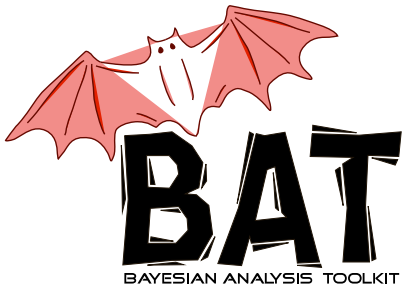
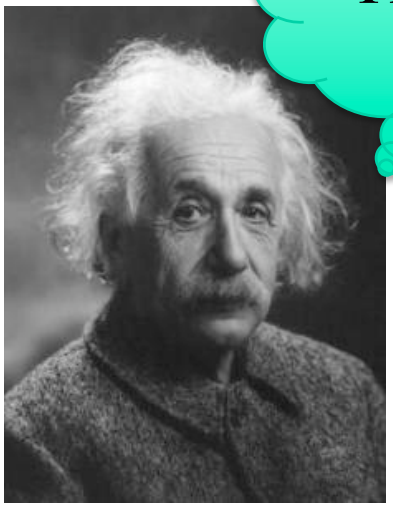
Data Analysis as Knowledge Update

A. Caldwell
Max Planck Institute for Physics

1. Logical framework for data analysis
2. The knowledge-update scheme
3. Example use in exponential slope measurement
    1. With data unfolding
    2. The BAT package
    3. Without data unfolding





Theory

$g(\vec{y}|\vec{\lambda}, M)$

$\vec{y}$ Are the theoretical observables

$\vec{\lambda}$ Are the parameters of the theory

M Is the model or theory

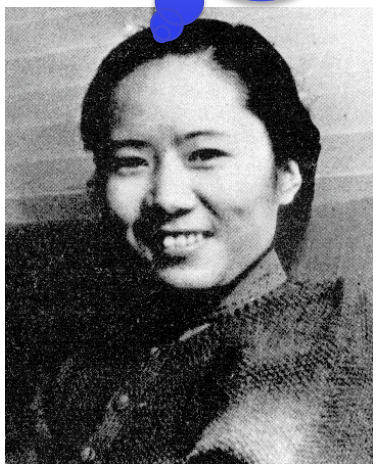
Modeling of experiment

$f(\vec{x}|\vec{\lambda}, M)$

compare

$\vec{D}$

Experiment

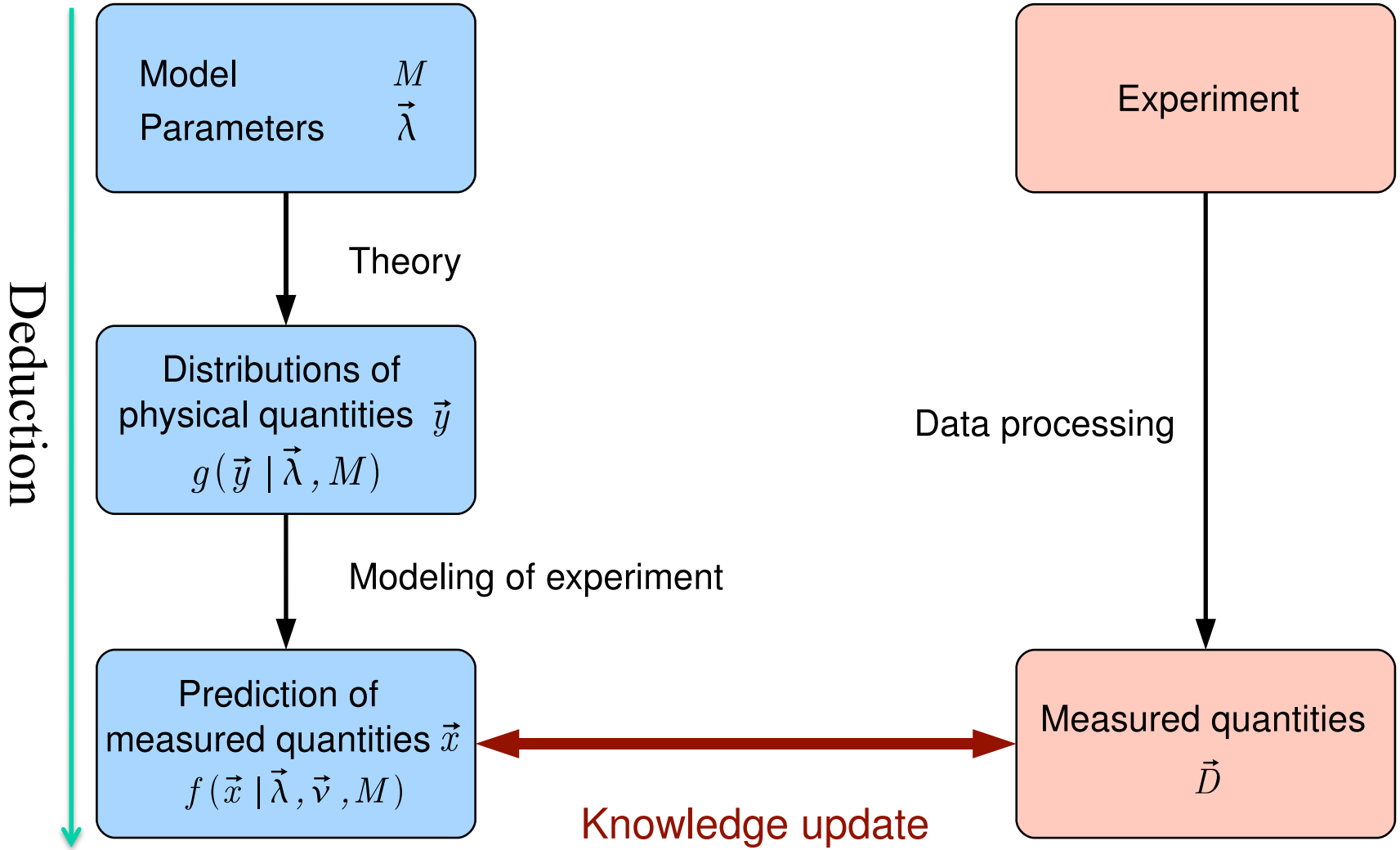


$\vec{x}$ Is a possible data outcome

Unfolding means trying to bring the data to the level of: $\vec{y}$

It is no longer 'pure' data – depends on model and theory.

How we learn


Model M
Parameters $\vec{\lambda}$
Theory
Distributions of physical quantities $\vec{y}$
$g(\vec{y} \mid \vec{\lambda}, M)$
Modeling of experiment
Prediction of measured quantities $\vec{x}$
$f(\vec{x} \mid \vec{\lambda}, \vec{\nu}, M)$
Deduction
Experiment
Data processing
Measured quantities
$\vec{D}$
Knowledge update

How we Learn

We learn by comparing measured data with distributions for predicted results assuming a theory, parameters, and a modeling of the experimental process.

What we typically want to know:

• Is the theory reasonable ? I.e., is the observed data a likely result from this theory (+ experiment).

• If we have more than one potential explanation, then we want to be able to quantify which theory is more likely to be correct given the observations

• Assuming we have a reasonable theory, we want to estimate the most probable values of the parameters, and their uncertainties. This includes setting limits (>< some value at XX% probability).

Logical Basis

Model building and making predictions from models follows deductive reasoning:

Given A➔B (major premise)
Given B➔C (major premise)
Then, given A you can conclude that C is true

etc.

Everything is clear, we can make frequency distributions of possible outcomes within the model, etc. This is math, so it is correct …

Logical Basis

However, in physics what we want to know is the validity of the model given the data. i.e., logic of the form:

Given A➔C
Measure C, what can we say about A ?

Well, maybe A_1➔C, A_2➔C, …

We now need inductive logic. We can never say anything absolutely conclusive about A unless we can guarantee a complete set of alternatives A_i and only one of them can give outcome C. This does not happen in science, so we can never say we found the true model.

Logical basis

Instead of truth, we consider knowledge

Knowledge = **justified ~~true~~ belief**

Justification comes from the data.

Start with some knowledge or maybe plain belief

Do the experiment

Data analysis gives updated knowledge

Bayesians and Frequentists

Frequentists make statements of the kind:

'Assuming the model is correct, this result will occur in XX% of the experiments'

The model is **assumed true**, and estimators for the true parameters in the model are produced from the data.

In the 'classical' approach, this is then converted to 'assuming the model, the bounds [a,b] will contain the true value in XX% of experiments performed' (confidence levels). Does not imply that the true value is in the range [a,b] with probability XX !

The decision on whether to then believe the model/parameters is left to the individual (subjective). *The inductive part of the reasoning is left out of the analysis.*

Bayesians and Frequentists

Bayesians make statements of the kind:

'the degree-of-belief in model A is XX (between 0,1)'

Given the new data, the degree-of-belief is updated using the frequencies of possible outcomes in the context of the models (full set)

Credible regions are then defined: with XX% credibility, the parameter is in the interval [a,b]. **Note – very different from a CL.**

The inductive part of the reasoning is built in to the analysis, and the connection between prior beliefs and posterior beliefs is made clear.

Subjective, but the subjective element is made explicit.

Bayesians and Frequentists

In both approaches, work with models and frequencies of outcomes within the model.

Many elements are the same: calculating the frequencies of possible outcomes given the model AND the experimental conditions; picking the most sensitive variables to test the theory, …

There is no right and wrong approach, but you have to understand what you get out of each type of analysis. E.g., don't confuse confidence levels with probabilities, p-values with support for a model, …

In the Bayesian approach, if you cannot formulate a prior (e.g., limits on SUSY parameters), then cannot talk about knowledge or degree-of-belief.

…

Formulation of Data Analysis

In the following, I will formulate data analysis as a knowledge-updating scheme.

Knowledge+data ➔ updated knowledge

This leads to the usual Bayes' equation, but I prefer this derivation to the usual one in the textbooks.

Formulation-introduction

The expected distribution (density) of the data assuming a model M and parameters $\vec{\lambda}$ is written as $P(\vec{x}|\vec{\lambda}, M)$ where $\vec{x}$ is a possible realization of the data. There are different possible definitions of this function.

Imagine we flip a coin 10 times, and get the following result:

T H T H H T H T T H

We now repeat the process with a different coin and get

T T T T T T T T T T

Which outcome has higher probability ?

Take a model where H, T are equally likely. Then,

outcome 1 $prob = (1/2)^{10}$

And

outcome 2 $prob = (1/2)^{10}$

Something seem wrong with this result ? This is because we evaluate many probabilities at once. The result above is the probability for any sequence of ten flips of a fair coin. Given a fair coin, we could also calculate the chance of getting n times H:

$$\binom{10}{n}\left(\frac{1}{2}\right)^{10}$$

And we find the following result:

n	p
0	$1 \cdot 2^{-10}$
1	$10 \cdot 2^{-10}$
2	$45 \cdot 2^{-10}$
3	$120 \cdot 2^{-10}$
4	$210 \cdot 2^{-10}$
5	$252 \cdot 2^{-10}$
6	$210 \cdot 2^{-10}$
7	$120 \cdot 2^{-10}$
8	$45 \cdot 2^{-10}$
9	$10 \cdot 2^{-10}$
10	$1 \cdot 2^{-10}$

There are many more ways to get 5 H than 0, so this is why the first result somehow looks more probable, even if each sequence has exactly the same probability in the model.

Maybe the model is wrong and one coin is not fair ? How would we test this ?

The message: there are usually many ways to define the probability for your data. Which is better, or whether to use several, depends on what you are trying to do.

E.g., have measured times in exponential decay. Can define the probability density as

$$P(\vec{t}|\tau) = \prod_{i=1}^{N} \frac{1}{\tau} e^{-t_i/\tau}$$

Or you can count events in a time interval and compare to expectations

$$P(\vec{t}|\tau) = \prod_{j=1}^{M} \frac{e^{-\nu_j} \nu_j^{n_j}}{n_j!}$$

ν_j = expected events in bin j

n_j = observed events in bin j

Formulation

We require that $P(\vec{x}|\vec{\lambda}, M) \geq 0 \quad \int P(\vec{x}|\vec{\lambda}, M) d\vec{x} = 1$

although as we will see the normalization condition is not really needed.

The modeling of the experiment will typically add other (nuisance) parameters. E.g., there are often uncertainties, such as, e.g., the energy scale of the experiment. Different assumptions on these lead to different predictions for the data. Can have $P(\vec{x}|\vec{\lambda}, \vec{\nu}, M)$

where $\vec{\nu}$ represents our nuisance parameters.

Formulation

For the model, we have $0 \leq P(M) \leq 1$. For a fully Bayesian analysis, we require

$$\sum_i P(M_i) = 1$$

For the parameters, assuming a model, we have:

$$\begin{aligned} P(\vec{\lambda}|M_i) &\geq 0 \\ \int P(\vec{\lambda}|M_i)d\vec{\lambda} &= 1 \end{aligned}$$

The joint probability distribution is $P(\vec{\lambda}, M) = P(\vec{\lambda}|M)P(M)$

and $$\sum_i P(M_i) \int P(\vec{\lambda}|M_i)d\vec{\lambda} = 1$$

Learning Rule

$$P_{i+1}(\vec{\lambda}, M|\vec{D}) \propto P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)$$

where the index represents a 'state-of-knowledge'

We have to satisfy our normalization condition, so

$$P_{i+1}(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)}{\sum_M \int P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)d\vec{\lambda}}$$

We usually write $P_i = P_0$. This is our 'prior' information before performing the measurement.

Learning Rule

$$P_{i+1}(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)}{\sum_M \int P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)d\vec{\lambda}}$$

The denominator is the probability to get the data summing over all possible models and all possible values of the parameters.

$$P(\vec{D}) = \sum_M \int P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)d\vec{\lambda}$$

so

$$P(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda}, M)P(\vec{\lambda}, M)}{P(\vec{D})}$$

Bayes Equation

Parameter Estimation

The posterior pdf gives the full probability distribution for all parameters, including all correlations – no approximations. If interested in subset of parameters, then marginalize. E.g., for one parameter:

$$P(\lambda_i|\vec{D},M) = \int P(\vec{\lambda}|\vec{D},M)d\vec{\lambda}_{\mathrm{J}\neq i}$$

Can calculate what you need from the posterior pdf. E.g.,

Mode $\max_{\lambda_i} \{P(\lambda_i|D,M)\}$ + probability intervals, …

Mean of λ_i $< \lambda_i >= \int P(\lambda_i|\vec{D},M)\lambda_i d\lambda_i$

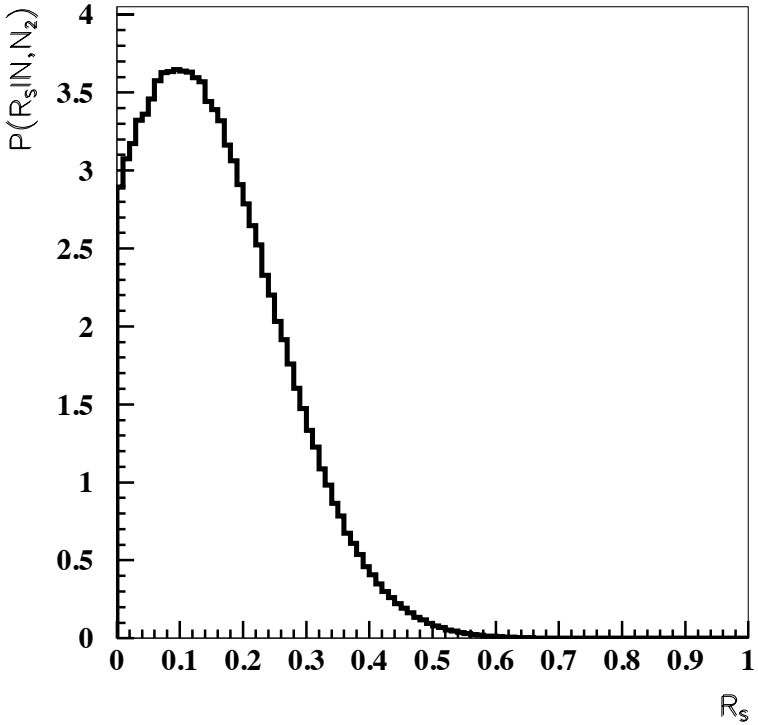
Median $\int_{\lambda_{min}}^{\lambda_{med}} P(\lambda_i|\vec{D},M)d\lambda_i = 0.5$

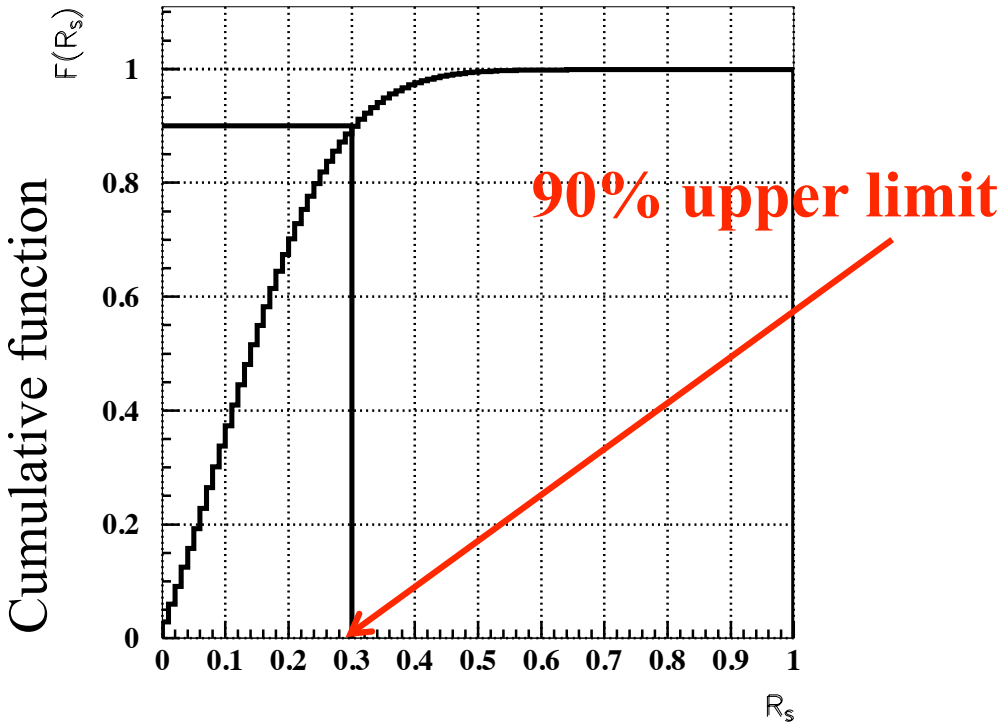
Can also perform uncertainty propagation w/o approximations

Setting Limits

Setting limits is easy – just integrate the posterior pdf. E.g., 90% upper limit:

$$0.9 = \int_{\lambda_{min}}^{\lambda_{upper}} P(\lambda_i|\vec{D}, M)d\lambda_i$$



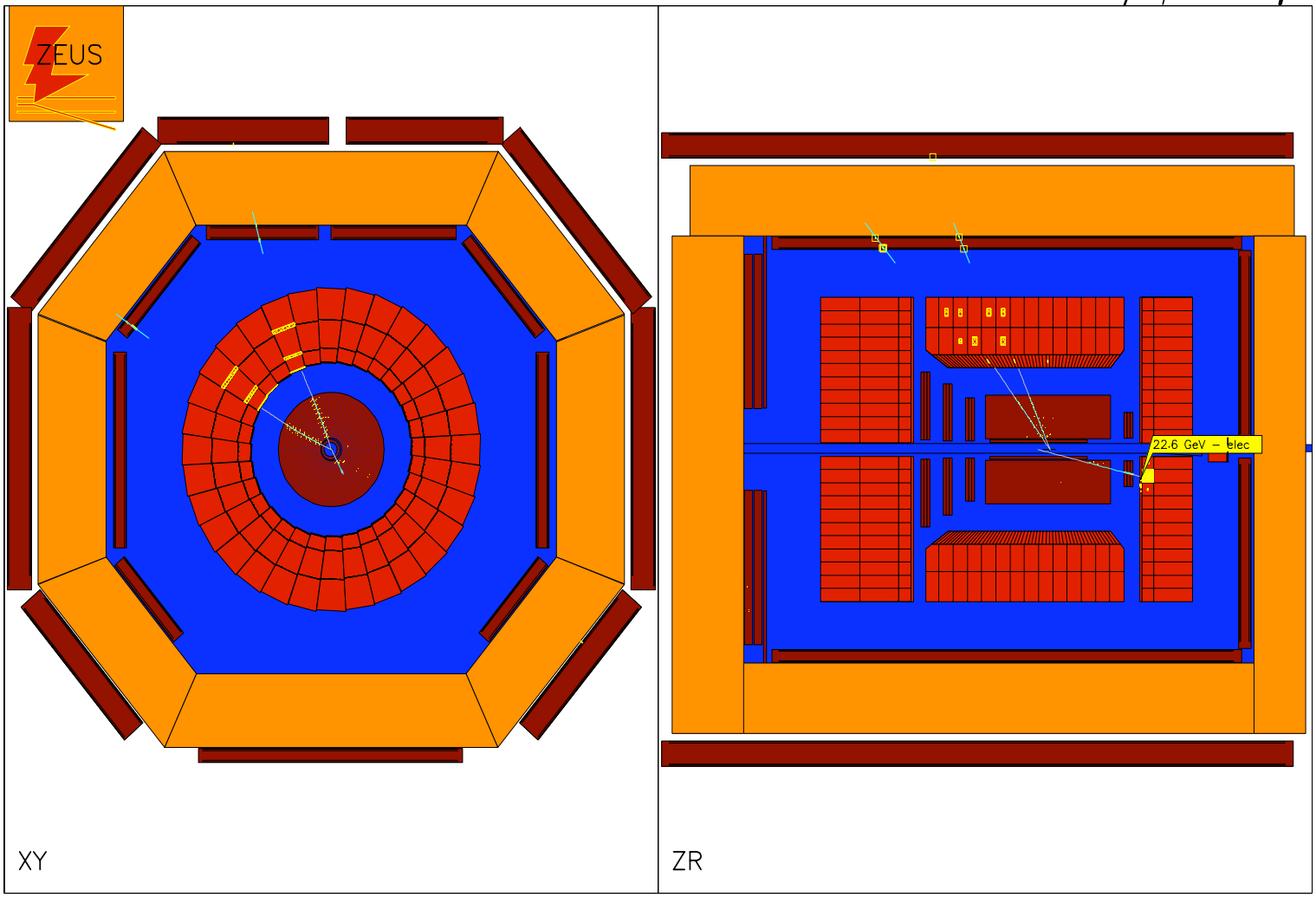


Or calculate contours in higher dimensional spaces

Detailed example

We consider the following process $eP \rightarrow ePJ/\psi$

$J/\psi \rightarrow \mu^+\mu^-$



Scattered proton in this event escapes down beampipe

kinematics

We are interested in the 4-momentum transfer squared to the proton (it is related to the gluon distribution in the proton via a Fourier transform). The expected form is:

$$\frac{d\sigma}{dt} \propto e^{B_S t}$$

Note that *t* is negative

There are two experimental issues to consider:

• *t* is smeared due to the transverse momentum spread of the incoming proton beam

• there is a similar process where the proton breaks up, but the fragments stay in the beampipe, which is a background to the process of interest. It is also expected to have an exponential distribution but with a different slope.

$$\frac{d\sigma}{dt} \propto e^{B_B t}$$

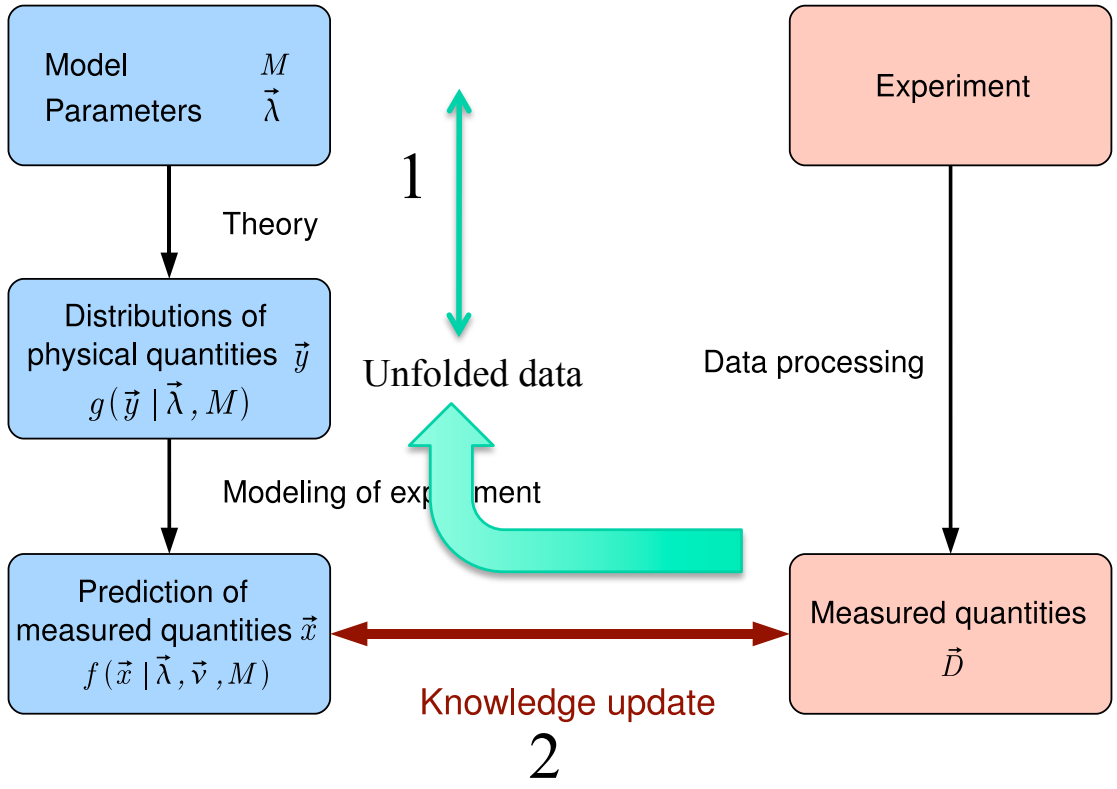
Analysis Task

The analysis task is to extract the slope parameter B_S

Consider two approaches:

1. Subtract the background, then unfold the resulting distribution, then fit extracted cross sections with simple exponential

2. Fit for the signal slope by reweighting the MC prediction for observed distribution for different different B_S



For each approach, we use a simulation to tell us how the observed distribution depends on the assumed model.

Generation of test data

We produce some simulated data as follows:

Assume $\sigma_S = \sigma_B$ $\mathcal{L}\sigma_S = 1000$ events

A data set is generated with 1000 signal and 1000 background events.
The slopes chosen are $B_S = 5\ \mathrm{GeV}^{-1}$ $B_B = 2\ \mathrm{GeV}^{-1}$

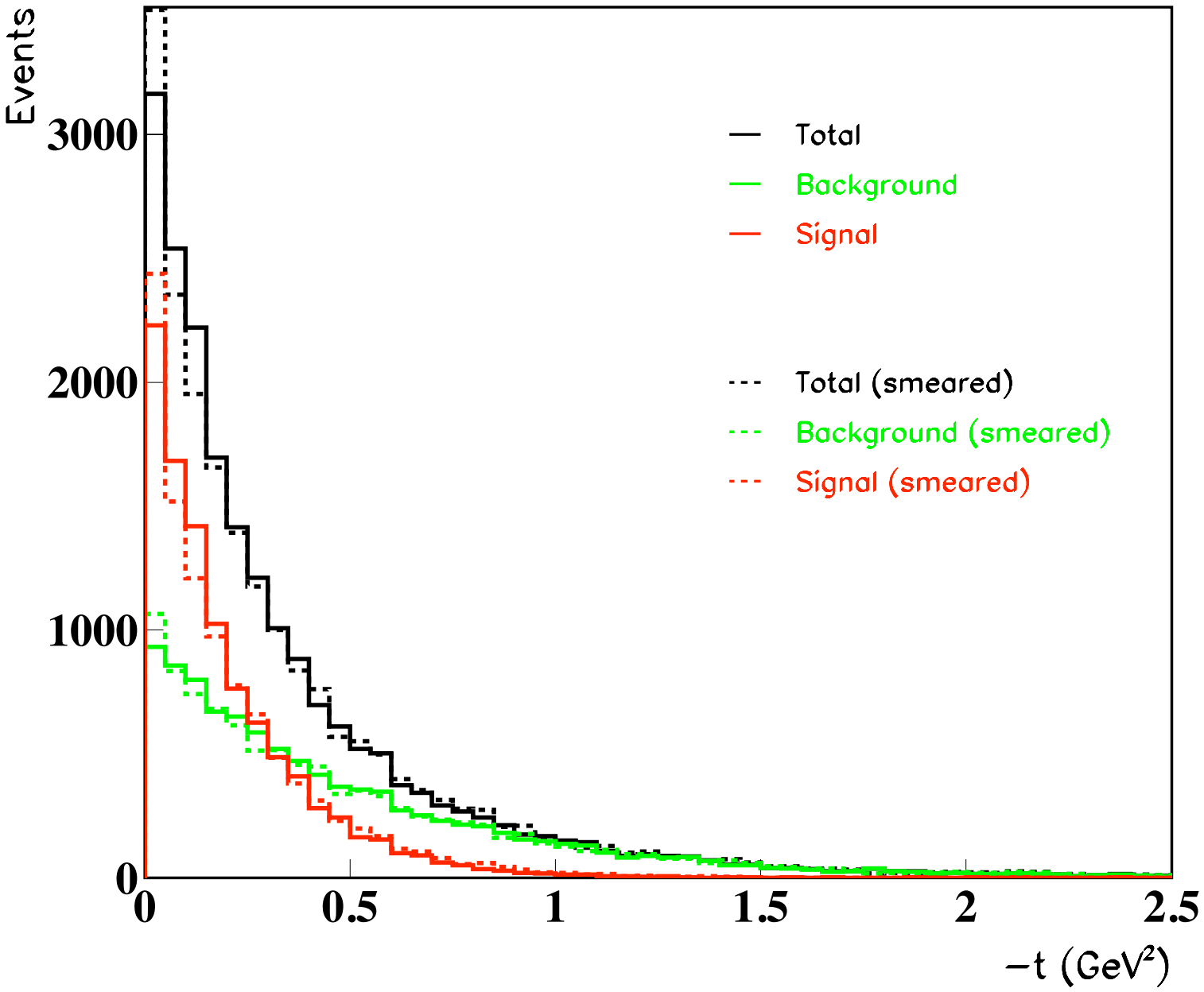
For each data event, a value of *t* is generated using

$$t = -\ln U/B$$

Exercise: show that this leads to an exponential distributrion with slope B.

Where *U* is a random number drawn from a uniform distribution (0,1) and *B* is the slope corresponding to the process at hand.
The value of *t* is then smeared to account for the rms transverse momentum of the incoming proton beam (100 MeV/c assumed).

Generated data



Analysis 1

The standard procedure at this point is to correct the data and present them in the form of cross sections. These cross sections with their uncertainties can then be used by model builders to fit their favorite models.

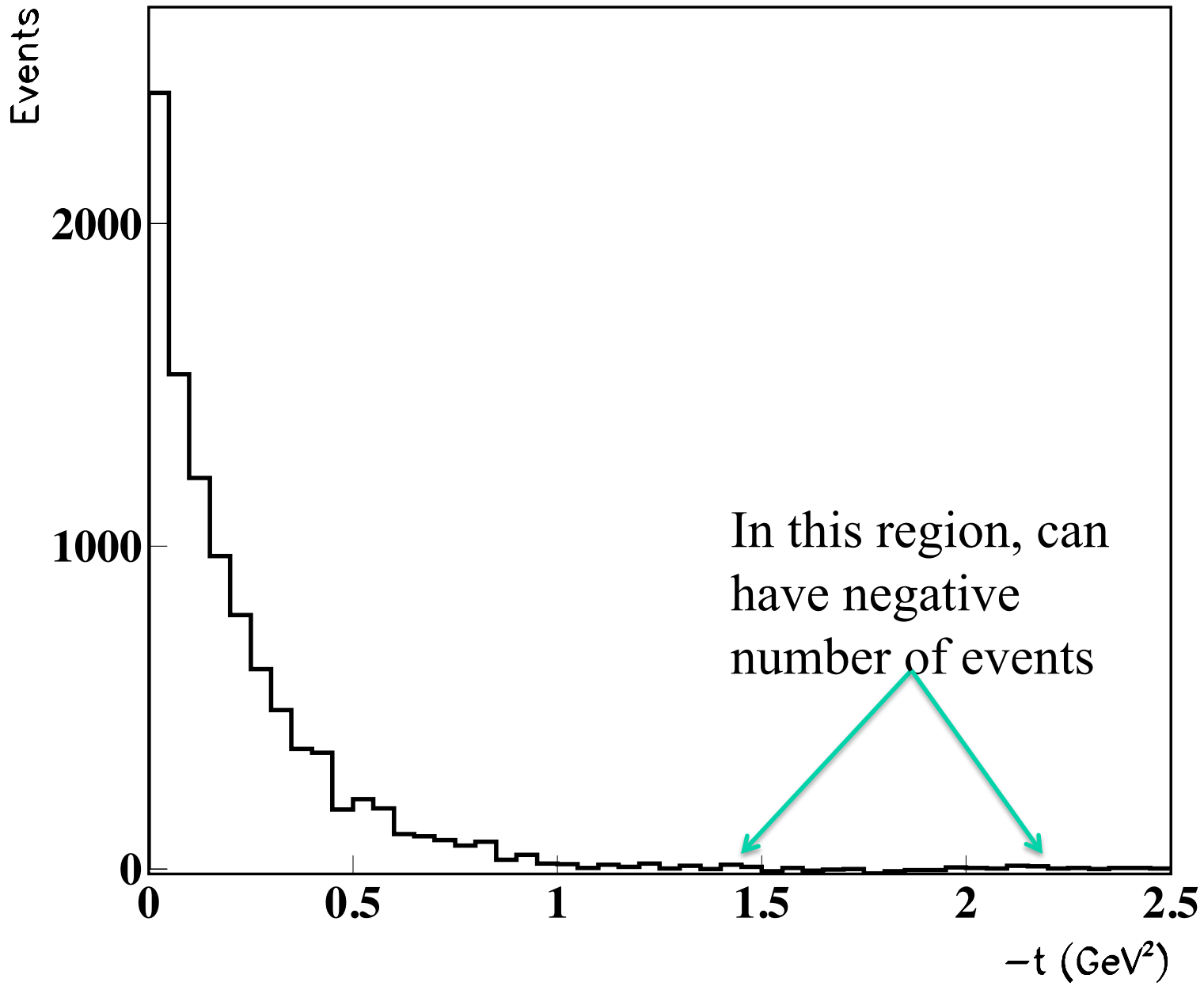
First, we subtract the background:

-Need to know the normalization of the background
-Need to know the slope of the background
-Need to know the smearing due to the experimental conditions

We initially assume we know these perfectly. I.e., we know σ_B and B_B

Procedure used: simulate a large background sample and normalize it to $\mathcal{L}\sigma_B = 1000$ events

Background corrected



Bin-by-bin unfolding

Given the background corrected event distribution, we produce cross sections using the so-called 'bin-by-bin' unfolding method.

$$\frac{d\sigma^{DATA}}{dt} = \frac{d\sigma^{MC}}{dt}\frac{N^{DATA}}{N^{MC}}$$

The *N*'s represent event counts in an interval of *t*. The simulation is used to take out the effects of the theoretical assumptions and all detector effects in one step. Some guidelines are usually used to decide where one can quote cross sections, such as:

- purity in the bin (a reasonable fraction of events reconstructed in the bin should have been initially generated in the bin)
- efficiency (there should be a high enough chance to keep events generated in the bin)

Bin-by-bin unfolding

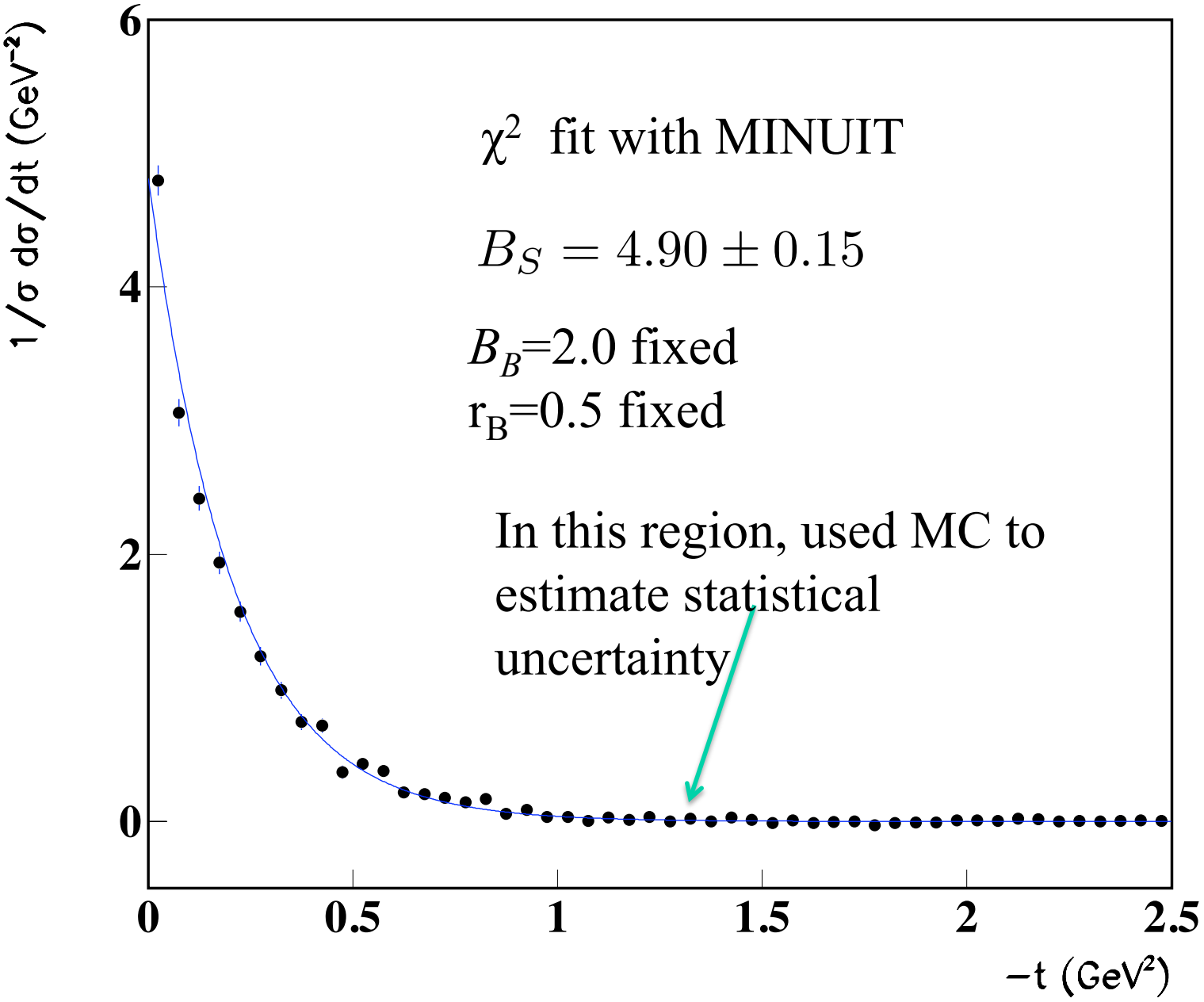
What uncertainty should be assigned to the cross section ? Standard procedure is to take for the statistical uncertainty

$$\delta\left(\frac{d\sigma^{DATA}}{dt}\right) = \left(\frac{d\sigma^{DATA}}{dt}\right)\frac{\sqrt{N^{TOT}}}{N^{DATA}}$$

Where N^{TOT} is the total number of events recorded in the bin. What if $N^{DATA}<0$? Can use theory to estimate.

Note that the statistical uncertainty defined this way depends on the resolution of the detector, the definition of the background, … In the tails of a steeply falling distribution, the measured number of events is always too high due to migrations, so the statistical errors are underestimated. This definition also has conceptual difficulties for small numbers of events. I.e., its what we often do, but not clear (to me) what it means.

Cross sections



Systematic uncertainties in bin-by-bin

Standard approach:

• recalculate cross sections where uncertain element shifted by one standard deviation. E.g., if r_B=0.5±0.1, recalculate cross section with r_B=0.6 and r_B=0.4

• make a table

Cross section bin	Systematic check	Nominal result	Syst 1	Syst 2	…
	1				
	2				
	…				

• redo B_S extraction for each systematic variation, add results in quadrature.

Systematic uncertainties bin-by-bin

Suppose we generated the MC with B_S=6, B_B=1.5, r_B=0.4 and the '1 sigma' uncertainties are 2,1,0.2

fit	B_S	Error
Nominal	5.16	0.14
B_S=8	5.15	0.14
B_S=4	5.17	0.14
B_B=2.5	3.84	0.13
B_B=0.5	5.23	0.11
r_B=0.6	5.67	0.16
r_B=0.2	4.49	0.11

$$B_S = 5.16 \pm 0.14^{+0.51}_{-1.48}$$

Positive and negative deviations added separately in quadrature

Iterative unfolding

Can try to improve on the previous results using iterative unfolding.

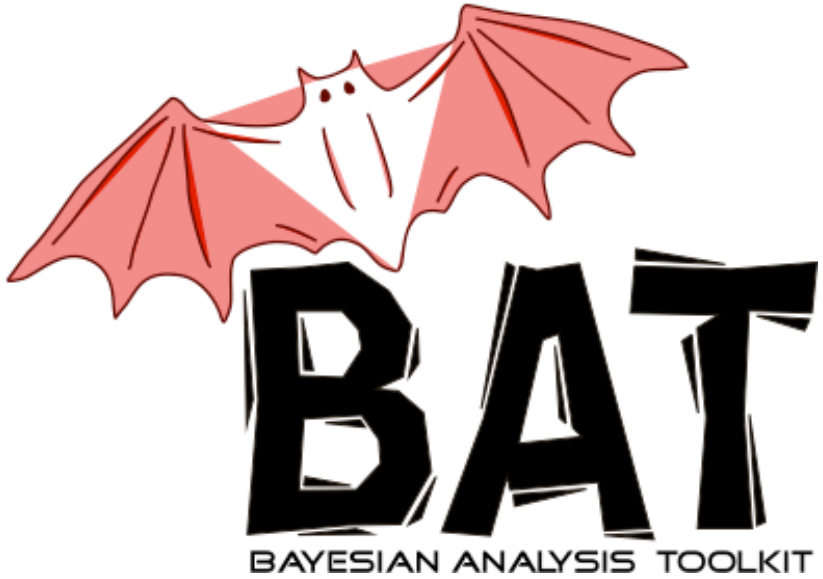
Basic idea: use comparison of unfolded data with model to determine which parameter values are best (and therefore best for unfolding). Repeat unfolding with improved parameter estimates.

However, in many cases do not need to do unfolding. If that is the case, don't do it.

Analysis without unfolding

Here, we will compare directly the observed t distribution with predictions using different slopes for the signal, background, background fraction, etc.

The analysis is performed using the Bayesian Analysis Toolkit – start with an introduction on this new fitting package.



BAT → Software package for solving data analysis problems

Code structured on Bayes' formula for parameter estimation

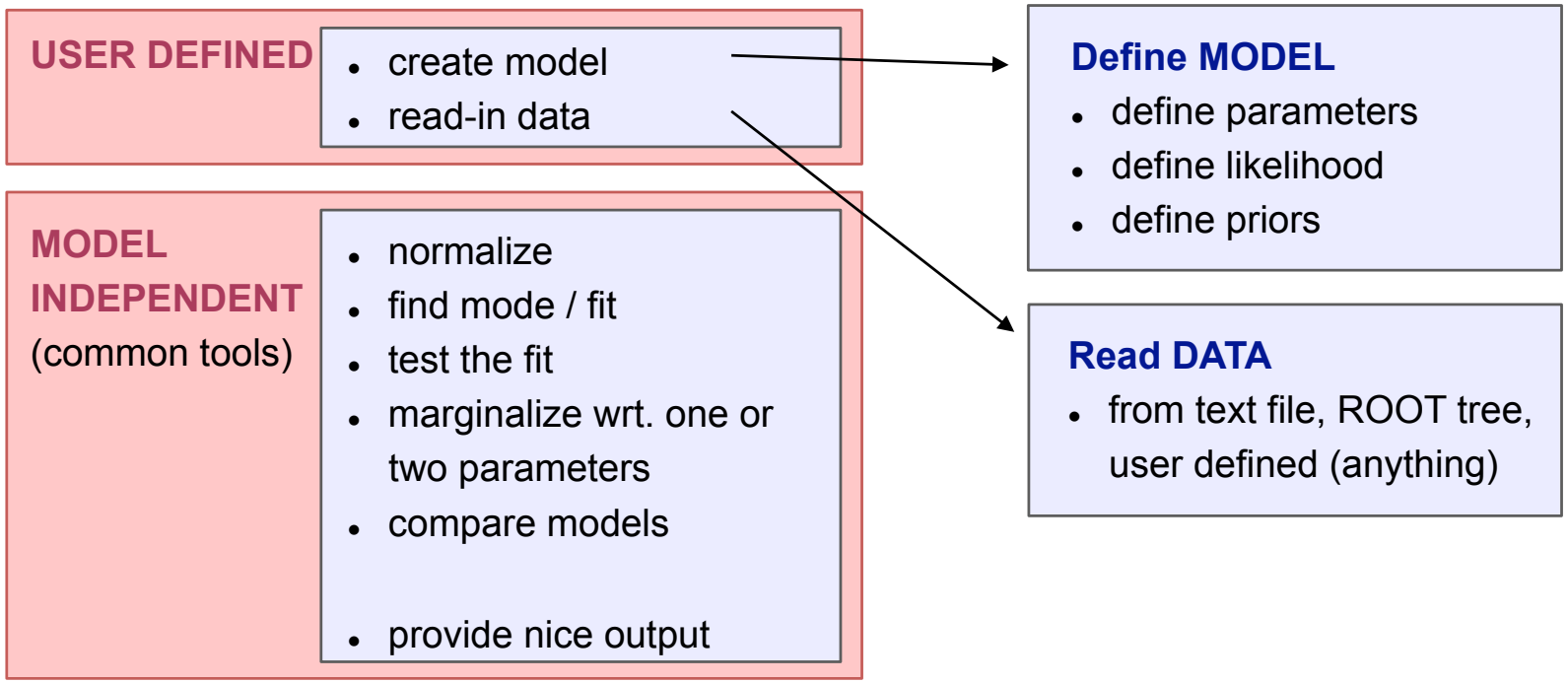
$$P(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda}, M)P(\vec{\lambda}, M)}{P(\vec{D})}$$

- **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, ..
- can be downloaded from: http://www.mppmu.mpg.de/bat

The idea

Separate the common parts from the rest

- case specific: the model and the data
- common tools: all the rest



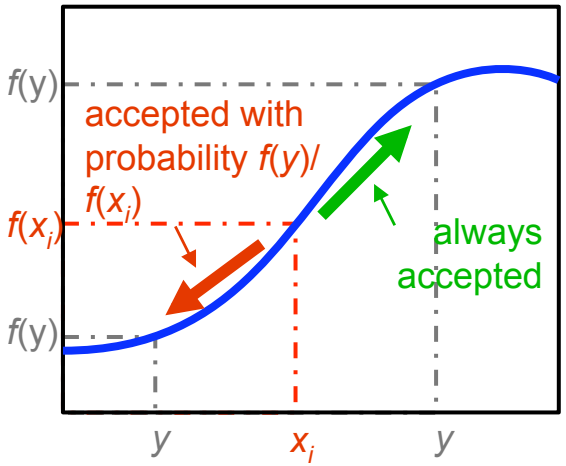
Markov Chain Monte Carlo (MCMC)

- generally it is very difficult to obtain the full posterior PDF
  - number of parameters can be large
  - different input data will result in a different posterior
- also the visualization of the PDF in more than 3 dimensions is rather impractical and hard to understand
- usually one looks at marginalized posterior wrt. one, two or three parameters
  - a projection of the posterior onto one (two, three) parameter
  - integrating all the other parameters out
  - still numerically difficult

- the Markov Chain Monte Carlo revolutionized the area of Bayesian analysis

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 around x_i
  - 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 $f(y)/f(x_i)$
  - If y not accepted, stay where you are, i.e., set $x_{i+1} = x_i$
  - Generate new y , repeat



- For each step fill the histogram with x_{i+1}
- For infinite number of steps the distribution in the histogram converges to ***f(x)*** **(except for the normalization)**

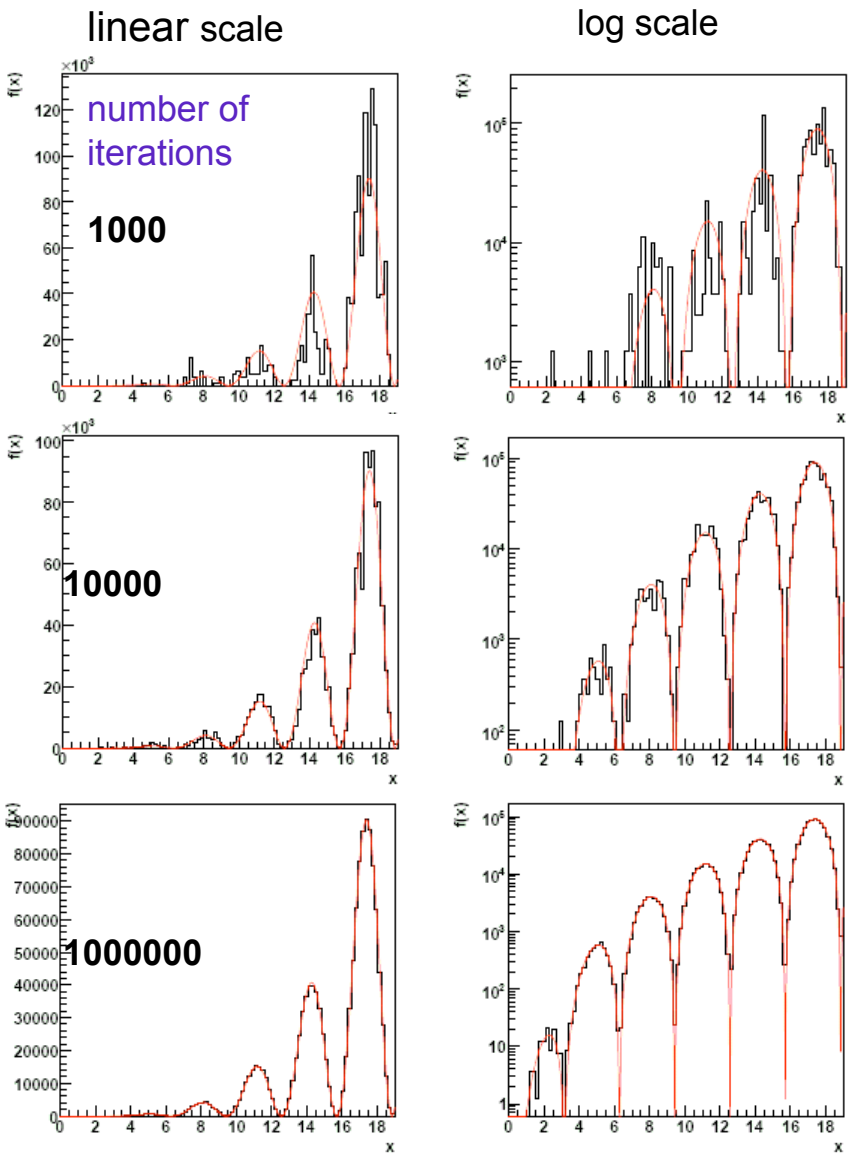
Exercise: try out the Metropolis algorithm to generate a Gaussian distribution from flat rn [0,1]

MCMC: an example

- mapping an arbitrary function:

$$\text{e.g.} \quad f(x) = x^4 \sin^2 x$$

- distribution sampled by MCMC in this case quickly converges towards the underlying distribution
- **mapping of complicated shapes with multiple minima and maxima**



Note:

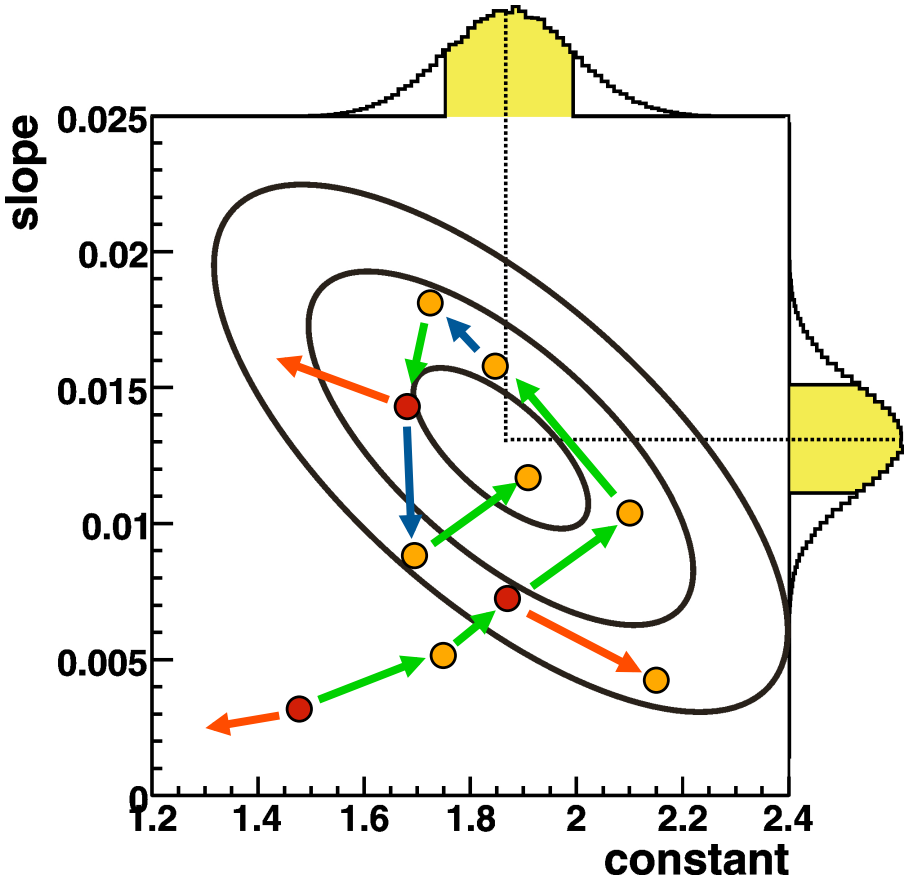
- MCMC has to become stationary to sample from underlying distribution
- in general the convergence is a non-trivial problem

Scanning parameter space with MCMC

- In Bayesian analysis use MCMC to scan parameter space of $\vec{\lambda}$
- MCMC converges towards underlying distribution
- Marginalize wrt. individual parameters while walking → obtain

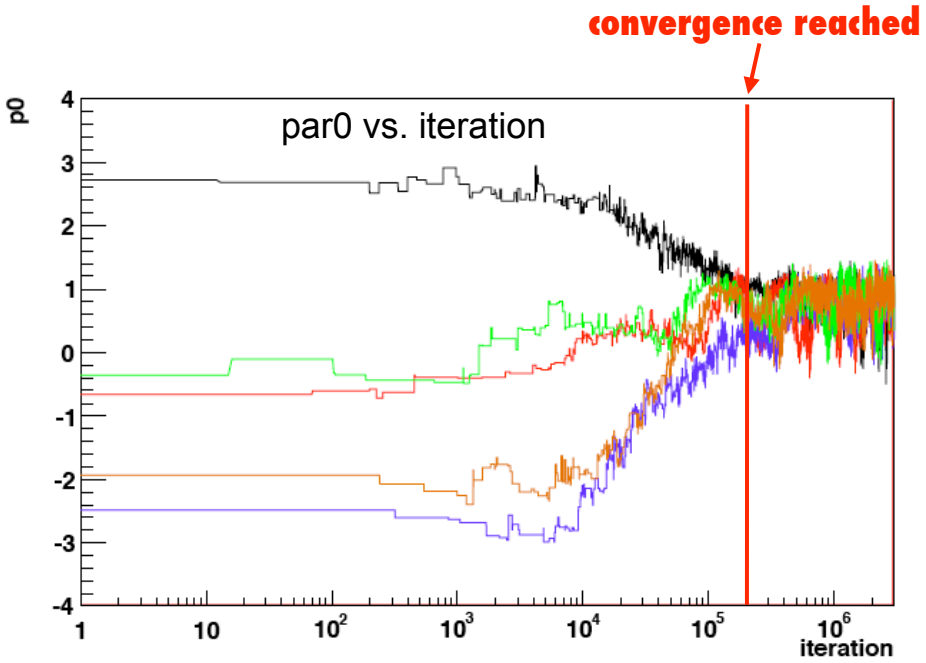
$$P(\lambda_i|\vec{D}) = \int P(\vec{\lambda}|\vec{D}) d\vec{\lambda}_{j\neq i}$$

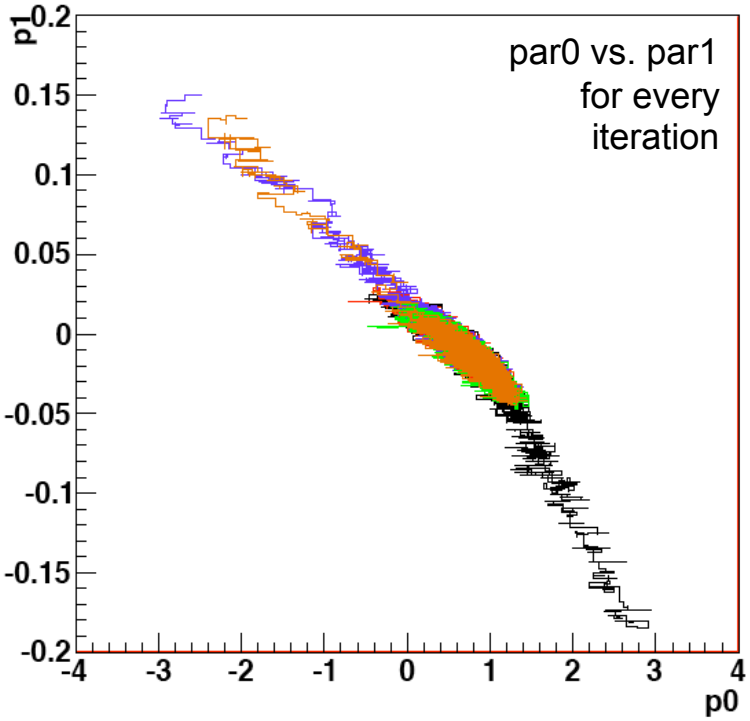
- Find maximum (mode)
- Uncertainty propagation



Analysis of Markov Chain

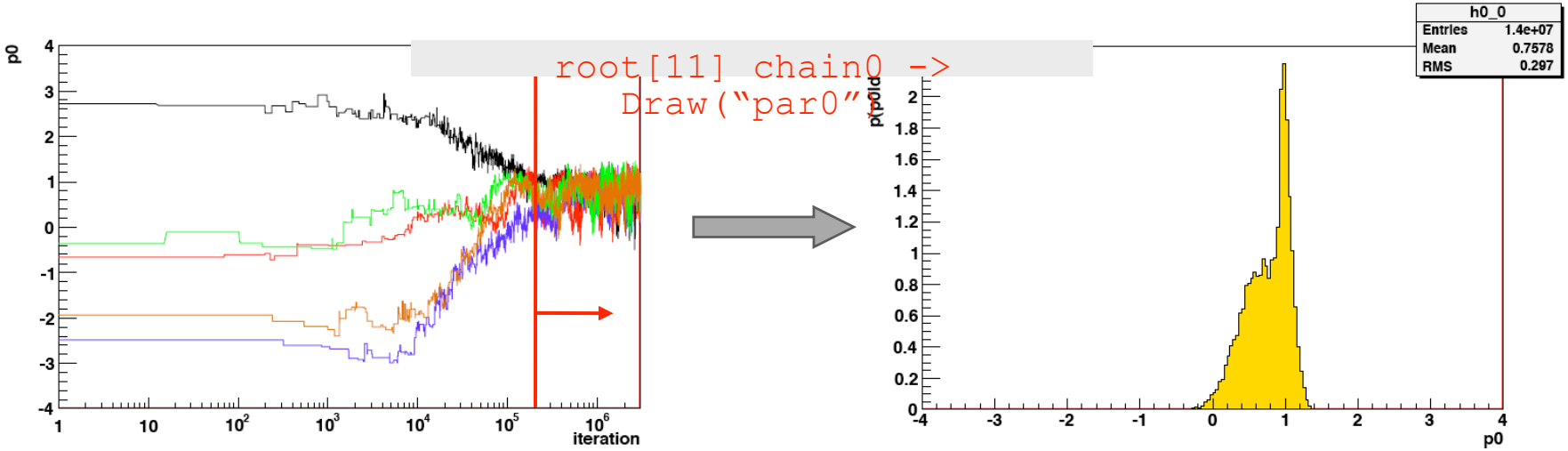
- the full chain(s) can be stored for further analysis and parameter tuning as ROOT TTree(s)
  - allows direct usage of standard ROOT tools for analysis
- Markov Chain contains the complete information about the posterior (except for the normalization)



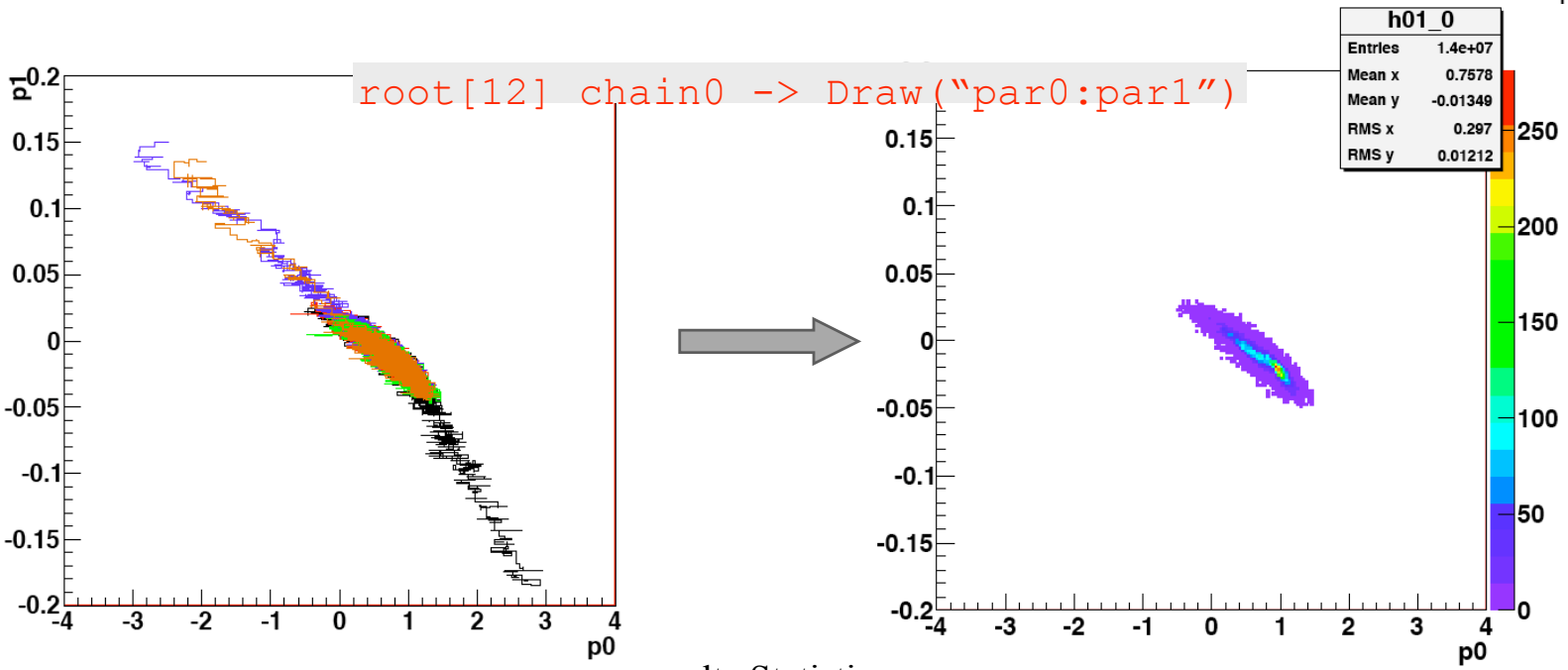


Obtaining marginalized distributions from TTree

```
root[11] chain0 ->
Draw("par0")
```




```
root[12] chain0 -> Draw("par0:par1")
```



Using observed data

We now use BAT to compare data and predictions directly.

The data is the number of events observed in a bin of t_{meas}
The MC is used to predict an observed number. The MC prediction depends on B_S,B_B,r_B where r_B is the fraction of background.

The probability of the data is $$P(\vec{D}|B_S, B_B, r_B) = \prod_i \frac{e^{-\nu_i}\nu_i^{n_i}}{n_i!}$$

n_i data events in bin i

ν_i MC prediction in bin i

$$\nu_i = (1 - r_B)\nu_{i,signal} + r_B\nu_{i,background}$$

Notes:

1. event counting, so Poisson distribution used
2. The data has no error ! Calculate probability for ν_i to give n_i

Weighted bin content

The MC prediction for a bin is updated depending on the parameter values as follows (for the signal, similar for background):

$$\nu_{i,signal} = \sum_{j=1}^{N_{signal}} I(t_{meas}, i) w_j$$

N_{signal} is the number of MC signal events

$$I(t_{meas}, i) = 1 \text{ if } i\Delta t \leq t_{meas} < (i+1)\Delta t$$

I starts at 0

Δt is the bin size

else

$$I(t_{meas}, i) = 0$$

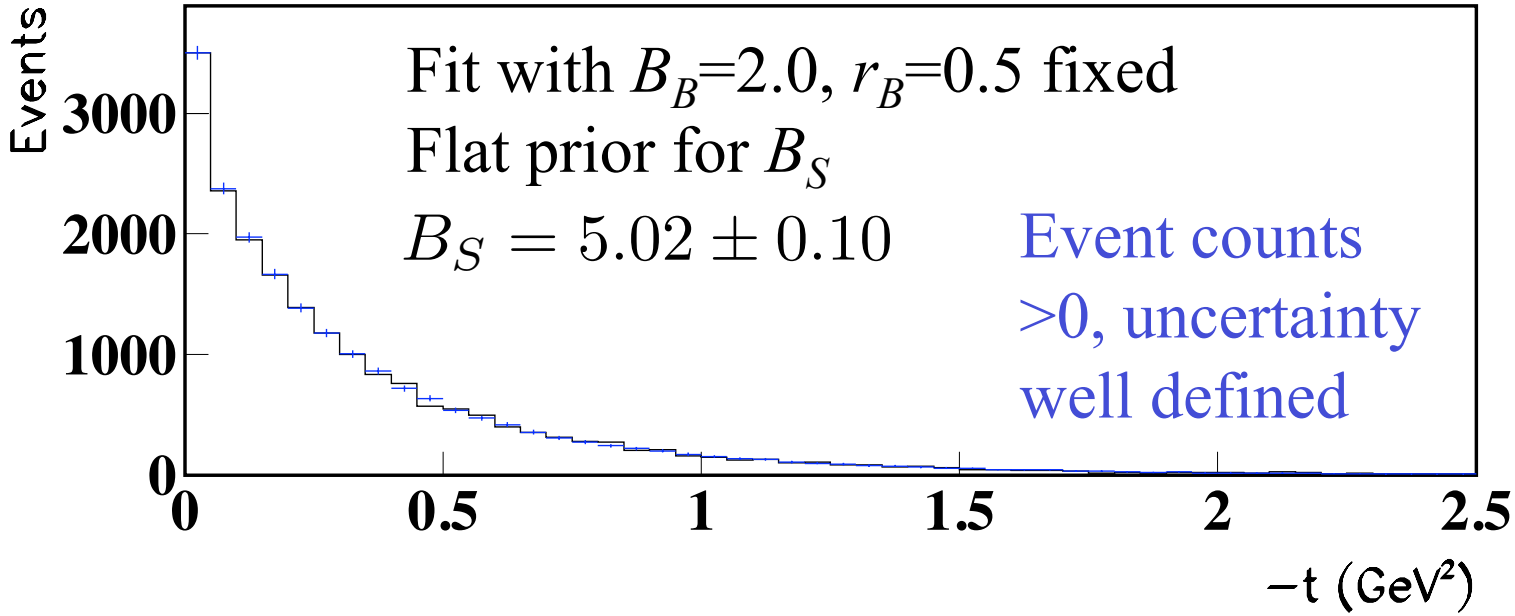
$$w_j = \frac{N_{data}}{N_{signal}} \frac{B_S}{B_{S,0}} \exp((B_S - B_{S,0}) t_{j,gen})$$

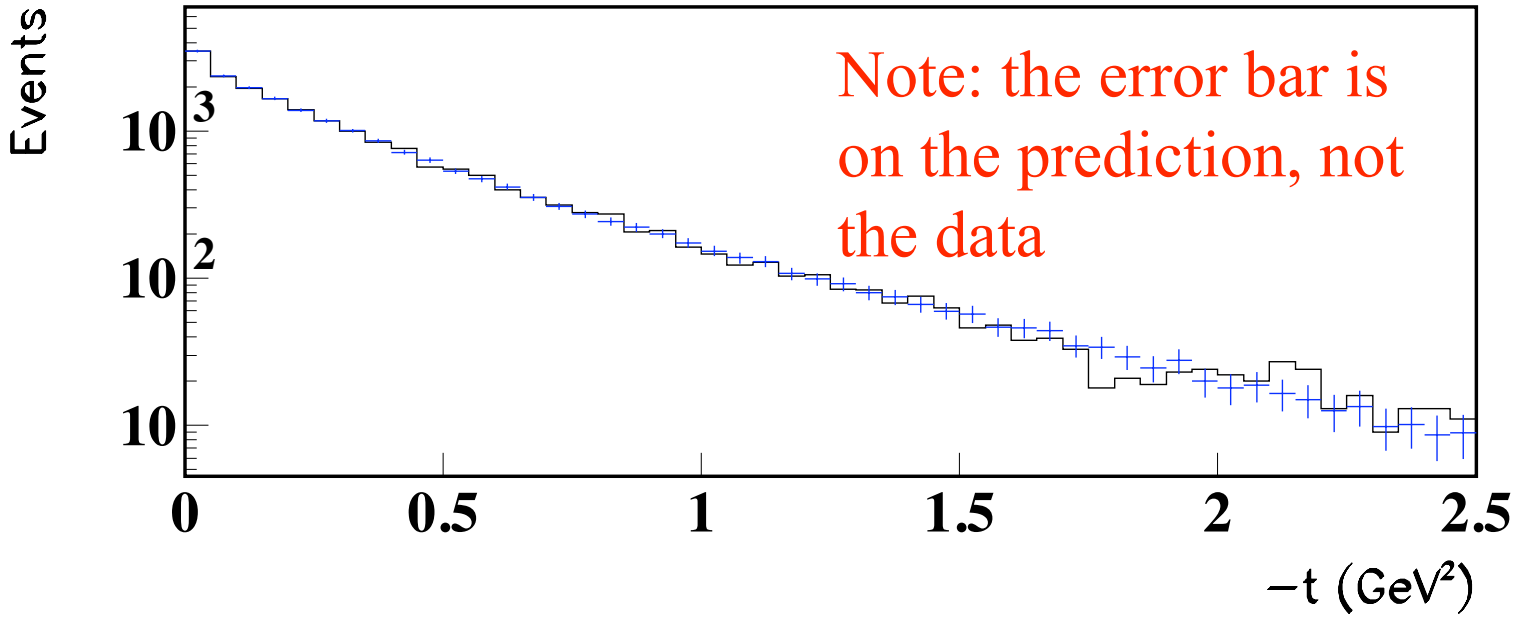
Exercise: show that this reweighting generates the correct t distribution

$B_{S,0}$ The value of B used in the generation

$t_{j,gen}$ The true *t* value for event *j*

Results of fit





Systematic uncertainties

Treatment of systematic uncertainties is straightforward. E.g., suppose we do not have perfect knowledge of the background fraction and background normalization. This can be built into the prior:

$$P_0(B_S, B_B, r_B) = P_0(B_S)P_0(B_B)P_0(r_B)$$

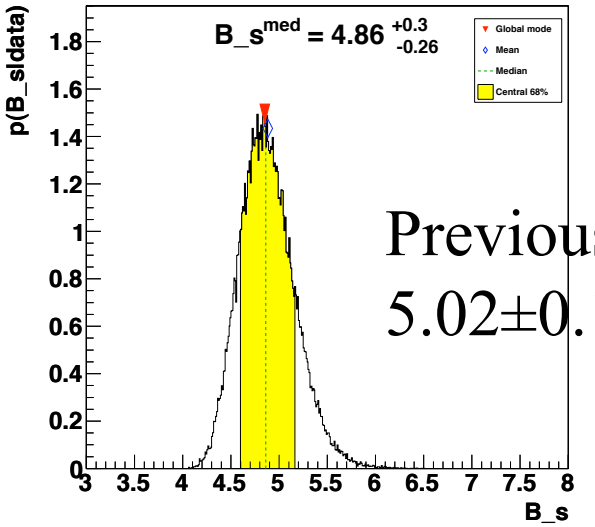
$$P_0(B_S) \sim \mathcal{N}(6, 2)$$

$$P_0(B_B) \sim \mathcal{N}(1.5, 1)$$

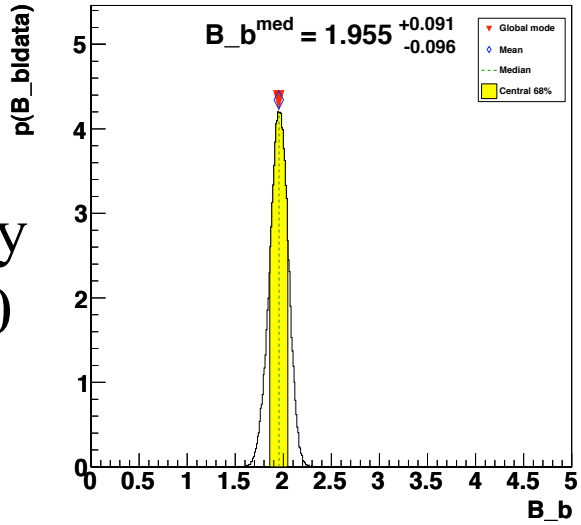
$$P_0(B_S) \sim \mathcal{N}(0.4, 0.2)$$

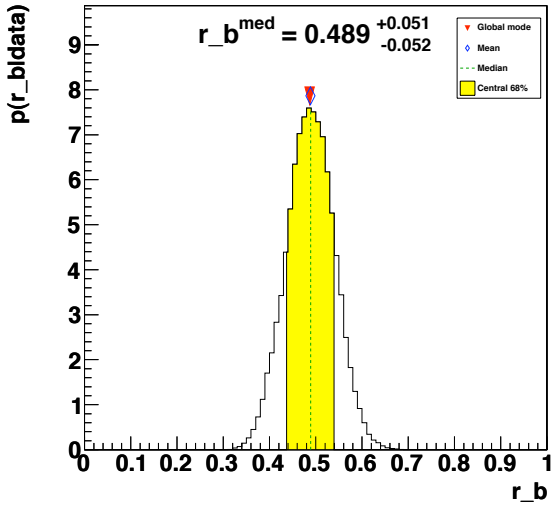
Signal thought to be around B_S=2, background around B_B=1.5, background fraction around 0.4

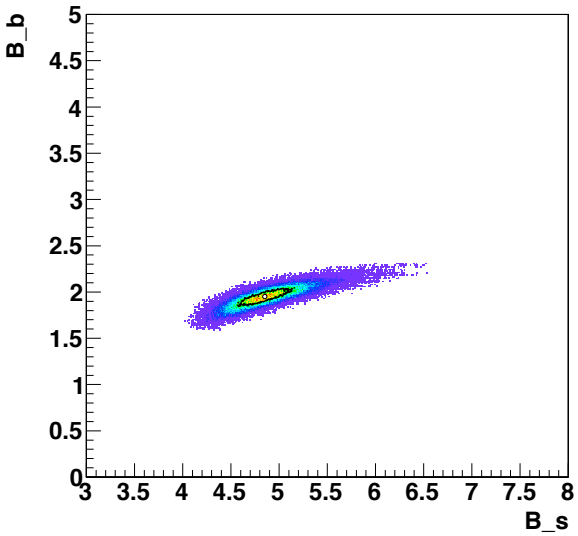
Instead of shifting each quantity by 1-sigma as in the bin-by-bin, we now sample directly from a Gaussian distribution. Systematics are included as nuisance parameters in the analysis.

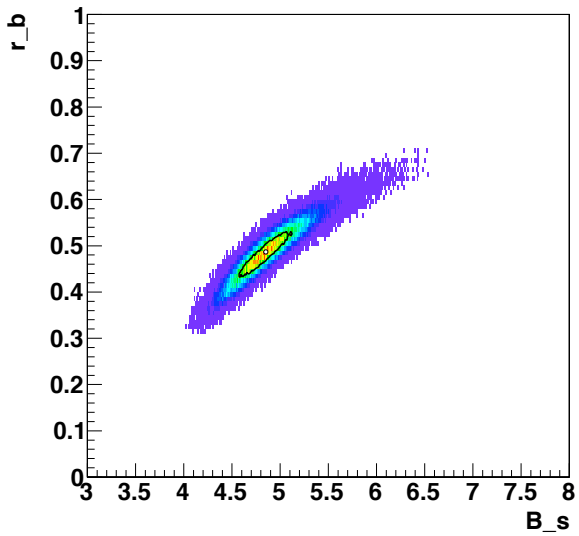


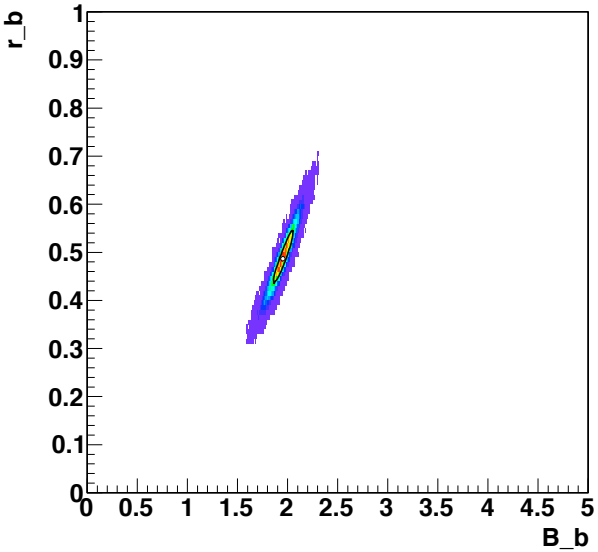
Previously
5.02±0.10



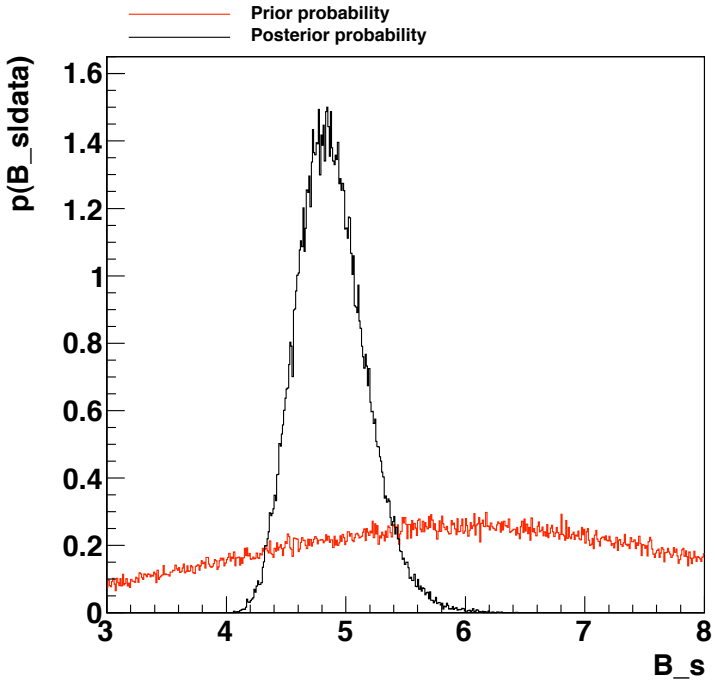


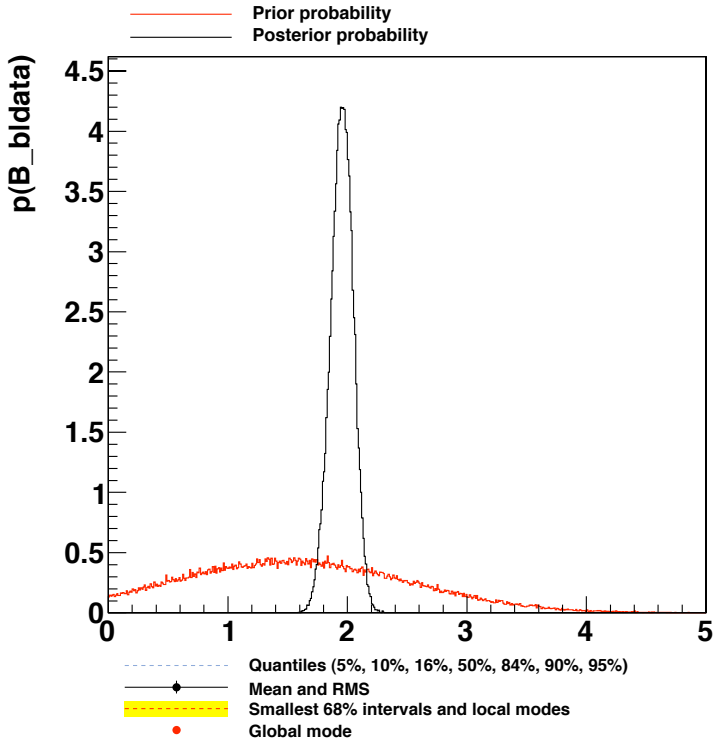




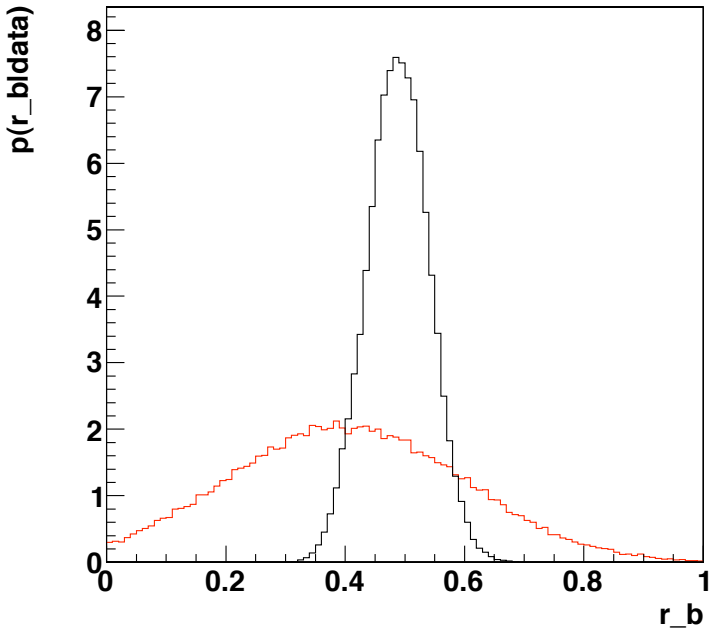


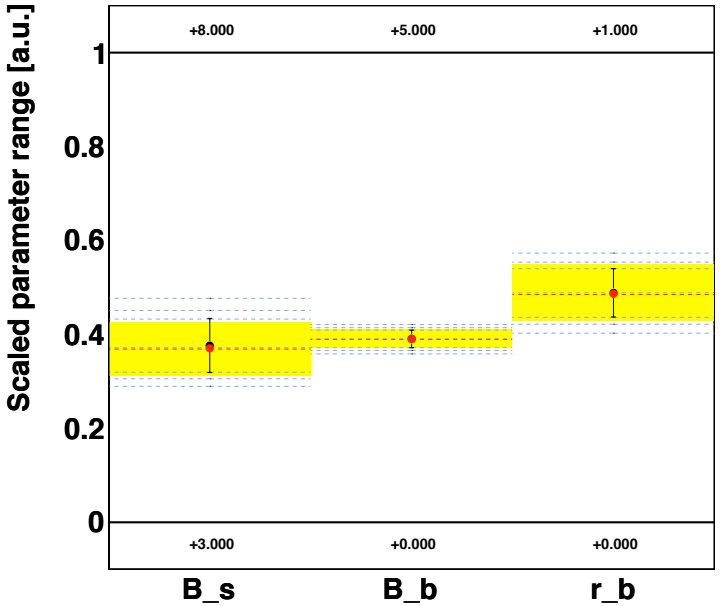
Standard BAT output





Standard BAT output: knowledge update using the data.





No separation of statistical and systematic uncertainties !

Discussion/conclusion

We have seen that we get very good results w/o unfolding. Also, the uncertainties are well-defined.

How would we report the result ?

1. Report $P(B_S|D)$, keep data analysis code and data around for future use. The paper would contain the observed event counts with the expectations from the model with best parameters, as well as the pdf for B_S.
2. Unfold the data once you have the best parameters values, report cross sections which others can fit. How to define the uncertainties ?