GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

UNIVERSITÄT SIEGEN

The Bayesian Analysis Toolkit

a C++ tool for Bayesian inference

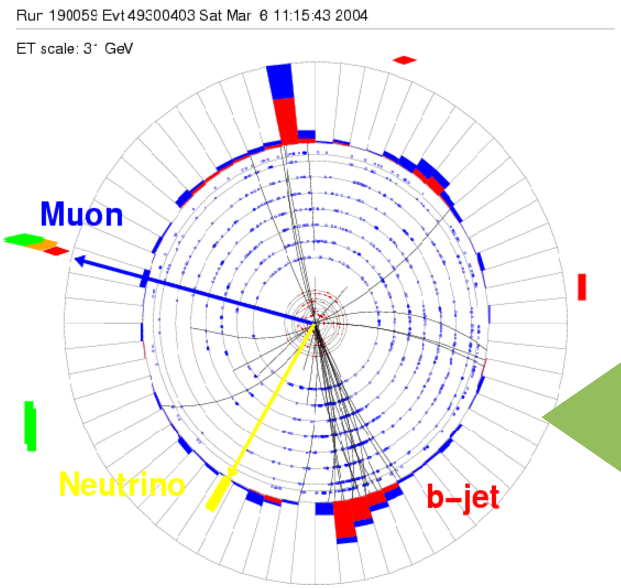
Kevin Kröninger – University of Göttingen / University of Siegen

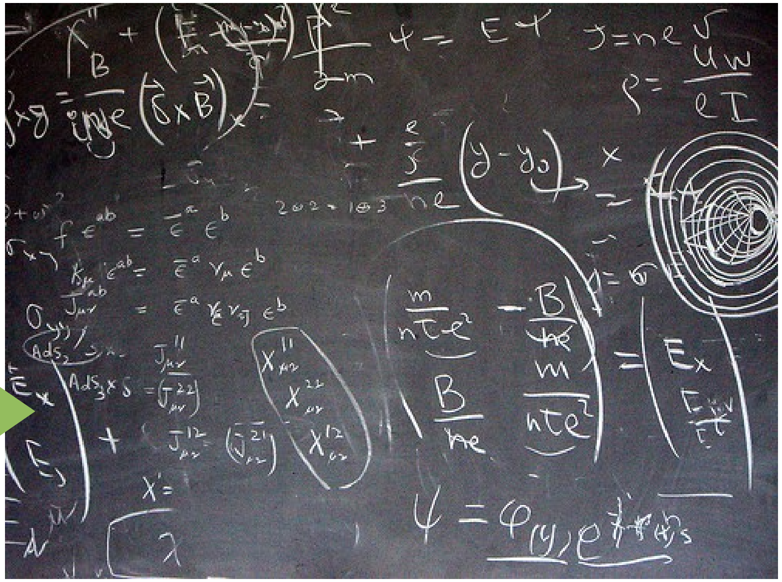
Bayes Forum, Munich, 13.04.2012

The BAT (wo)men: Frederik Beaujean, Allen Caldwell, Daniel Greenwald, Daniel Kollar, Kevin Kröninger, Shabnaz Pashapour, Arnulf Quadt

BAT

BAYESIAN ANALYSIS TOOLKIT





Experiment Data analysis Theory

Questions in data analysis:

- What does the data tell us about our model? Parameter estimation
- Which model is favored by the data? Model comparison
- Is the model compatible with the data? Goodness-of-fit test

Need methods and tools to extract information

Outline:

- Requirements / Implementation / Tools
- Markov Chain Monte Carlo
- MCMC implementation in BAT
- A working example
- Some propaganda
- Summary

Requirements and solutions

Requirements:

- Allow to phrase arbitrary models and data sets
- Interface to (HEP) software
- Estimate parameters (point estimates)
- Find probability densities (interval estimates)
- Propagate uncertainties
- Compare models
- Test validity of model against the data

Solutions:

- C++ library based on ROOT*.
- Models are implemented as (base) classes and need to be defined by the user, or
- A set of pre-defined models can be used.
- A set of algorithms can used to perform the actual analysis

*Framework for handling large data sets, graphical representation and analysis tools

Requirements and solutions

Requirements:

- Allow to phrase arbitrary models and data sets
- Interface to HEP software
- Estimate parameters (point estimates)
- Find probability densities (interval estimates)
- Propagate uncertainties
- Compare models
- Test validity of model against the data

Solutions:

- Minimization can be done via a Minuit interface or via Simulated Annealing.
- Marginalization and uncertainty estimation can be done via Markov Chain Monte Carlo (MCMC).
- Propagation of uncertainties (without Gaussian assumptions) can also be done via MCMC

Requirements and solutions

Requirements:

- Allow to phrase arbitrary models and data sets
- Interface to HEP software
- Estimate parameters (point estimates)
- Find probability densities (interval estimates)
- Propagate uncertainties
- Compare models
- Test validity of model against the data

Solutions:

- Direct comparison of model probabilities (Bayes factors)
- Integration methods from Cuba* library linked
- Perform *p*-value tests

*A collection of numerical integration methods e.g., VEGAS

Implementation

USER DEFINED

- create model
- read-in data

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 (anything)
- interface to user-defined software

COMMON METHODS

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

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

Implementation

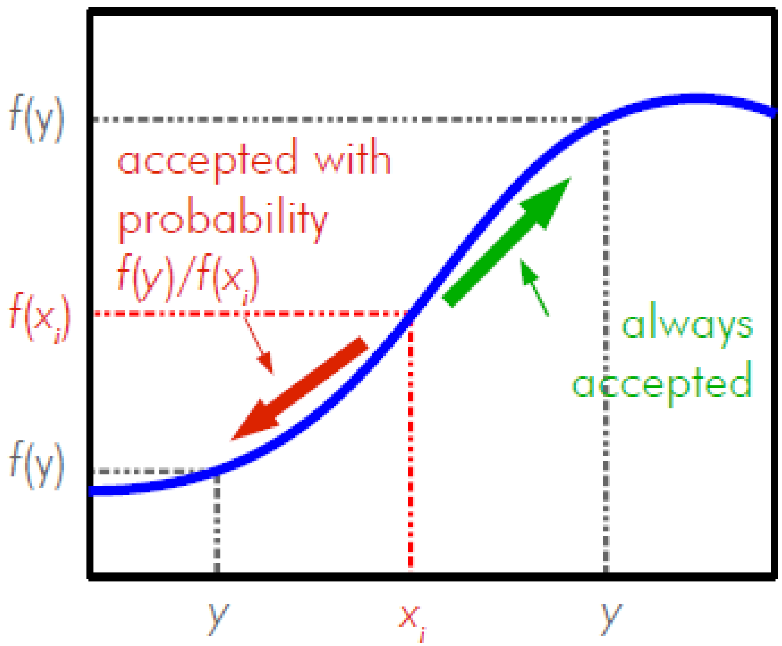
Tools:

- Point estimates:
  - Minuit
  - Simulated Annealing
  - MCMC
  - simple Monte Carlo
- Marginalization:
  - MCMC
  - simple Monte Carlo
- Integration:
  - sampled mean
  - importance sampling
  - CUBA (Vega, Suave, Divonne, Cuhre)
- Sampling:
  - simple Monte Carlo
  - MCMC
- Error propagation
  - MCMC

How does MCMC work?

- Output of Bayesian analyses are posterior probability densities, i.e., functions of an arbitrary number of parameters (dimensions).
- Sampling large dimensional functions is difficult.
- Idea: use random walk heading towards region of larger values (probabilities)
- Metropolis algorithm

N. Metropolis *et al.*,
J. Chem. Phys. 21 (1953) 1087.



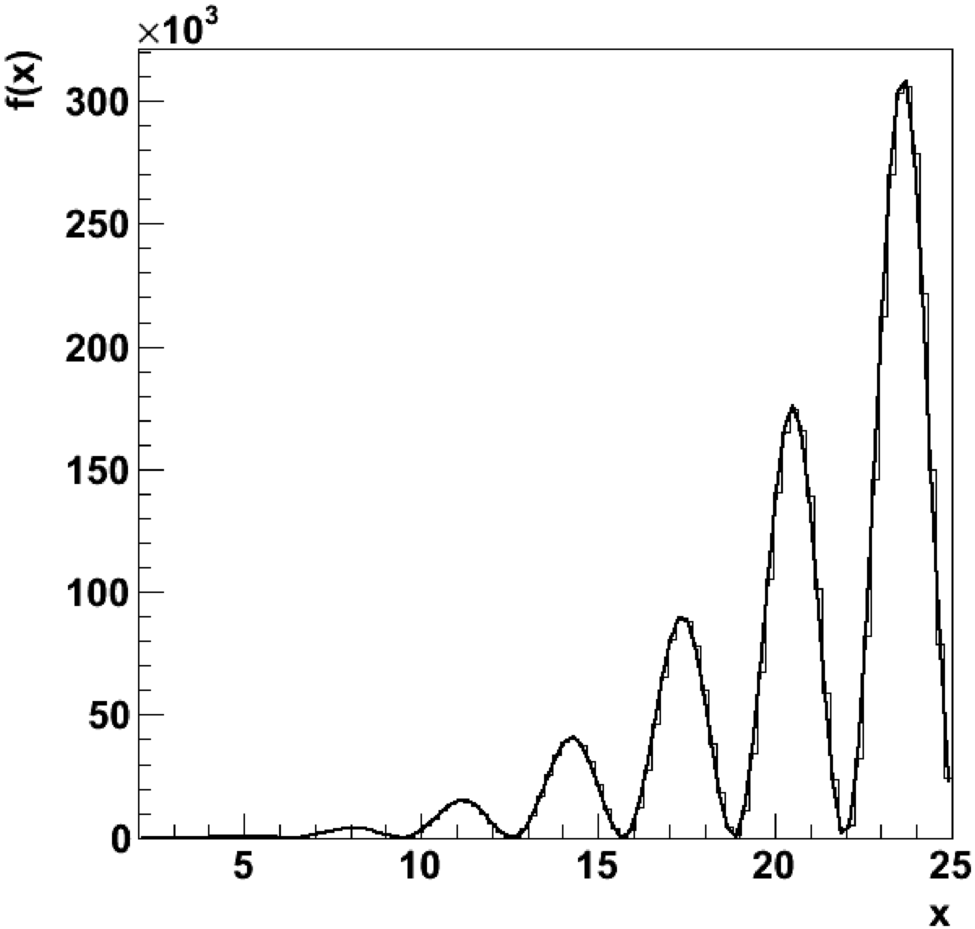
- Start at some randomly chosen x_i
- Randomly generate y around x_i
- If $f(y) > f(x_i)$ set $x_{i+1} = y$
- If $f(y) < f(x_i)$ set $x_{i+1} = y$ with prob. $p=f(y)/f(x_i)$
- If y is not accepted set $x_{i+1} = x_i$
- Start over

Does it work for difficult functions?

- Test MCMC on a function:

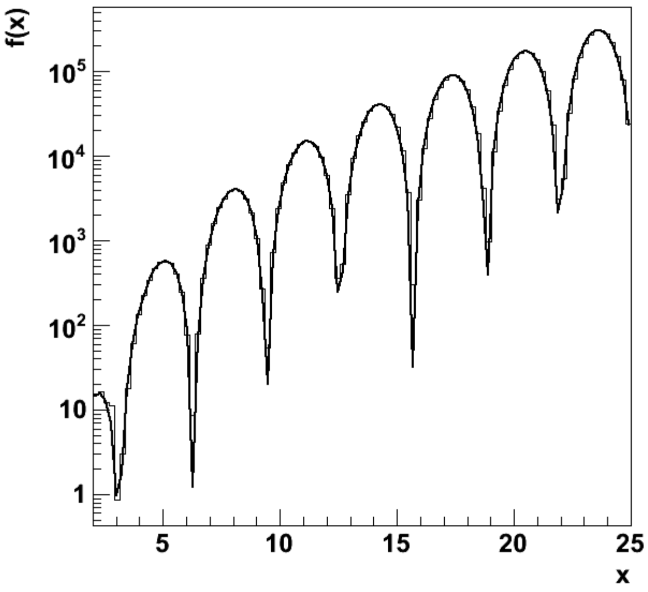
$$f(x) = x^4 \sin(x^2)$$

- Compare MCMC distribution to analytic function
- Several minima/maxima are no problem.
- Different orders of magnitude are no problem.

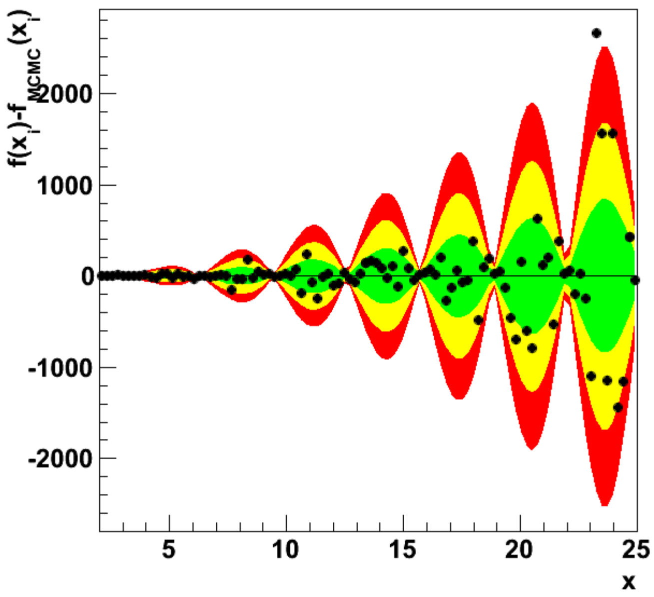


An example of MCMC

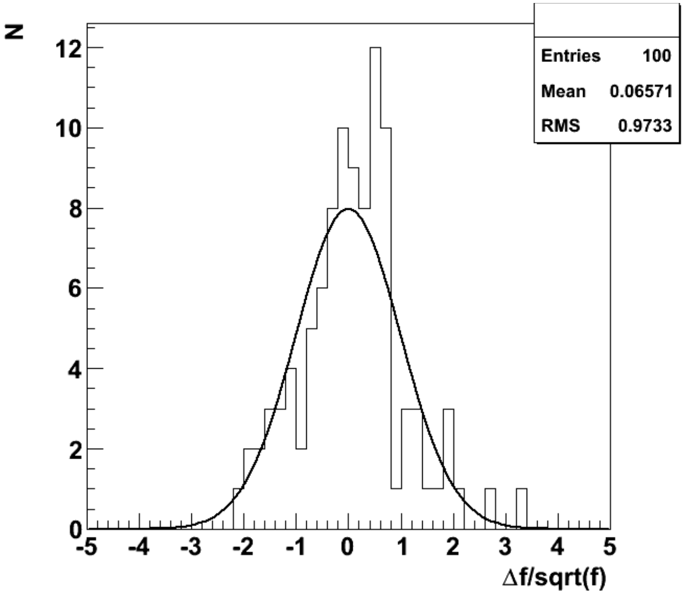
f(x) vs. x



χ^2 vs. x



Pull distribution



For more examples, see our test suite on the BAT web page.

How does MCMC help in Bayesian inference?

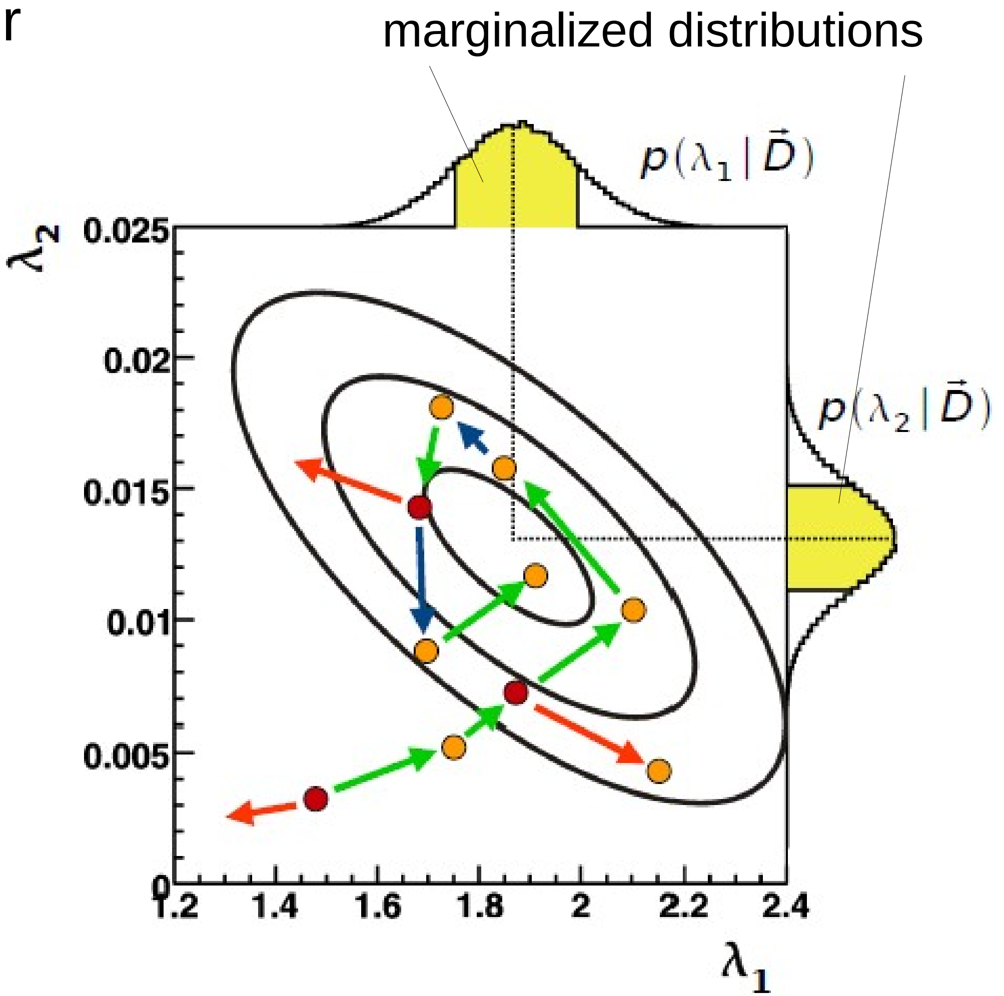
- Use MCMC to sample the posterior probability, i.e.

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

- Marginalization of posterior:

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

- Fill a histogram with just one coordinate while sampling
- Error propagation: calculate any function of the parameters while sampling
- Point estimate: find mode while sampling



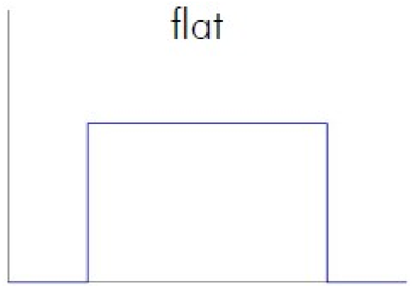
Metropolis is ~3 lines of code, fairly easy, but ...

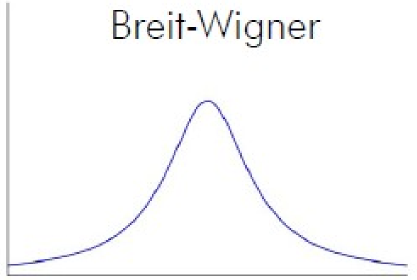
Technical details:

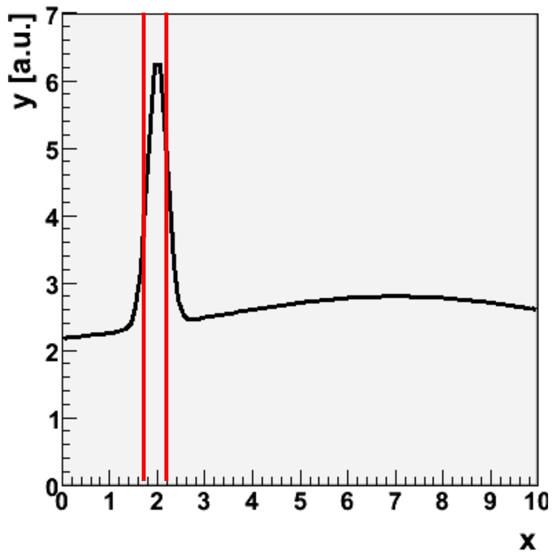
- How are the new points generated? Proposal function
- How many points can we afford to throw away? Efficiency
- How many iterations do we need? Convergence criterion
- How correlated are the points? Auto-correlation/lag

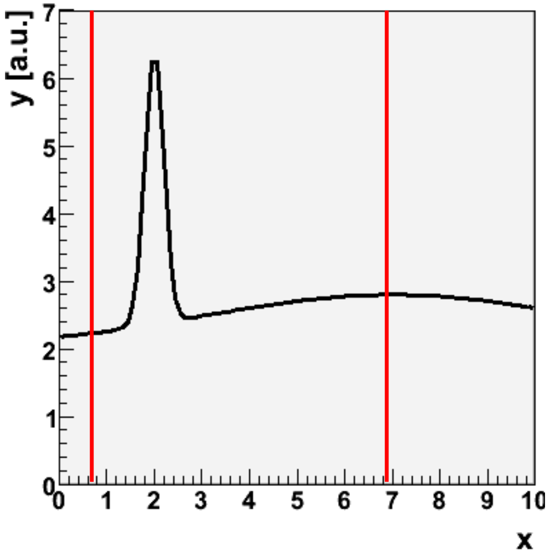
How are the new points generated?

- Proposal function: probability density of the step size used in the random walk
- Should be independent of the underlying distribution, i.e., the same everywhere
- Shape is important (default: Breit-Wigner)
- Width defines efficiency = fraction of accepted points









- Small width = large efficiency
- Large width = small efficiency
- Trade off: efficiency ~25%

Convergence criterion

How many iterations do we need?

- MCMC distribution should converge to underlying function.
- In practice: need to stop the chain at some point. Need criteria.
- Two strategies:
- Single chain convergence:
  - Could monitor auto-correlation
  - Very CPU-time intensive
  - Could be done offline
- Multi-chain convergence:
  - Test convergence of multiple chains wrt each other
  - Use Gelman&Rubin criterion

Gelman & Rubin convergence:

- Calculate average variance of all chains

$$W = \frac{1}{m}\frac{1}{n-1}\sum_{j=1}^{m}\sum_{i=1}^{n}(x_i - \bar{x}_j)^2$$

- Estimate variance of target distribution

$$\hat{V} = (1-\frac{1}{n})W + \frac{1}{m-1}\sum_{j=1}^{m}(\bar{x}_j - \bar{x})^2$$

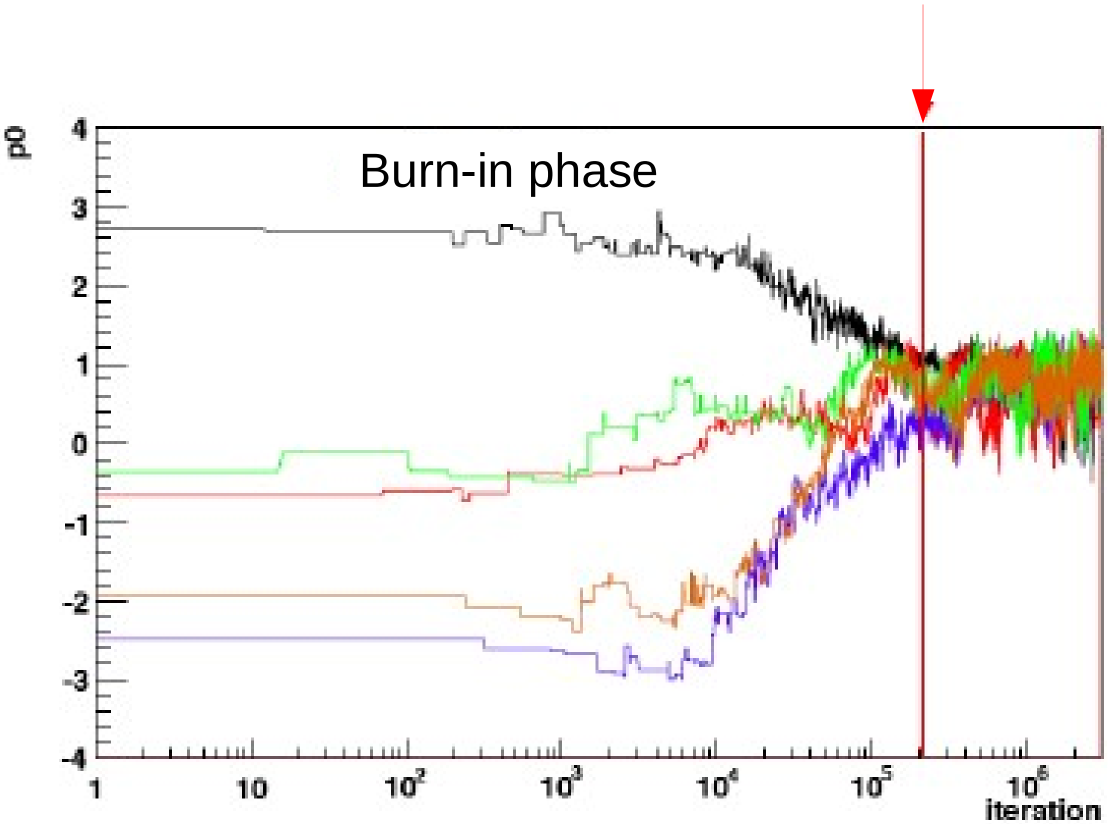
- Calculate ratio and compare with stopping criterion (relaxed version):

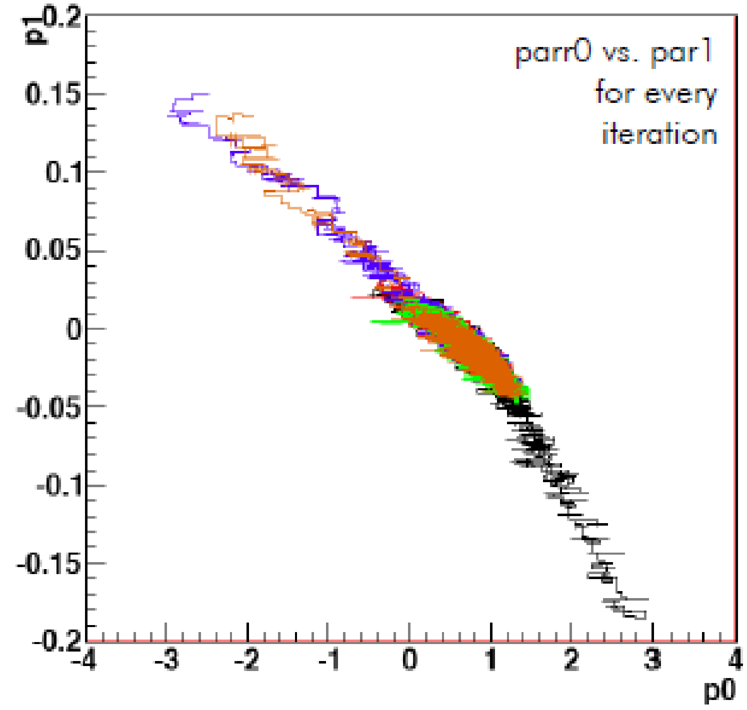
$$r = \sqrt{\frac{\hat{V}}{W}}$$ < 1.x (x = 0.1 default)

Gelman & Rubin, StatSci 7, 1992

Convergence criterion

Convergence a la Gelman & Rubin





Parameter 0 value vs. iteration

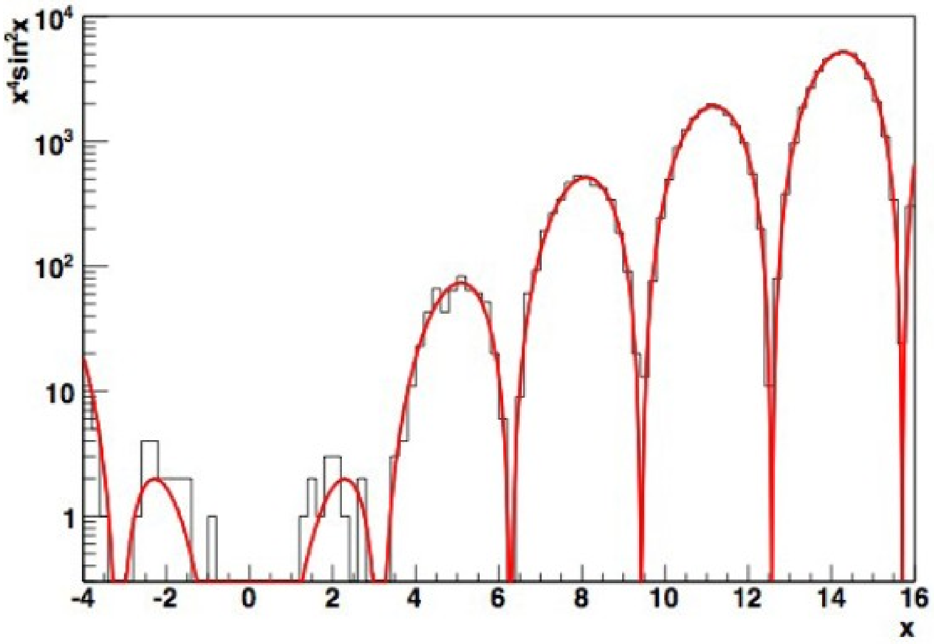
Parameter 1 vs parameter 0

Auto-correlation and lag

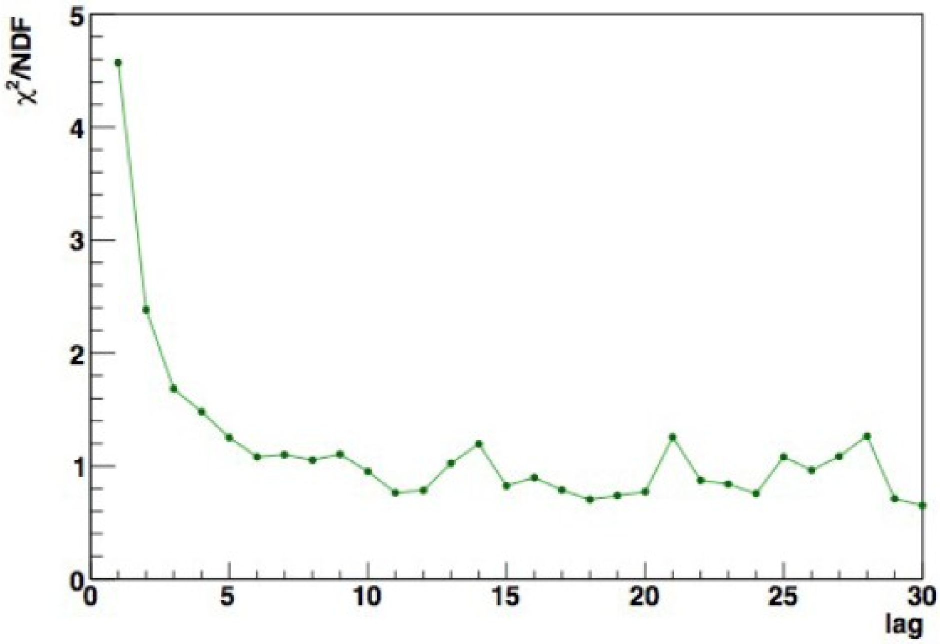
How correlated are the points?

- Simple Monte Carlo sampling and “unbiased” random walk create sets of points without (auto-correlation) while MCMC algorithm can cause auto-correlation, e.g., when rejecting a point (since the old one is taken again)
- Size of the correlation depends on the underlying posterior and the proposal function
- Can thin the MCMC sample by introducing a lag, i.e., take only every n^{th} point to calculate the marginalized distributions
- Cost: need to run a factor of n longer to get the same stat. precision

Auto-correlation and lag



f(x) vs. x

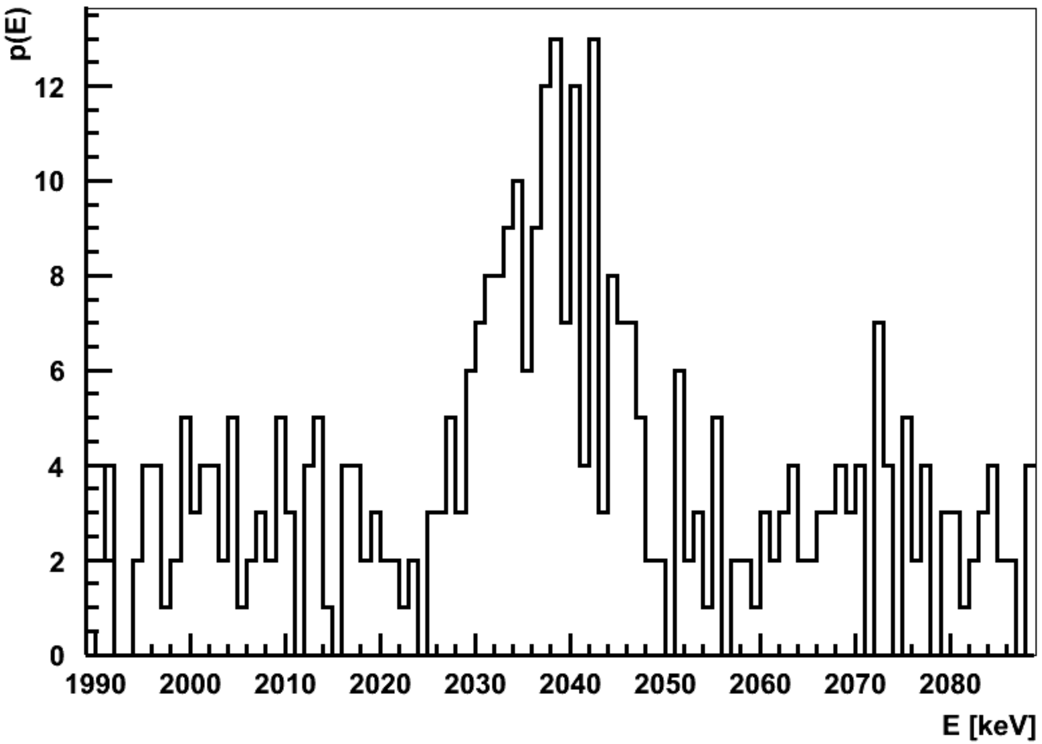


χ^2 vs. lag

A working example

Phrasing the problem:

- Estimate signal strength of Gaussian signal on top of flat background
- Data generated with the following settings:
  - Gaussian signal:
    - position μ = 2039 keV
    - width σ = 5 keV
    - strength <S> = 100
  - Flat background:
    - strength <B> = 3/keV
- Number of events per bin fluctuate with Poisson distribution



A working example

Statistical modeling:

- Statistical model:
  - Gaussian signal on top of flat background
  - 4 fit parameters:
    - Gaussian signal (3)
    - Flat background (1)
- Prior knowledge:
  - Background: 300 ± 20 in 100 keV (e.g., from sideband analysis)
  - Signal strength: exponentially decreasing (e.g., theoretical intuition)
  - Signal position: flat (e.g., no idea about the mass of a resonance)
  - Signal width: 5 ± 1 keV (detector resolution)
  - Signal and background efficiency fixed to 1 (in this example)

Statistical modeling:

- Likelihood:
  - Binned data
  - Number of expected events per bin:

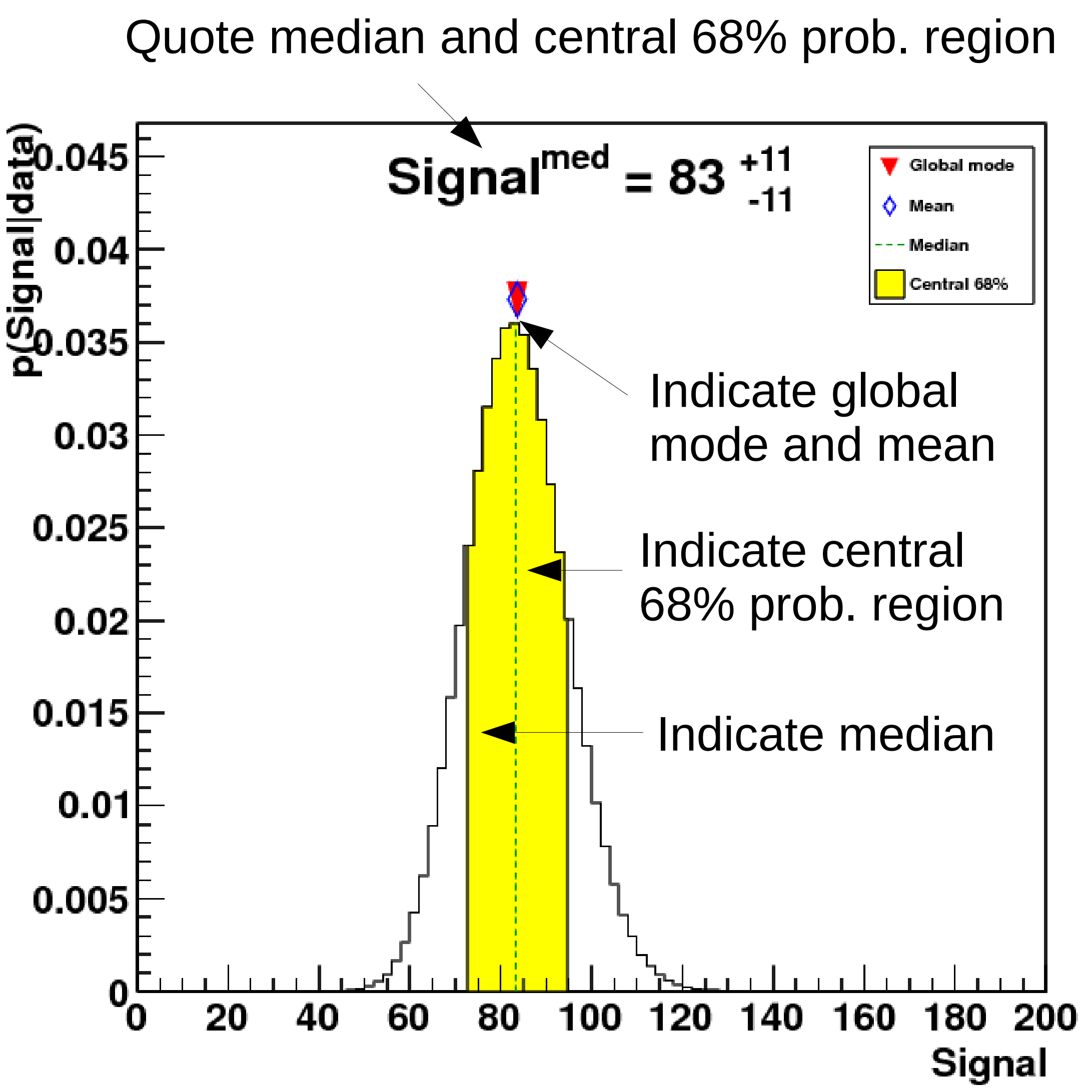
$$\lambda_i = \int_{\Delta x_i} \frac{S}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \mathrm{d}x + \frac{0.01 \cdot B}{\Delta x_i}$$

  - Assume independent Poisson fluctuations in each bin
  - Likelihood:

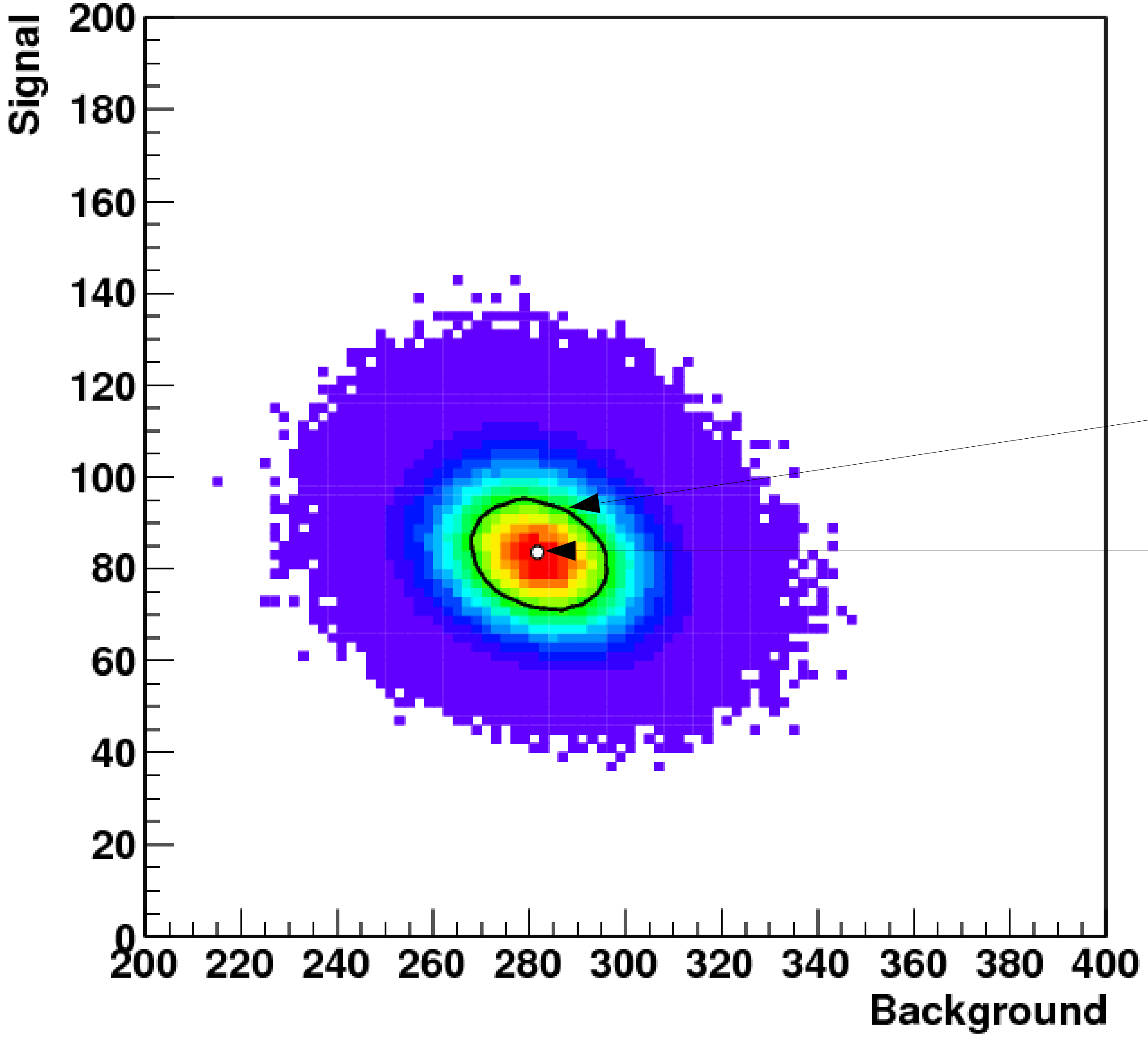
$$p(D|S,\mu,\sigma,B) = \prod_{i=1}^{N_{bins}} \frac{\lambda_i^{n_i}}{n_i}! e^{-\lambda_i}$$

Marginalized distributions:

- Project posterior onto one parameter axis
- Global mode and mode of marginalized distributions do not have to coincide
- Full (correlated) information in Markov Chain
- Default output:
  - Mean ± std. deviation
  - Median and central int.
  - Mode and smallest int.
- All 1-D and 2-D distributions are written out during main run



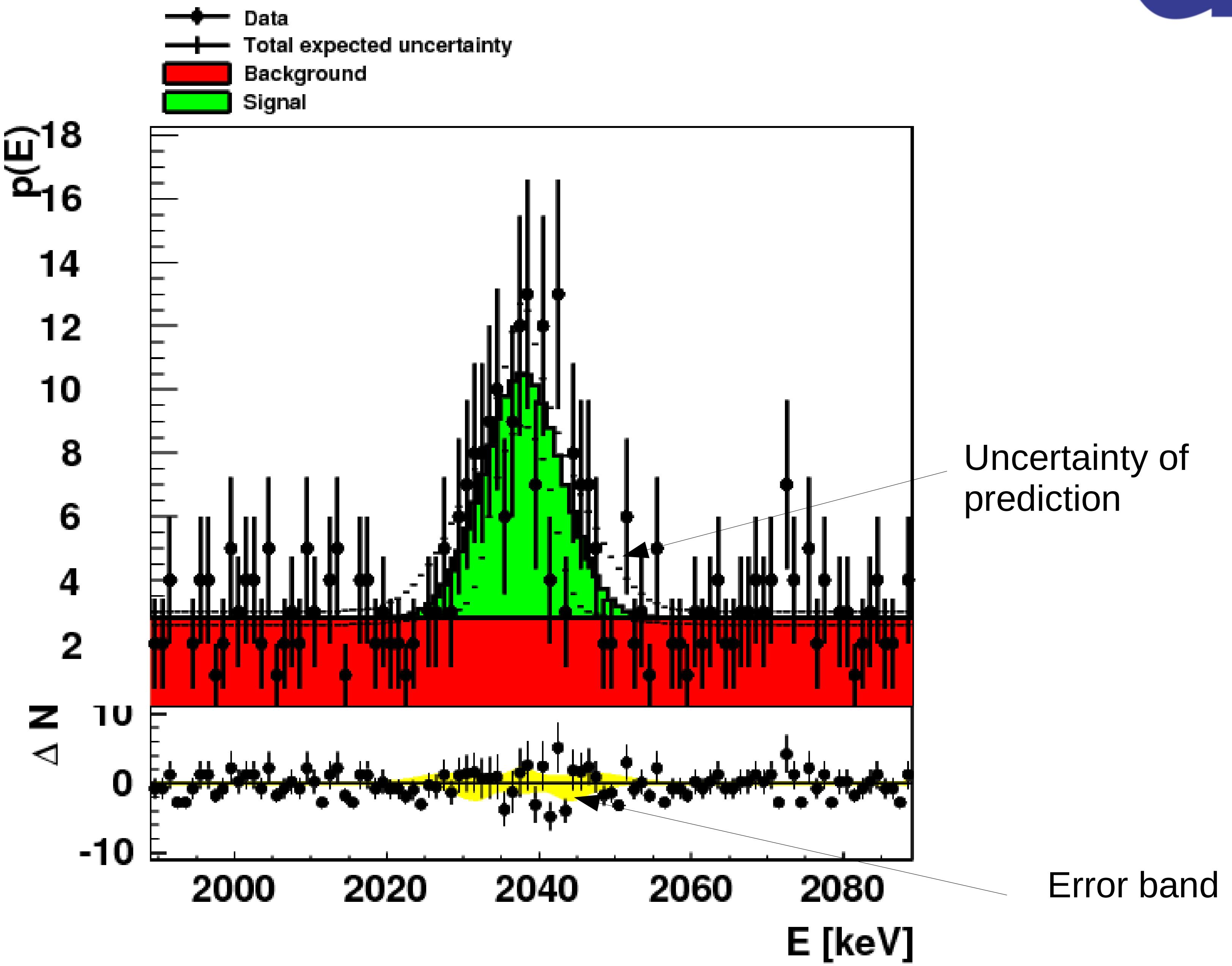
An example: 2D distributions



Indicate smallest interval containing 68% probability

Indicate global mode

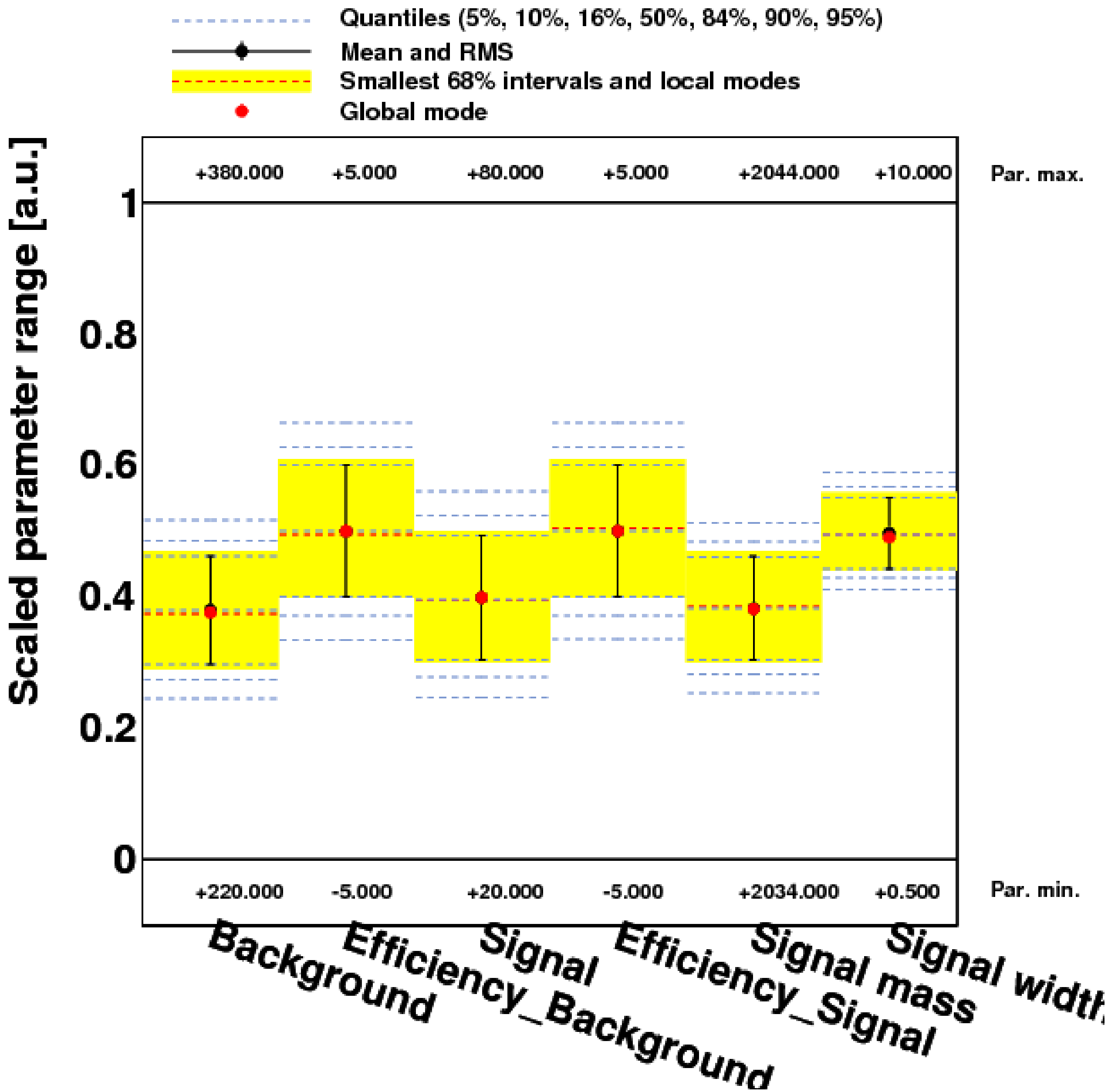
An example: the fit result



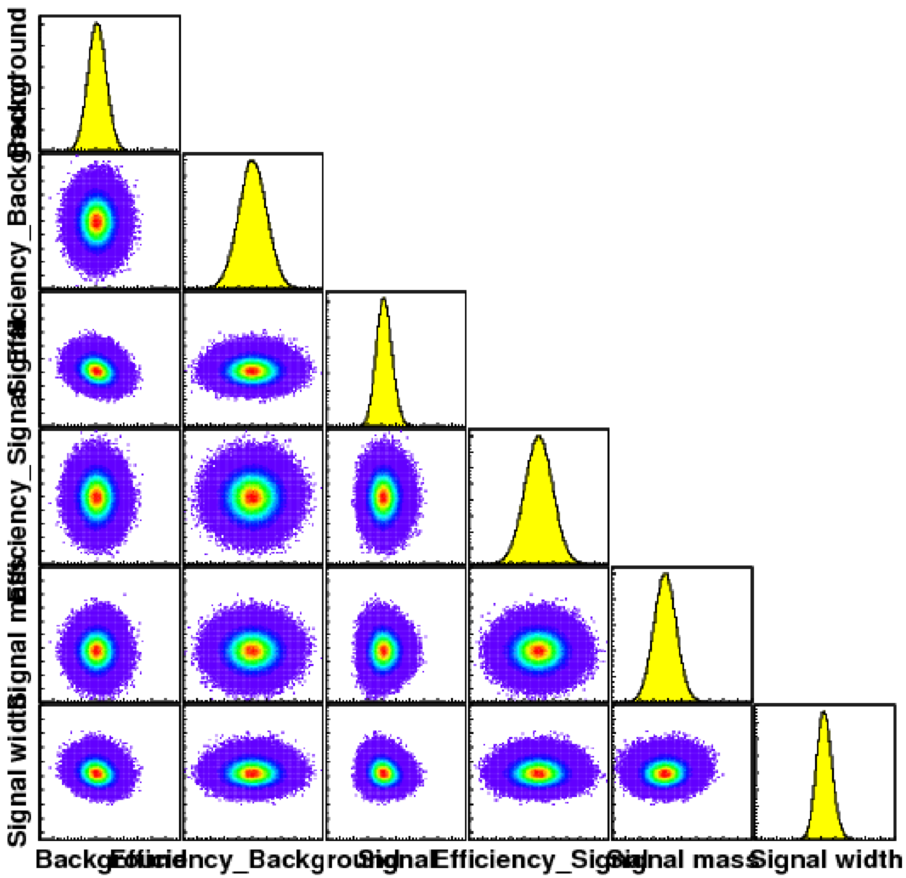
An example: the summary

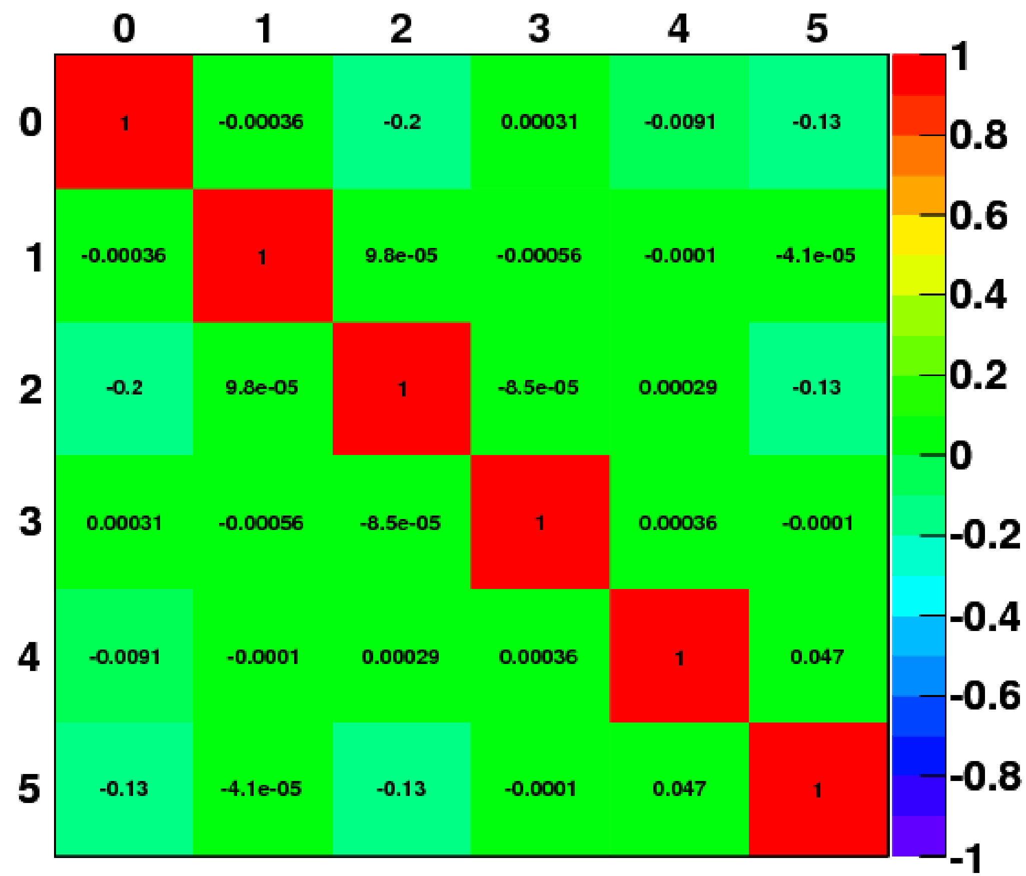

```
Results of the marginalization
==============================
List of parameters and properties of the marginalized
distributions:
  (0) Parameter "Background":
     Mean +- sqrt(V):              282.2 +- 13.04
     Median +- central 68% interval: 282.1 +  13.11 - 12.92
     (Marginalized) mode:          281
      5% quantile:                 260.8
     10% quantile:                 265.5
     16% quantile:                 269.2
     84% quantile:                 295.7
     90% quantile:                 299
     95% quantile:                 303.8
     Smallest interval(s) containing 68% and local modes:
      (268, 298) (local mode at 281 with rel. height 1; rel. area 0.7169)

 (2) Parameter "Signal":
     Mean +- sqrt(V):              83.59 +- 11.09
     Median +- central 68% interval: 83.28 +  11.35 - 10.73
     (Marginalized) mode:          83
      5% quantile:                 65.85
     10% quantile:                 69.55
     16% quantile:                 72.55
     84% quantile:                 95.13
     90% quantile:                 97.99
     95% quantile:                 102.4
     Smallest interval(s) containing 68% and local modes:
      (72, 96) (local mode at 83 with rel. height 1; rel. area 0.6806)

  (4) Parameter "Signal mass":
     Mean +- sqrt(V):              2038 +- 0.8009
     Median +- central 68% interval: 2038 +  0.7945 - 0.7922
     (Marginalized) mode:          2038
      5% quantile:                 2037
     10% quantile:                 2037
     16% quantile:                 2037
     84% quantile:                 2039
     90% quantile:                 2039
     95% quantile:                 2039
     Smallest interval(s) containing 68% and local modes:
      (2037, 2039) (local mode at 2038 with rel. height 1; rel. area 0.6844)
...

Results of the optimization
===========================
Optimization algorithm used:Metropolis MCMC
List of parameters and global mode:
 (0) Parameter "Background": 22.68%
 (2) Parameter "Signal": 19.76%
 (4) Parameter "Signal mass": 23.64%
 (5) Parameter "Signal width": 19.82%

Status of the MCMC
==================
Convergence reached:                    yes
Number of iterations until convergence: 24000
Number of chains:                       10
Number of iterations per chain:         10000000
Average efficiencies:
 (0) Parameter "Background": 20.03%
 (2) Parameter "Signal": 17.35%
 (4) Parameter "Signal mass": 24.52%
 (5) Parameter "Signal width": 19.56%
```

An example: the summary

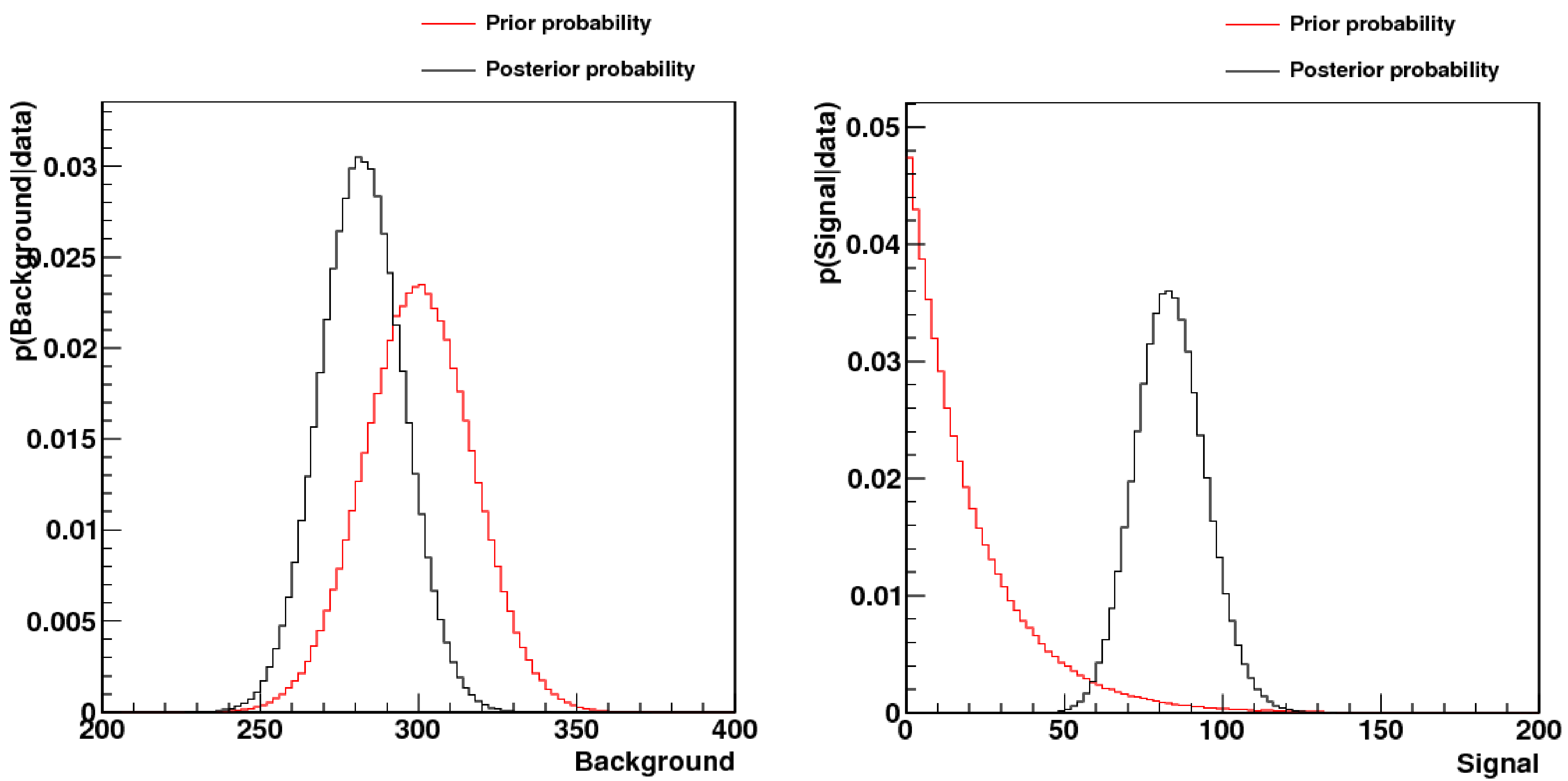


An example: correlation

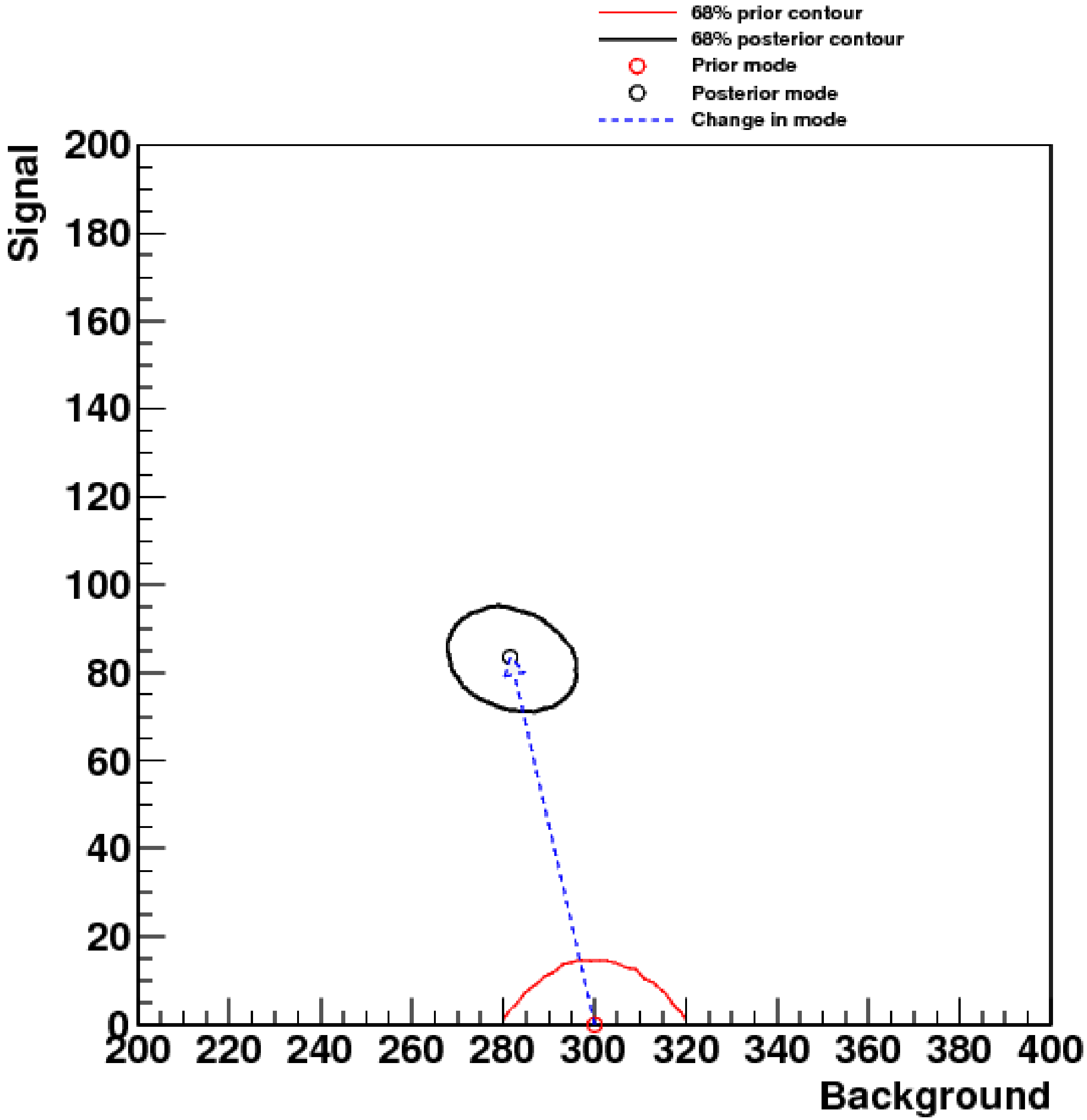


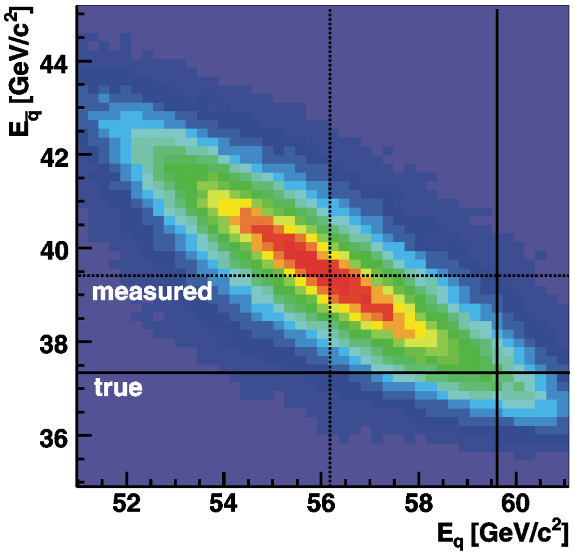


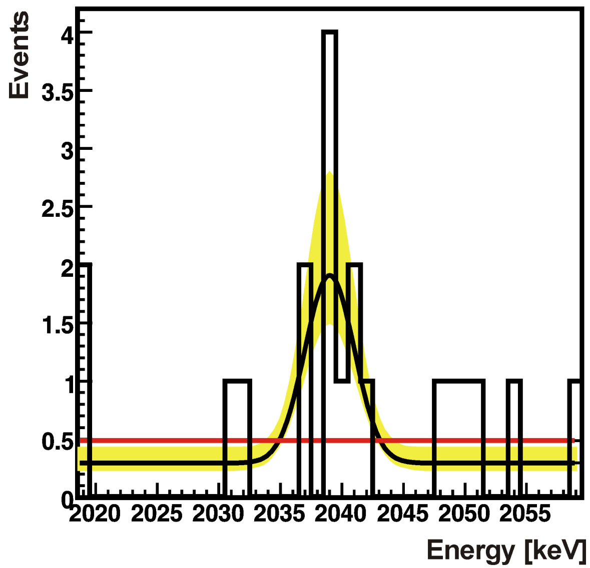
An example: knowledge update

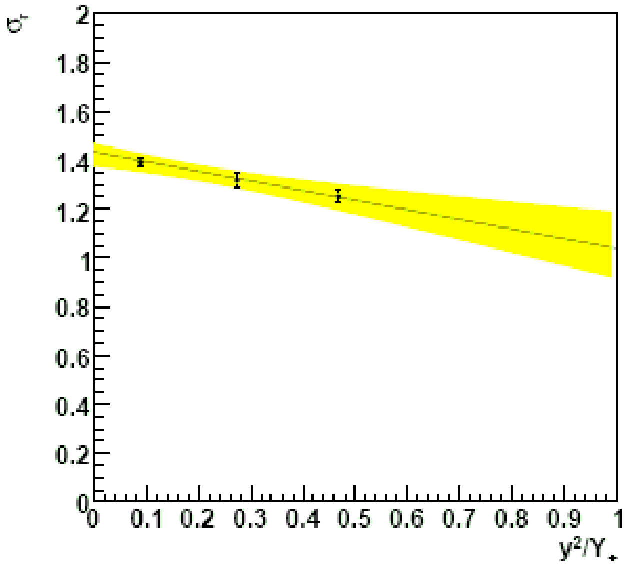


An example: knowledge update

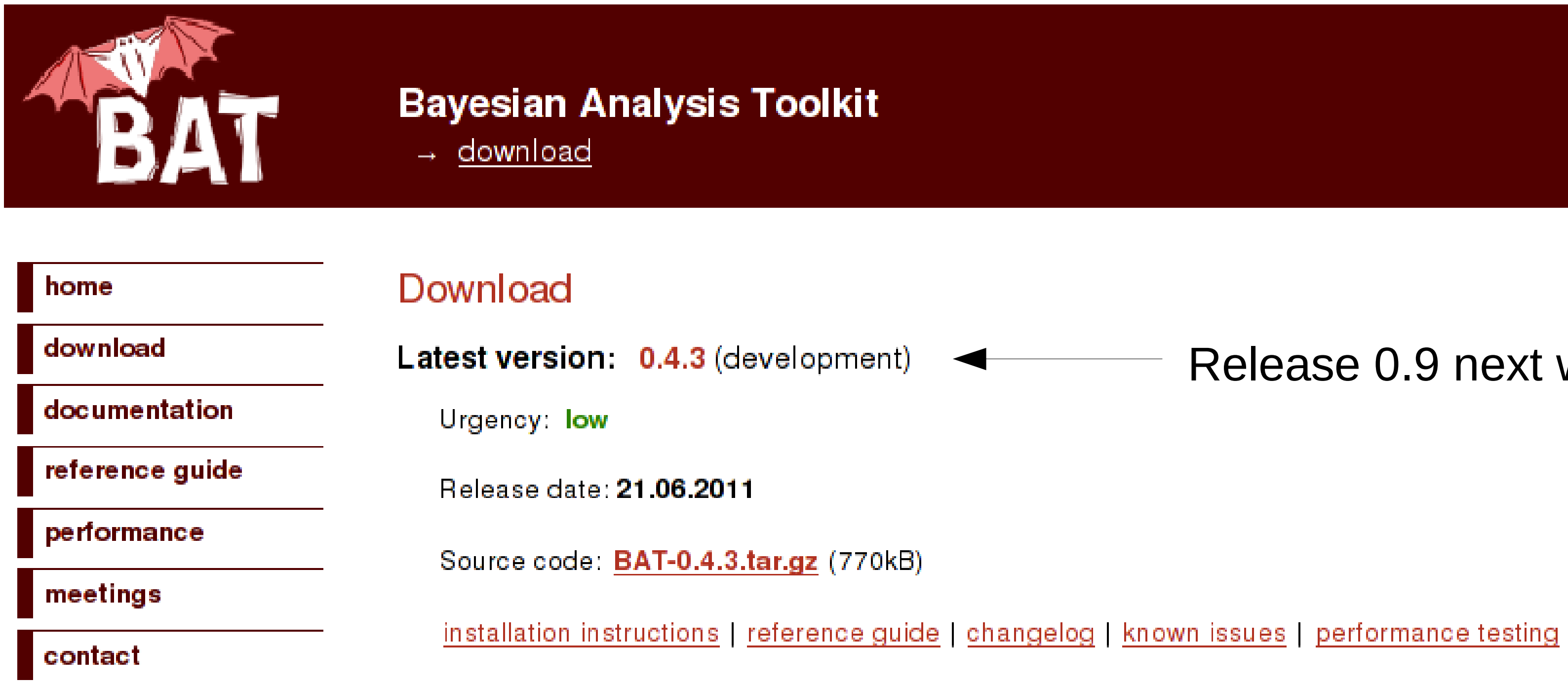








- Quentin Buat, *Search for extra dimensions in the diphoton final state with ATLAS* [arXiv:1201.4748]
- ATLAS collaboration, *Search for excited leptons in proton-proton collisions at sqrt(s) = 7 TeV with the ATLAS detector* [arXiv:1201.3293]
- I. Abt *et al.*, *Measurement of the temperature dependence of pulse lengths in an n-type germanium detector,* Eur. Phys. J. Appl. Phys.56:10104,2011 [arXiv:1112.5033]
- ATLAS collaboration, *Search for Extra Dimensions using diphoton events in 7 TeV proton-proton collisions with the ATLAS detector* [arXiv:1112.2194]
- ATLAS collaboration, *A measurement of the ratio of the W and Z cross sections with exactly one associated jet in pp collisions at sqrt(s) = 7 TeV with ATLAS*, Phys.Lett.B708:221-240,2012 [arXiv:1108.4908]
- ZEUS collaboration, *Search for single-top production in ep collisions at HERA*, Phys.Lett.B708:27-36,2012 [arXiv:1111.3901]
- CMS collaboration, *Search for a W' boson decaying to a muon and a neutrino in pp collisions at sqrt(s) = 7 TeV*, Phys.Lett.B701:160-179,2011 [arXiv:1103.0030]
- ZEUS collaboration, *Measurement of the Longitudinal Proton Structure Function at HERA,* Phys.Lett.B682:8-22,2009 [arXiv:0904.1092]



Release 0.9 next week

Contact:

- Web page: http://www.mppmu.mpg.de/bat/
- Contact: bat@mppmu.mpg.de
- Paper on BAT:

  A. Caldwell, D. Kollar, K. Kröninger, BAT - The Bayesian Analysis Toolkit
  Comp. Phys. Comm. 180 (2009) 2197-2209 [arXiv:0808.2552].

Tutorials

license contact

Bayesian Analysis Toolkit
→ tutorials

Last updated: February 21st, 2011

- home
- download
- documentation
- reference guide
- performance
- meetings
- contact

BAT Tutorials

The tutorials are intended for the latest version of BAT (unless stated otherwise). However, after a new release they may need some adjustment to work. We try to do the necessary adjustments shortly after the release.

Title	Category	Level
Mesuring a decay rate	counting experiment	basic
Estimating trigger efficiencies	fitting	basic
Charged current cross-section analysis	limit setting	basic
Signal search in the presence of background	hypothesis testing, template fitting	intermediate
Combination of cross-sections	combination	intermediate

Tutorials:

- Set of tutorials on the web page for first steps, including solutions

Current projects:

- BAT version v1.0
- ROOT-less version > v1.0
- Parallelization
- Graphical representation of uncertainty bands
  as in Eur. Phys. J. Plus 127 (2012) 24 [arXiv:1112.2593]

Warrant:

- If you are interested joining the effort, please get in touch with us
- Also have (Bsc./MSc.) thesis projects to offer

Summary

Summary:

- Bayesian inference requires some computational effort (e.g., nuisance parameters)
- Markov Chain Monte Carlo is the key tool to solve these issues
- BAT is a tool to combine Bayesian inference with MCMC
- Toolbox with more algorithms (integration, optimization, etc.)
- C++ library, modular, easy to use
- Informative output with predefined plots, numbers, etc.
- Did not talk about hypothesis testing and goodness-of-fit, p-values, Bayes factors, information criteria
- Upgrade of BAT ongoing, more to come
- Participation and feedback are always welcome

What exactly is being done in BAT?

- Step 1: Starting points
  - Random within parameter space (default)
  - Center of each dimension
  - User-defined
- Step 2: Burn-in
  - Use multiple chains (default: 5)
  - Run until convergence is reached and chains are efficient
  - Or run until the maximum number of iterations is reached
  - Chains are efficient if the efficiency is between 15% and 50%
  - Run in sequences to adjust the width of the proposal functions:
    - If efficiency > 50%: increase the width
    - If efficiency < 15%: decrease the width

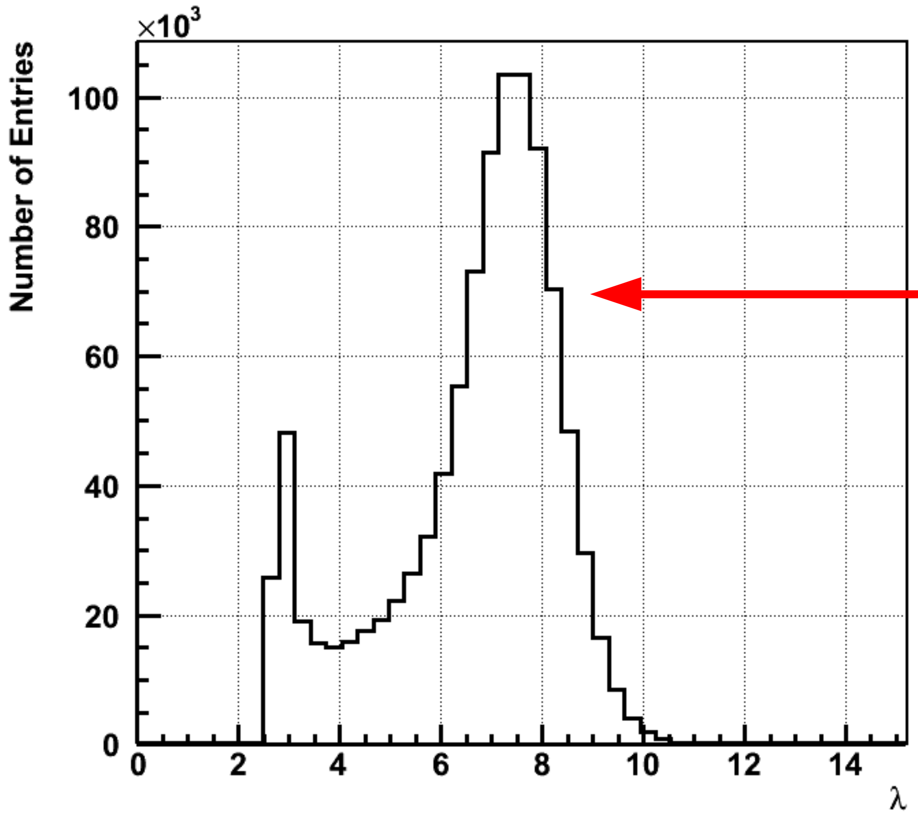
What exactly is being done in BAT?

- Step 3: Main run
  - Use width obtained from efficiency optimization and convergence (fixed)
  - Run for a specified number of iterations
  - Perform analysis-specific calculations (next slide)
  - Store information of every *n*th iteration (consider lag)

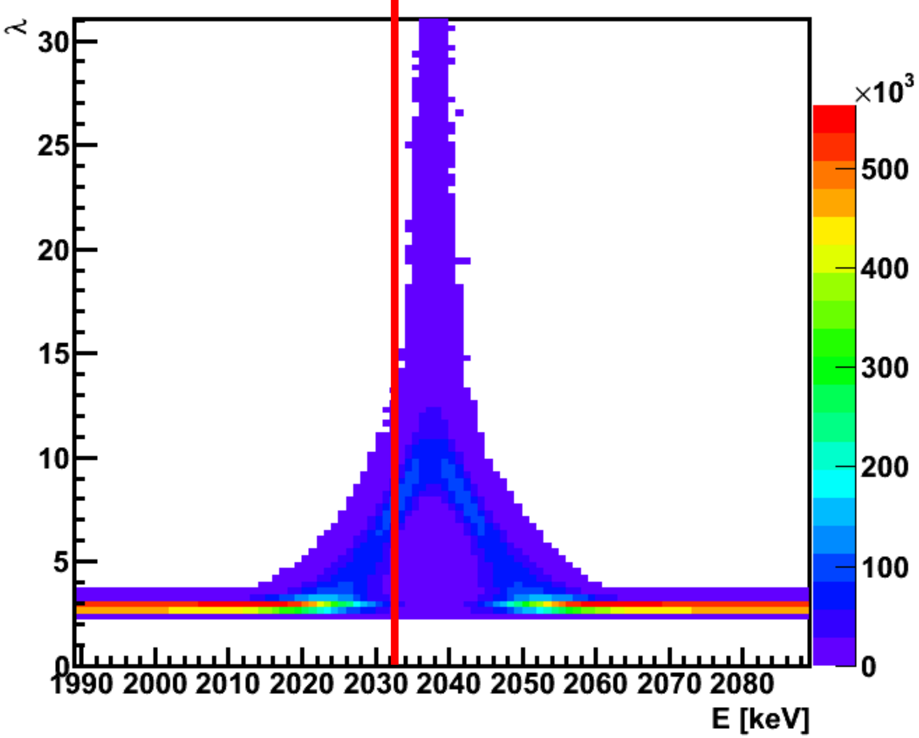
What is done in each step?

- Marginalization:
  - Fill 1-D and 2-D histograms
  - Large number: $N\cdot(N+1)/2$, e.g., for N=50 there are 1275 histograms
  - Individual histograms can be switched on/off
- Optimization:
  - Search for maximum of posterior
  - Not precise, but helpful as starting point for other algorithms
- Error propagation:
  - Calculate arbitrary (user-defined) functions from parameters
- Misc:
  - Write points to ROOT tree for offline analysis
  - Perform any user-defined analysis, histogram filling, etc.

An example: error propagation



Posterior probability for the number of expected events with energy E=2032 keV



Sum of all possible fit functions weighted with posterior: calculate fit function at energy E for all parameter values

Use as error band