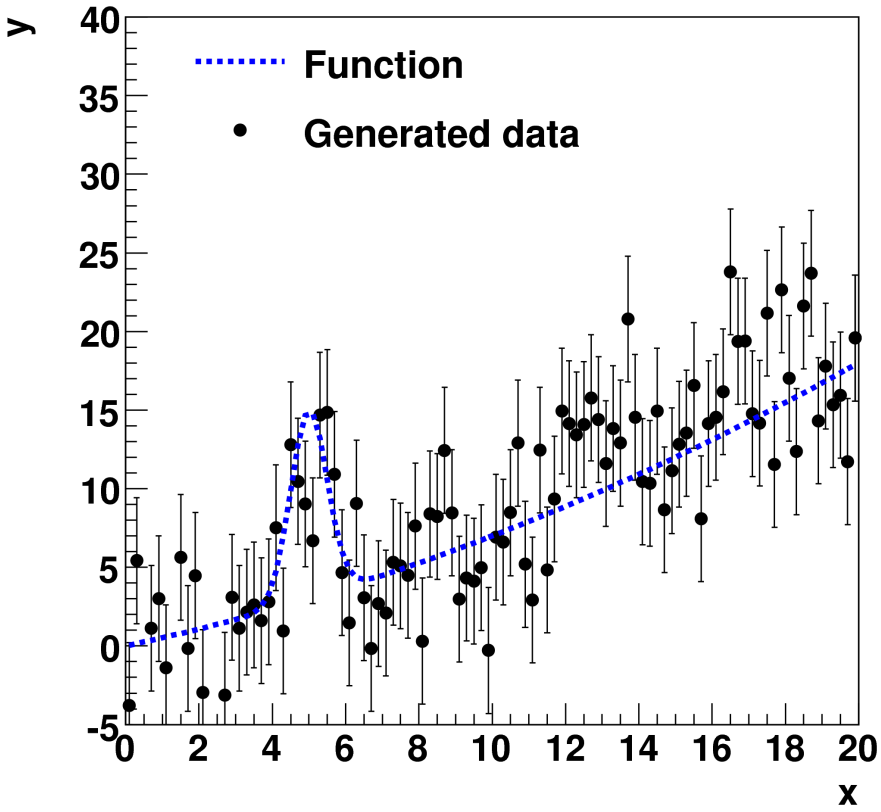


Use whenever you want to fit a histogram.

example03 - GraphFitter

The example from the BAT paper

- Fit data set using:
  - I. 2nd order polynomial (no peak)
  - II. gaussian peak + constant
  - III. gaussian peak + straight line
  - IV. gaussian peak + 2nd order pol.

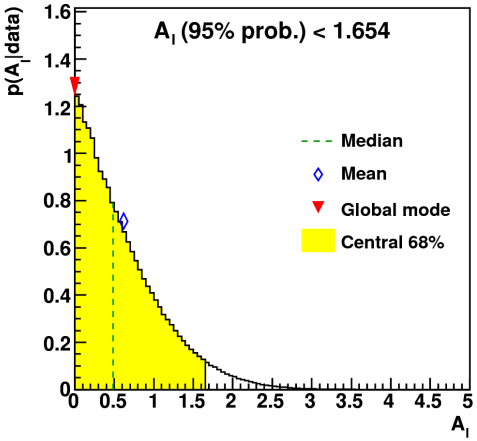


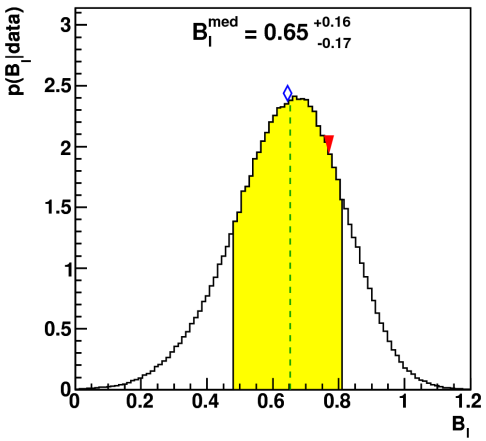
- Assume flat a priori probabilities in certain ranges of parameters, i.e. $p_0(\vec{\lambda}) = \text{const.}$
- Search for peak in range from 2. to 18. with maximum sigma of 4.
- Data were generated as gaussian peak + 2nd order polynomial

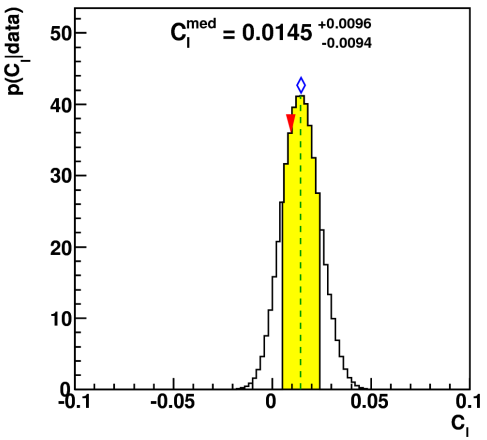
Posterior probability density

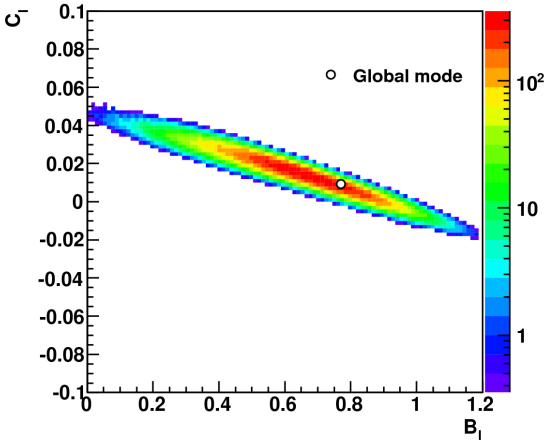
2nd order polynomial

Marginalized probability wrt. one or two parameters









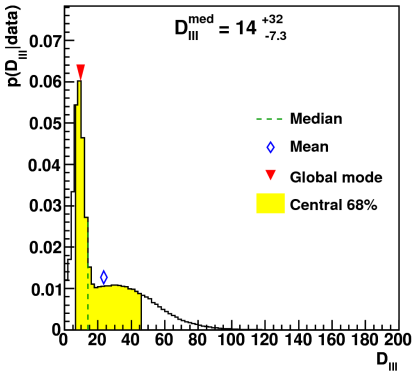
$$p(\lambda_i \mid \vec{D}) = \int p(\vec{D} \mid \vec{\lambda})\, p_0(\vec{\lambda})\, d\vec{\lambda}_{j \neq i}$$

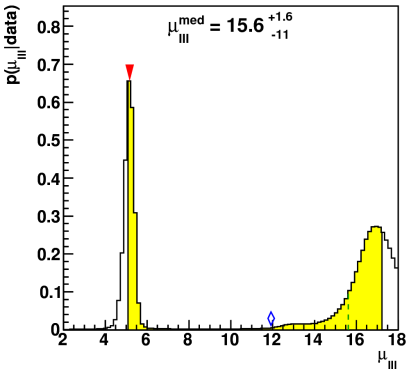
- Integrated over all other parameters
- All calculated during single MCMC run
- Stored as TH1D and TH2D
- In general, mode of the marginalized distribution not equal to global mode
- Extracted values left to the user
- Default output:
  - Mean
  - Central 68% interval
  - Limits
  - Mode
  - Median

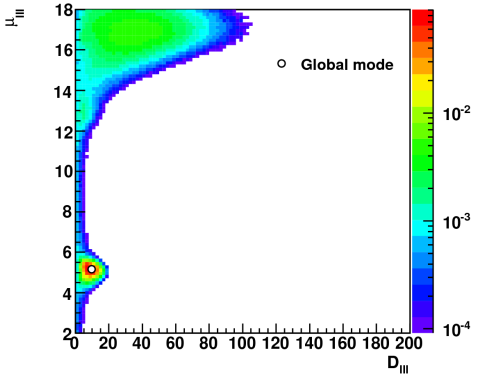
All information about the posterior PDF is in the Markov chain (stored as ROOT Tree)

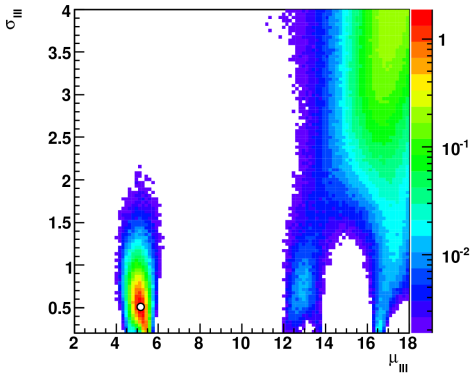
Posterior probability density

peak + straight line

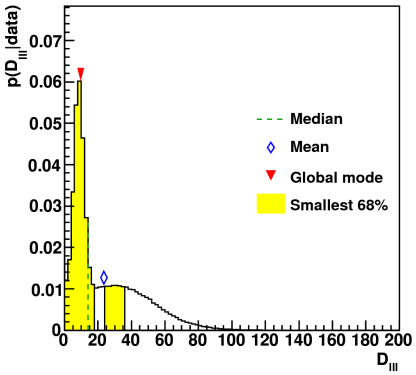


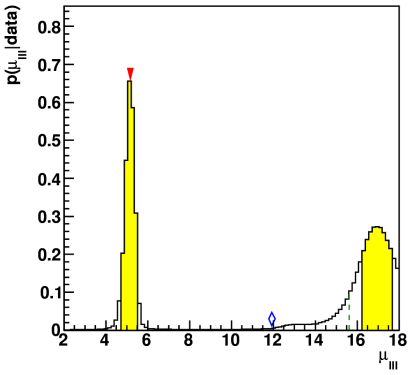






- Central interval not always optimal
- Multiple maxima in parameter space
- MCMC follows probability distributions with complicated shapes

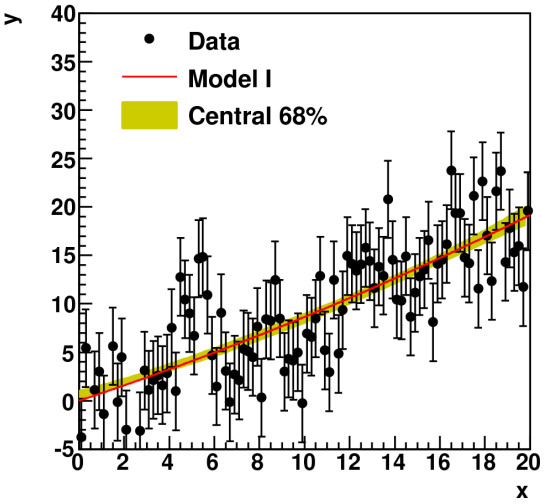


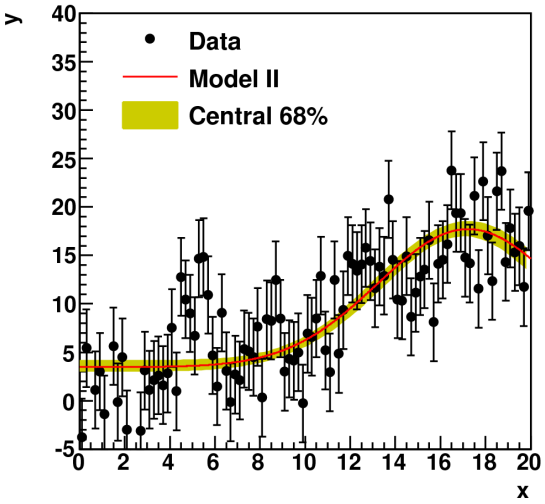


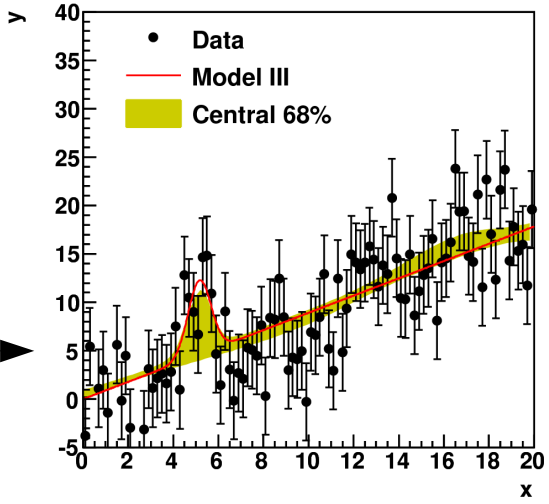
- Optionally calculate smallest interval

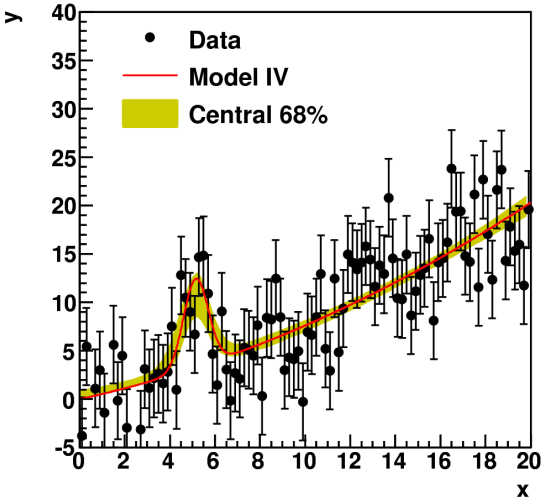
Fits + Uncertainty bands

- Uncertainty band calculated during Markov Chain run
  - calculate $f(x)$ at many different x using sampled λ
  - fill 2D histogram in (x,y)
  - after run look at distribution of y at given x and calculate central 68% interval
- Fit can lie outside of the central 68% interval
  - again, central 68% not optimal



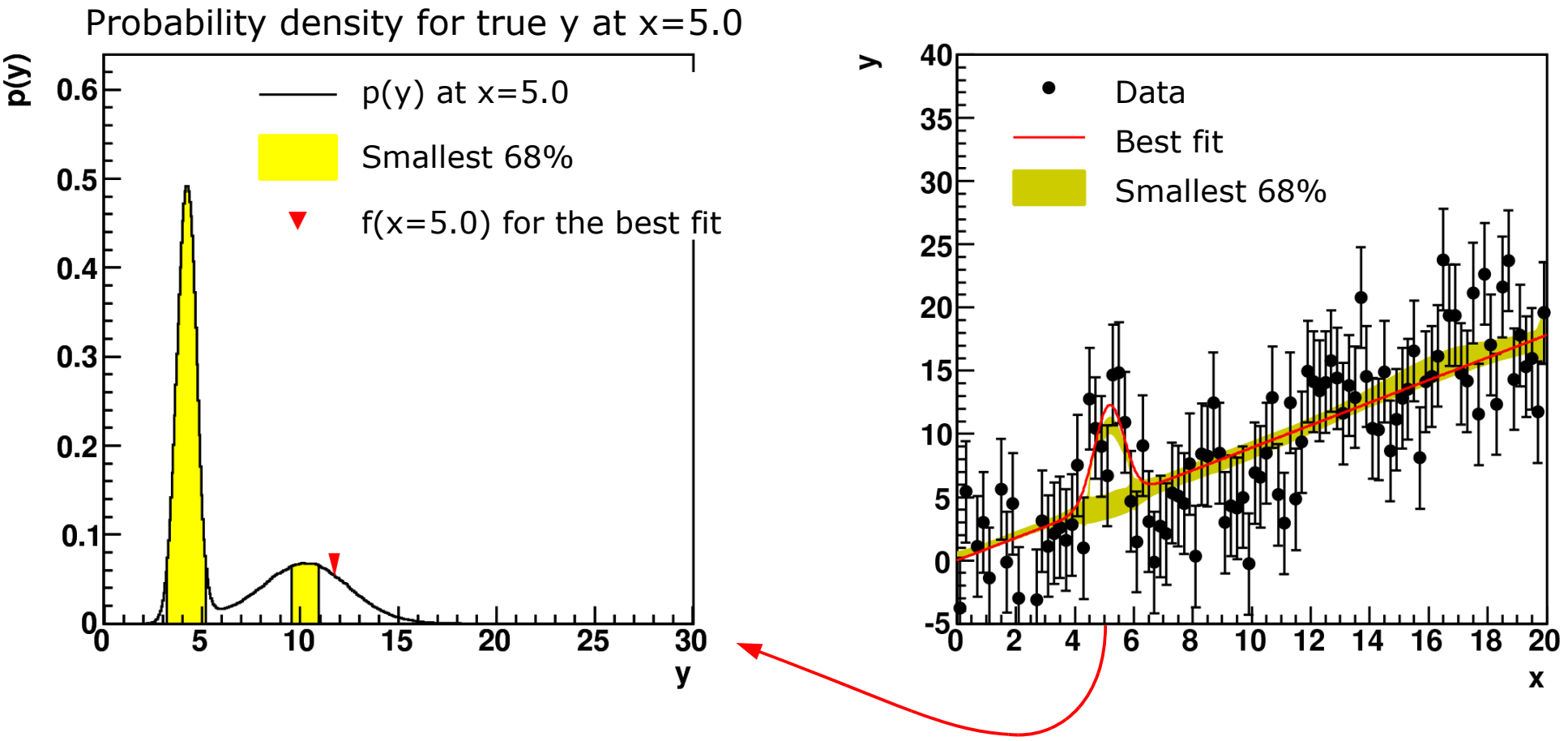






Uncertainty multi-band

- Calculating f(x) at many x for every set of parameters sampled in the Markov chain ⇔ uncertainty propagation
- Can lead to multiple uncertainty bands



example04 - EfficiencyFitter

Simple ROOT macro:

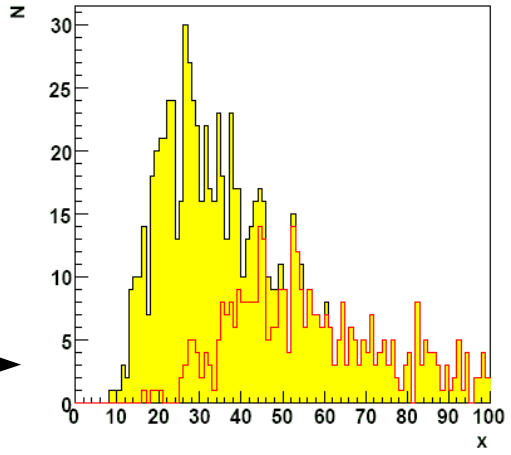
- takes two histograms as an input:
  - histogram with the whole set of events/entries
  - histogram with subset of events/entries
- uses binomial uncertainties

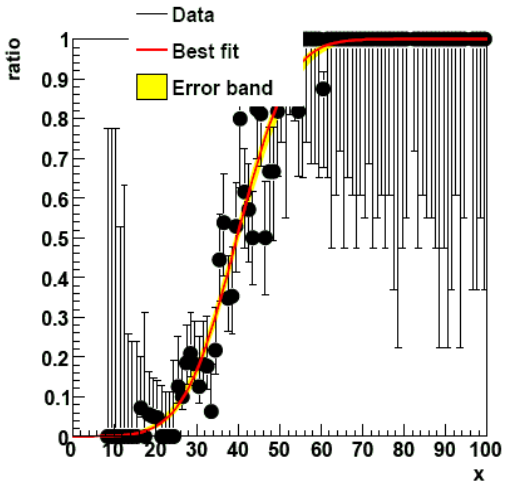
```
TH1D * hfull = ...;
TH1D * hsub = ...;
TF1 * f1 = ...;

BCEfficiencyFitter * ef = new BCEfficiencyFitter(hfull, hsub, f1);
ef -> Fit();
ef -> DrawFit("", true);
...
```

Use whenever you want to fit an efficiency.

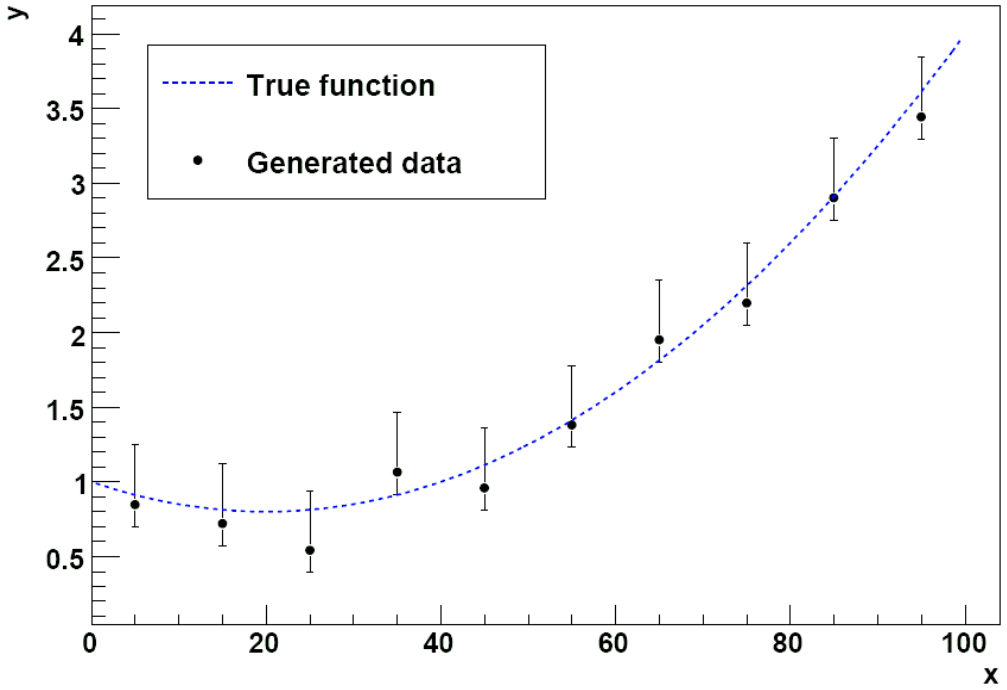
Typical example:

→ fitting the trigger efficiency

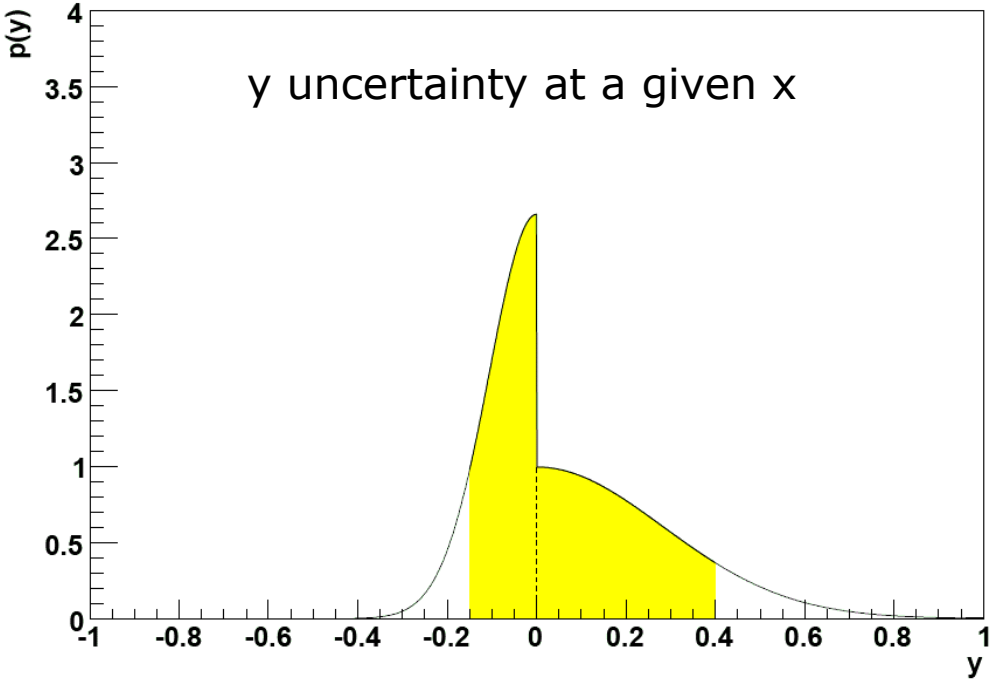




Example with asymmetric errors



- Data generated according to 2nd order polynomial
- Fit using straight line and 2nd order polynomial
- assuming 2 half gaussians for the description of the asymmetric uncertainty



Example code: Model definition

USER MODEL EXAMPLE – **2nd order polynomial** (model class)

```
void ModelPol2::DefineParameters() {    // define parameters of the model
      this -> AddParameter("A",  0.,   5.);  // index 0
      this -> AddParameter("B",  0.,   1.2); // index 1
      this -> AddParameter("C", -0.1., 0.1); // index 2
}                                           // fit function is  f(x) = A + Bx + Cx^2

double ModelPol2::LogLikelihood(vector <double> params) { // define likelihood
      double lprob = 0.;
      double A = params[0], B = params[1], C = params[2];
      for(int i=0; i<this -> GetNDataPoints(); i++) {  // loop over all data points
            BCDataPoint * data = this -> GetDataPoint(i);
            double        x = data -> GetValue(0);
            double        y = data -> GetValue(1);
            double yerrdown = data -> GetValue(2);  // asymmetric errors on all points
            double   yerrup = data -> GetValue(3);

            double yexp     = A + x*B + x*x*C;  // calculate expectation value

            double yerr     = (y<yexp) ? yerrdown : yerrup; // decide which uncertainty is applicable

            lprob += BCMath::LogGaus(y, yexp, yerr, true);
      }
      return lprob;
}

double ModelPol2::LogAPrioriProbability(vector <double> params) { // define prior
      return 0.;  // flat prior probability for all parameters in their range
}
```

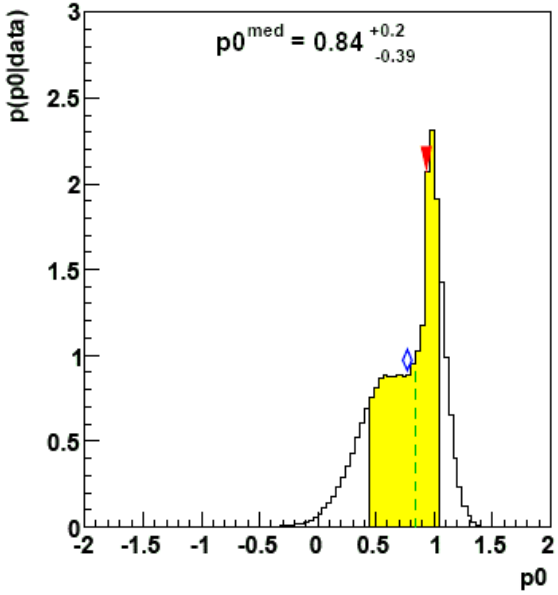
Example code: Main program

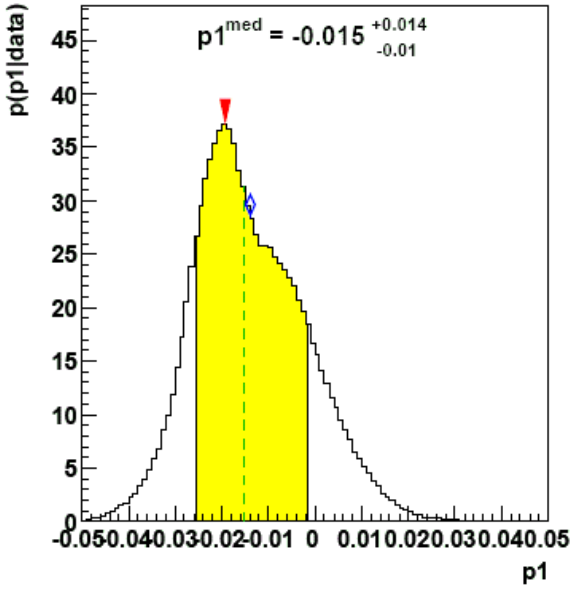
USER MODEL EXAMPLE – **2nd order polynomial** (simple main program)

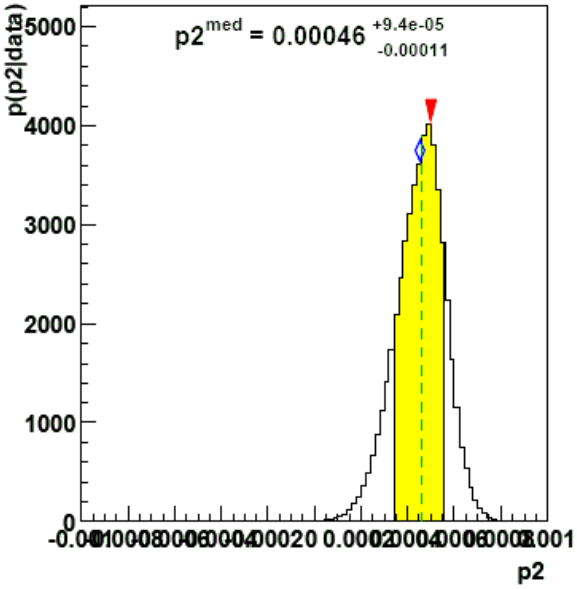
```
int main()
{
     ModelPol2 * mymodel = new ModelPol2("2Dpol");  // create model object

     DataSet * mydata = new DataSet("measurement1");  // create data object
     mydata -> ReadDataFromFileTxT("measurement1.dat",3); // read in data, 3 columns: x,y,yerr
     mymodel -> SetDataSet(mydata);  // assign data to model

     mymodel -> Normalize();  // integrate to get the normalization

     mymodel -> MarginalizeAll();  // marginalization
     mymodel -> PrintAllMarginalized("mymodel_all.ps");

     mymodel -> CalculatePValue(mymodel -> GetBestFitParameters());

     BCModelOutput * myout = new BCModelOutput(mymodel,"mymodel.root");
     myout -> WriteMarginalizedDistributions();
     myout -> Close();

     mymodel -> PrintResults();

     // add more things to do

     return 0;
}
```

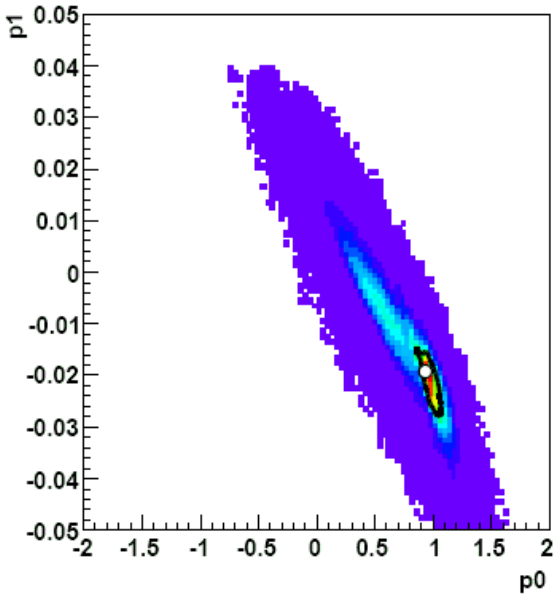
Example with asymmetric errors

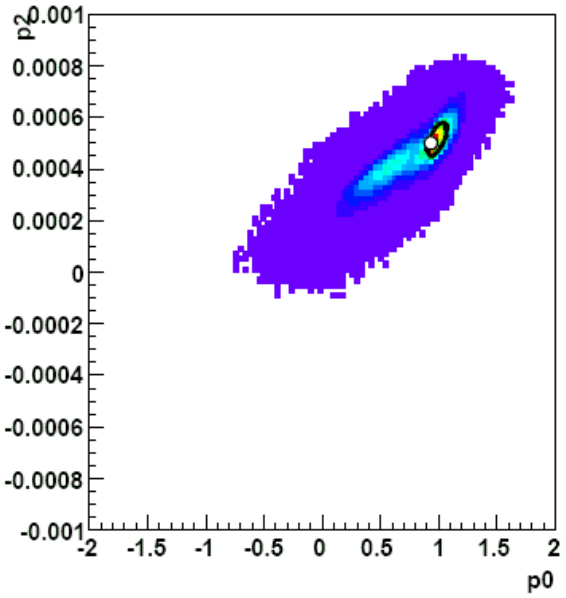
Marginalized distributions

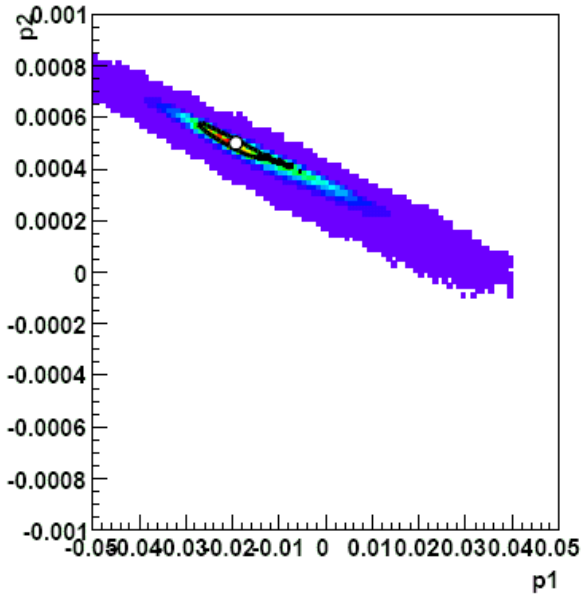






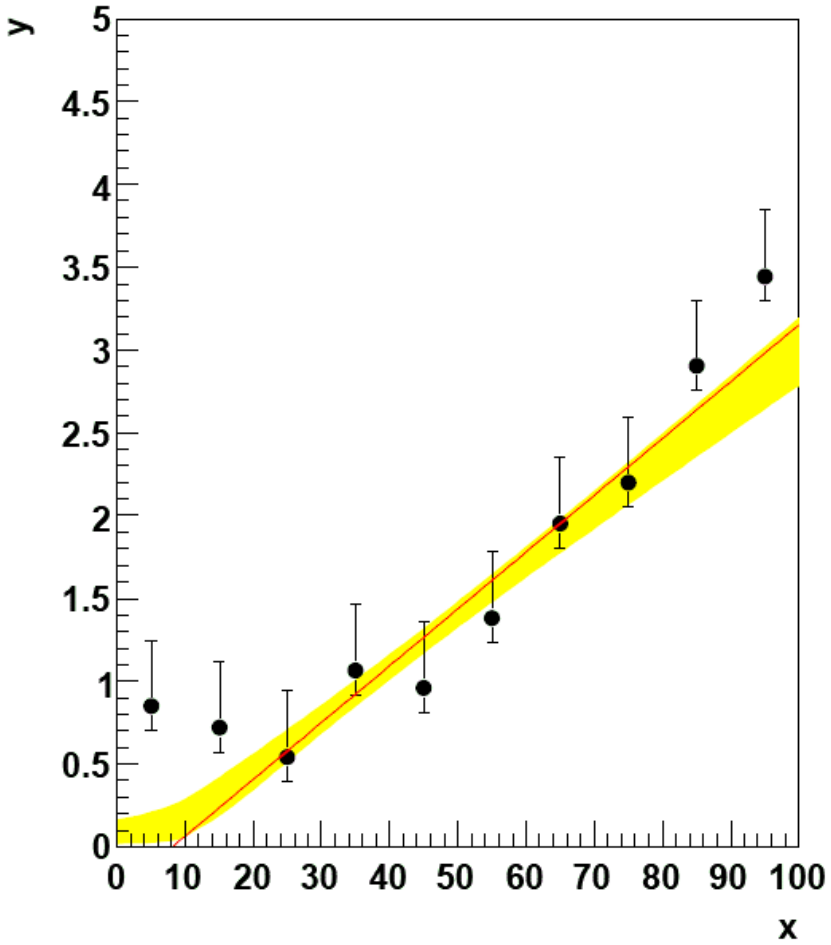


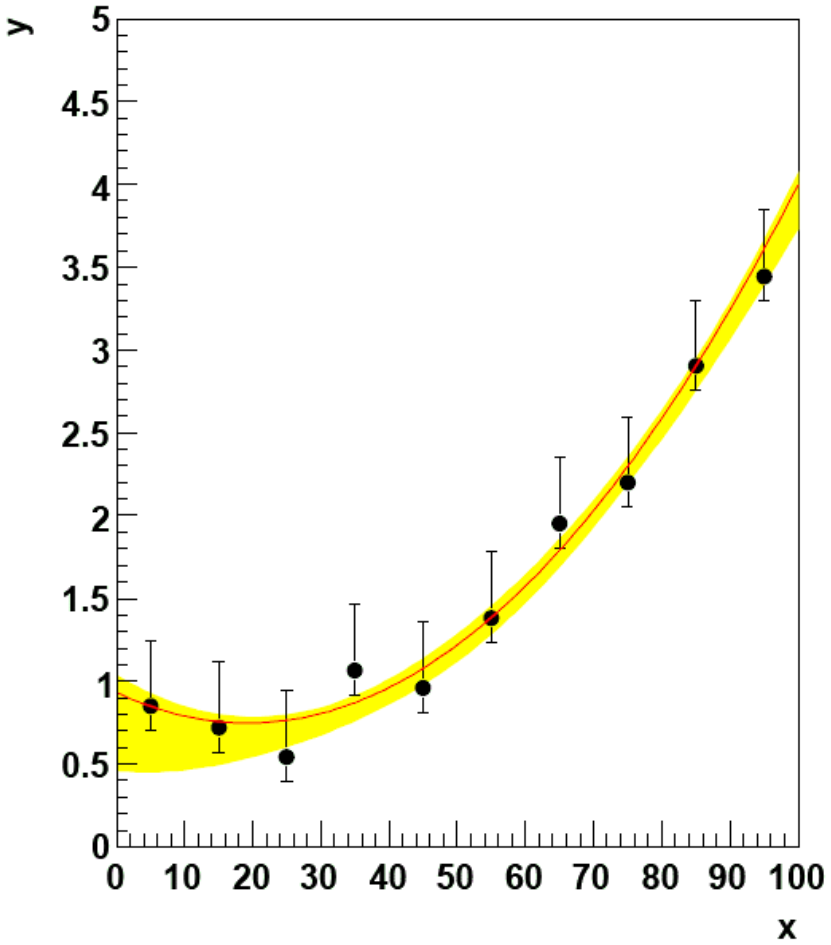




Example with asymmetric errors

Fit + error band





Days 2 & 3

- Day 2
  - limit setting
  - adding systematic uncertainties
  - non-flat priors

- Day 3
  - hypothesis testing
  - model comparison
  - small signal on top of large background

- tutorials for can be followed on the web
  - http://www.mppmu.mpg.de/bat

- let us know your comments, suggestions, problems
  - bat@mppmu.mpg.de