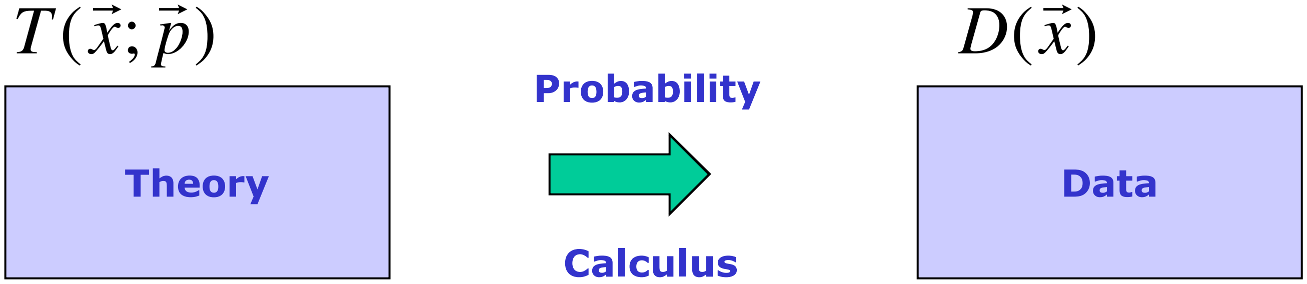
Parameter estimation
χ^2 and likelihood

- — Introduction to estimation
- — Properties of χ^2, ML estimators
- — Measuring and interpreting Goodness-Of-Fit
- — Numerical issues in fitting
- — Understanding MINUIT
- — Mitigating fit stability problems
- — Bounding fit parameters
- — Simultaneous fitting
- — Multidimensional fitting
- — Fit validation studies
  - — Fit validity issues at low statistics
- — Toy Monte Carlo techniques

Wouter Verkerke, UCSB

Parameter estimation – Introduction



$T(\vec{x};\vec{p})$

Theory

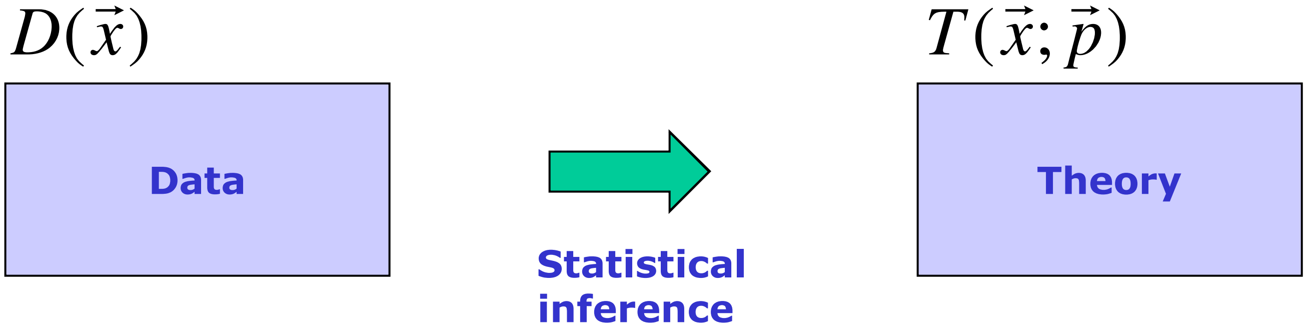
Probability

Calculus

$D(\vec{x})$

Data

- Given the theoretical distribution parameters p, what can we say about the data



$D(\vec{x})$

Data

Statistical inference

$T(\vec{x};\vec{p})$

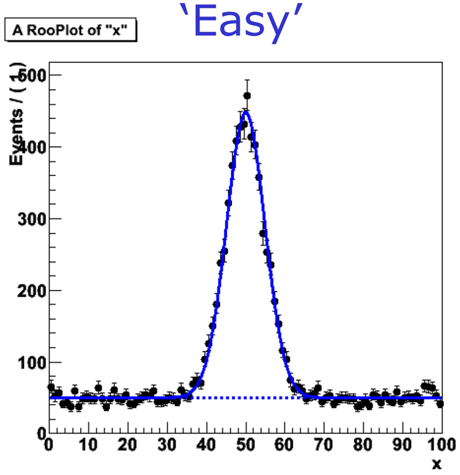
Theory

- **Need a procedure to estimate *p* from *D***

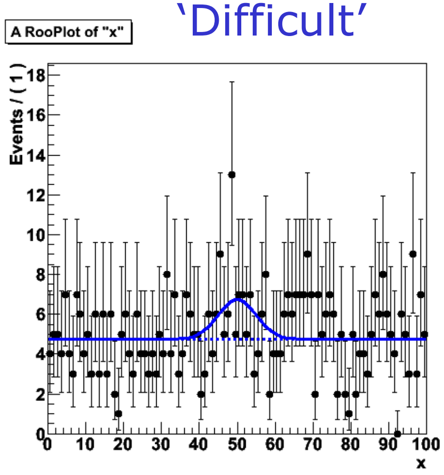
Multiple methods

- Many ways to infer information on model (parameter) from data
  - χ^2 fit → $p = 5.2 \pm 0.3$
  - Likelihood fit → $p = 4.7 \pm 0.4$
  - Bayesian interval → $p \in [\ 4.5 - 5.9\ ]$ at 68% credibility
  - Frequentist interval → $p \in [\ 4.4 - 5.8\ ]$ at 68% confidence level
- When data is abundant, methods usually give consistent answers
- Issues and differences between methods arise when experimental result contains little information

'Easy'



'Difficult'



Multiple methods

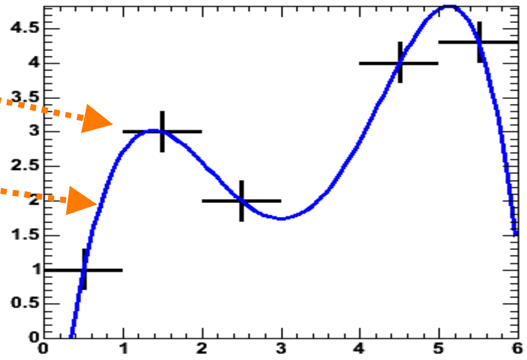
- Many ways to infer information on model (parameter) from data
  - χ^2 fit → $p = 5.2 \pm 0.3$
  - Likelihood fit → $p = 4.7 \pm 0.4$
  - Bayesian interval → $p \in [\ 4.5 - 5.9\ ]$ at 68% credibility
  - Frequentist interval → $p \in [\ 4.4 - 5.8\ ]$ at 68% confidence level

- Will first focus χ^2 and likelihood estimation procedures
  - Well known, often used
  - Explore assumptions, limitations

- In the next module focus on interpreting experiments with little information content

A well known estimator – the χ^2 fit

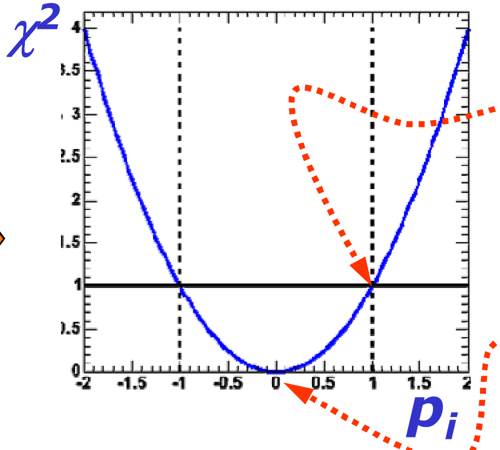
- Given a set of points $\{(\vec{x}_i, y_i, \sigma_i)\}$ and a function *f(**x**,**p**)* define the χ^2



$$\chi^2(\vec{p}) = \sum_i \frac{(y_i - f(\vec{x};\vec{p}))^2}{\sigma_y^2}$$

- Estimate parameters by minimizing the χ^2(p) with respect to all parameters p_i
  - *In practice, look for*

$$\frac{d\chi^2(p_i)}{dp_i} = 0$$



Error on $\boldsymbol{p_i}$ is given by χ2 variation of +1

Value of $\boldsymbol{p_i}$ at minimum is estimate for $\boldsymbol{p_i}$

- Well known: but why does it work? Is it always right? Does it always give the best possible error?

Basics – What is an estimator?

- An ***estimator*** is a ***procedure*** giving a value for a parameter or a property of a distribution as a function of the actual data values, i.e.

$$\hat{\mu}(x) = \frac{1}{N}\sum_i x_i$$

← **Estimator of the mean**

$$\hat{V}(x) = \frac{1}{N}\sum_i (x_i - \vec{\mu})^2$$

← **Estimator of the variance**

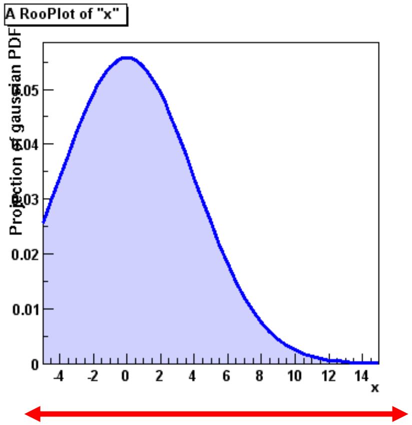
- A perfect estimator is
  - Consistent: $\lim_{n\to\infty}(\hat{a}) = a$
  - Unbiased – *With finite statistics you get the right answer on average*
  - Efficient $V(\hat{a}) = \left\langle (\hat{a} - \langle \hat{a} \rangle)^2 \right\rangle$ ← This is called the **Minimum Variance Bound**
  - ***There are no perfect estimators!***

How to model your data

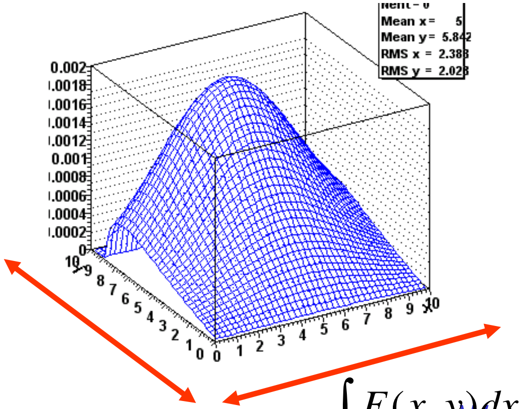
- Approach in χ^2 fit very empirical – Function *f*(*x*,*y*) can be any arbitrary function
- Many techniques (Likelihood, Bayesian, Frequentist) require a more formal approach to data modeling through probability density functions
- We can characterize data distributions with *probability density functions* F(x;p)
  - **x** = observables (measured quantities)
  - **p** = parameters (model/theory parameters)
- Properties
  - Normalized to unity with respect to observable(s) x
  - Positive definite – F(x;p)>=0 for all (x,p)

$$\int F(\vec{x};\vec{p})d\vec{x} \equiv 1$$

$$F(\vec{x};\vec{p}) \geq 0$$



$$\int F(x)dx \equiv 1$$



$$\int F(x,y)dxdy \equiv 1$$

Probability density functions

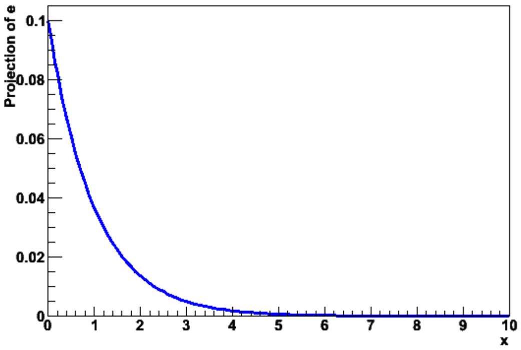
- Properties
  - Parameters can be physics quantities of interest (life time, mass)

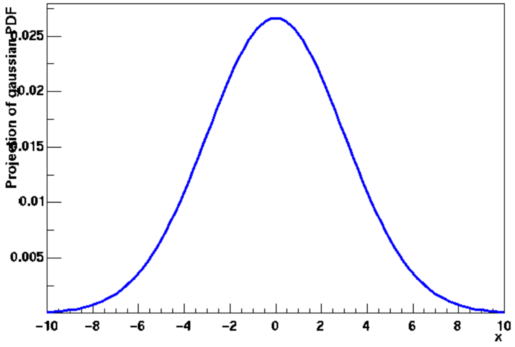
Decay time distribution
observable x (decay time)
parameter θ (lifetime)

$$f(x;\theta) = \frac{1}{\theta} e^{-x/\theta}$$

Invariant mass distribution
observable x (inv. mass)
parameter m (physics mass)
parameter σ (decay width)

$$f(x;m) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2} \left(\frac{x-m}{\sigma} \right)^2 \right)$$





  - Vehicle to infer physics parameters from data distributions

Likelihood

- The *likelihood* is the value of a probability density function evaluated at the measured value of the observable(s)
  - Note that likelihood is only function of parameters, not of observables

$$L(\vec{p}) \quad = \quad F(\vec{x} \equiv \vec{x}_{data}; \vec{p})$$

- For a dataset that consists of multiple data points, the product is taken

$$L(\vec{p}) = \prod_i F(\vec{x}_i; \vec{p}), \quad \text{i.e.} \quad L(\vec{p}) = F(x_0; \vec{p}) \cdot F(x_1; \vec{p}) \cdot F(x_2; \vec{p}) \ldots$$

Probability, Probability Density, and Likelihood

- Poisson *probability* $P(n|\mu) = \mu^n \exp(-\mu)/n!$
- Gaussian *probability density function (pdf)* $p(x|\mu,\sigma)$: $p(x|\mu,\sigma)dx$ is differential of probability dP.
- In Poisson case, suppose n=3 is observed. Substituting n=3 into $P(n|\mu)$ yields the *likelihood function* $L(\mu) = \mu^3 \exp(-\mu)/3!$
  - Key point is that $L(\mu)$ is *not* a probability density in μ. (It is not a density!)
  - Area under L is meaningless. That's why a new word, "likelihood", was invented for this function of the parameter(s), to distinguish from a pdf in the observable(s)! Many people nevertheless talk about 'integrating the likelihood' → confusion about what is done in Bayesian interval (more later)
  - Likelihood Ratios $L(\mu1) /L(\mu2)$ are useful and frequently used.

Change of variable x, change of parameter θ

- For pdf $p(x|\theta)$ and (1-to-1) change of variable from x to $y(x)$:

$$p(y(x)|\theta) = p(x|\theta) \ / \ |dy/dx|.$$

- Jacobian modifies probability *density, guaranties that*

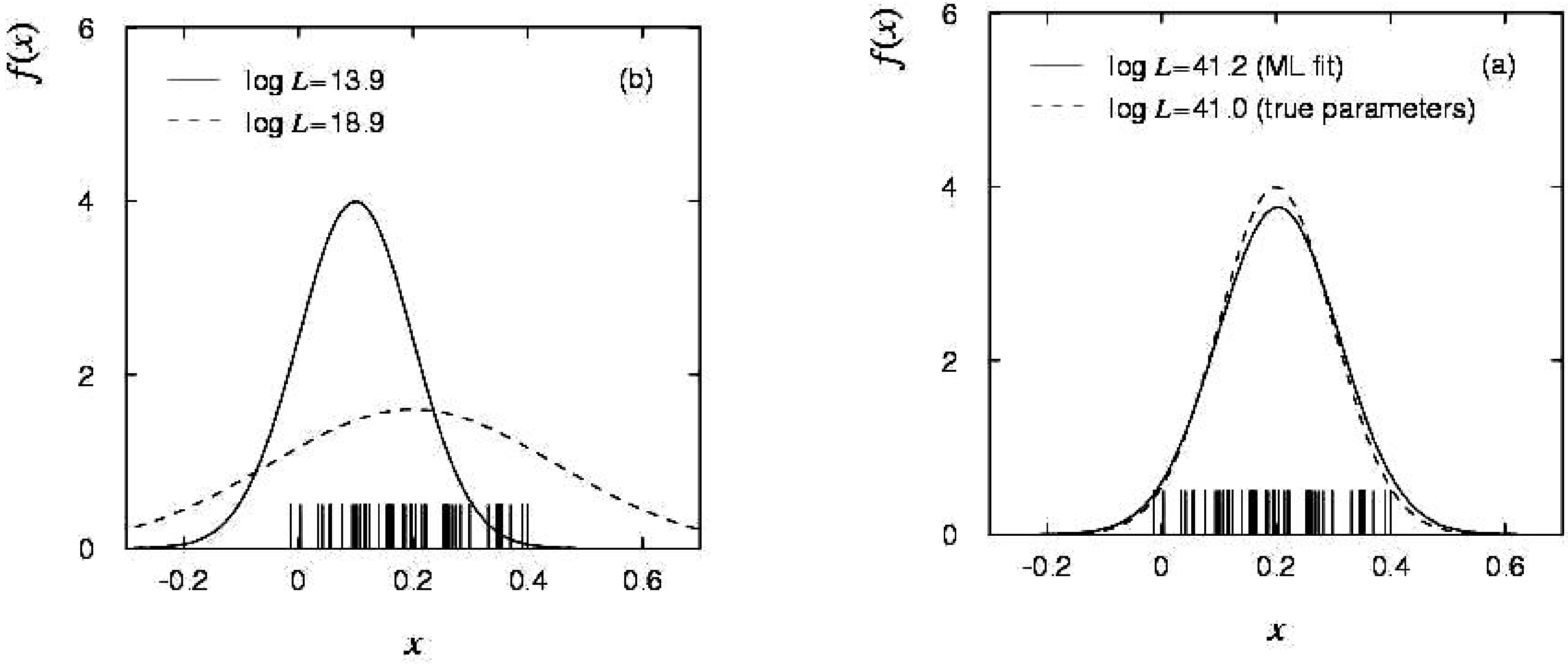
$$P(\ y(x_1) < y < y(x_2)\ ) = P(x_1 < x < x_2), \text{ i.e., that}$$

- *Probabilities are invariant under change of variable x.*
  - Mode of probability *density* is *not* invariant (so, e.g., criterion of maximum probability density is ill-defined).
  - Likelihood *ratio* is invariant under change of variable x. (Jacobian in denominator cancels that in numerator).

- For likelihood $L(\theta)$ and reparametrization from θ to $u(\theta)$: $L(\theta) = L(u(\theta))$ (!).
  - Likelihood $L(\theta)$ is invariant under reparametrization of parameter θ (reinforcing fact that L is *not* a pdf in θ*).*

Parameter estimation using Maximum Likelihood

- Likelihood is high for values of ***p*** that result in distribution similar to data



- Define the maximum likelihood (ML) estimator(s) to be the parameter value(s) for which the likelihood is maximum.

Parameter estimation – Maximum likelihood

- Computational issues
  - For convenience the negative log of the Likelihood is often used as addition is numerically easier than multiplication

$$-\ln L(\vec{p}) = -\sum_i \ln F(\vec{x}_i;\vec{p})$$

  - Maximizing L(p) equivalent to minimizing –log L(p)

- In practice, find point where derivative is zero

$$\left.\frac{d\ln L(\vec{p})}{d\vec{p}}\right|_{p_i=\hat{p}_i} = 0$$

Variance on ML parameter estimates

- The ML **estimator** for the **parameter variance** is

$$\hat{\sigma}(p)^2 = \hat{V}(p) = \left(\frac{d^2 \ln L}{d^2 p} \right)^{-1}$$

  - I.e. variance is estimated from 2nd derivative of –log(L) at minimum
  - Valid if estimator is **efficient** and **unbiased**!

From Rao-Cramer-Frechet inequality

$$V(\hat{p}) \geq 1 + \frac{db}{dp} \Big/ \left(\frac{d^2 \ln L}{d^2 p} \right)$$

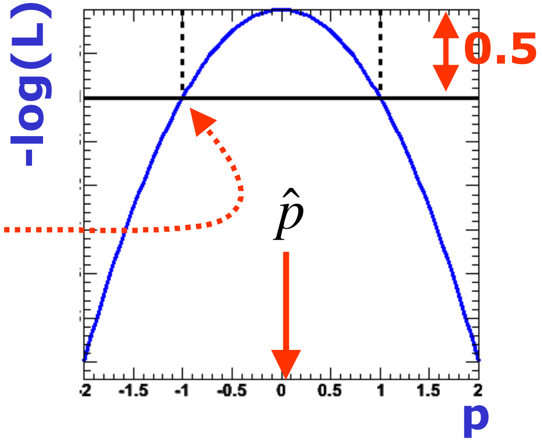
b = bias as function of p, inequality becomes equality in limit of efficient estimator

- **Visual interpretation** of variance estimate
  - Taylor expand –log(L) around minimum

$$\ln L(p) = \ln L(\hat{p}) + \left. \frac{d \ln L}{dp} \right|_{p=\hat{p}} (p-\hat{p}) + \tfrac{1}{2} \left. \frac{d^2 \ln L}{d^2 p} \right|_{p=\hat{p}} (p-\hat{p})^2$$

$$= \ln L_{\max} + \left. \frac{d^2 \ln L}{d^2 p} \right|_{p=\hat{p}} \frac{(p-\hat{p})^2}{2}$$

$$= \ln L_{\max} + \frac{(p-\hat{p})^2}{2\hat{\sigma}_p^2} \quad \Rightarrow \quad \ln L(p \pm \sigma) = \ln L_{\max} - \frac{1}{2}$$



Properties of Maximum Likelihood estimators

- In general, Maximum Likelihood estimators are
  - **Consistent** (gives right answer for N→∞)
  - **Mostly unbiased** (bias ∝1/N, may need to worry at small N)
  - **Efficient for large N** (you get the smallest possible error)
  - **Invariant**: (a transformation of parameters will Not change your answer, e.g $(\hat{p})^2 = \widehat{(p^2)}$

Use of 2nd derivative of –log(L) for variance estimate is usually OK

- MLE efficiency theorem: the MLE will be unbiased and efficient if an unbiased efficient estimator exists
  - Proof not discussed here
  - Of course this does not guarantee that any MLE is unbiased and efficient for any given problem

More about maximum likelihood estimation

- It’s not ‘right’ it is just sensible

- It does not give you the ‘most likely value of p’ –
  it gives you *the value of p for which this data is most likely*

- Numeric methods are often needed to find
  the maximum of ln(L)
  - Especially difficult if there is >1 parameter
  - Standard tool in HEP: MINUIT (more about this later)

- Max. Likelihood does **not** give you a goodness-of-fit measure
  - If assumed F(x;p) is not capable of describing your data for any p,
    the procedure will not complain
  - The absolute value of L tells you nothing!

Relation between Likelihood and χ^2 estimators

- Properties of χ^2 estimator follow from properties of ML estimator using *Gaussian probability density functions*

$$F(x_i, y_i, \sigma_i; \vec{p}) = \exp\left[-\left(\frac{y_i - f(x_i;\vec{p})}{\sigma_i}\right)^2\right]$$

Probability Density Function in p for single data point $x_i(\sigma_i)$ and function $f(x_i;p)$

Take log,
Sum over all points ($\boldsymbol{x_i}$,$\boldsymbol{y_i}$,σ_i)

$$\ln L(\vec{p}) = -\tfrac{1}{2}\sum_i \left(\frac{y_i - f(x_i;\vec{p})}{\sigma_i}\right) = -\tfrac{1}{2}\chi^2$$

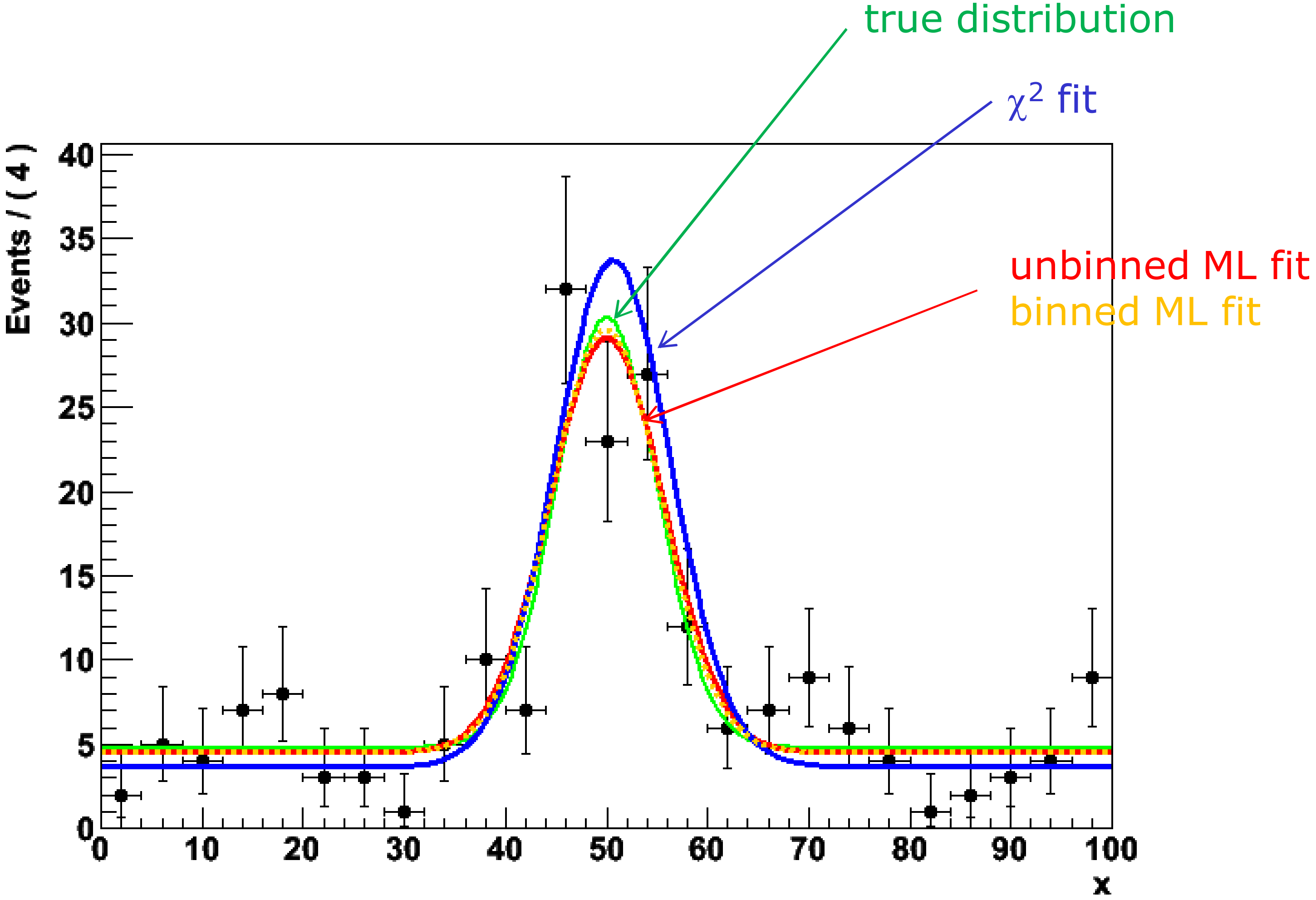
The Likelihood function in p for given points $x_i(\sigma_i)$ and function $f(x_i;p)$

- The χ^2 estimator follows from ML estimator, i.e it is
  - **Efficient, consistent, bias 1/N, invariant**,
  - **But only in the limit that the error on *x_i* is truly Gaussian**
  - i.e. need n_i > 10 if y_i follows a Poisson distribution

- Bonus: Goodness-of-fit measure – $\chi^2 \approx 1$ per d.o.f

Example of χ^2 vs ML fit

- Example with many low statistics bins



true distribution

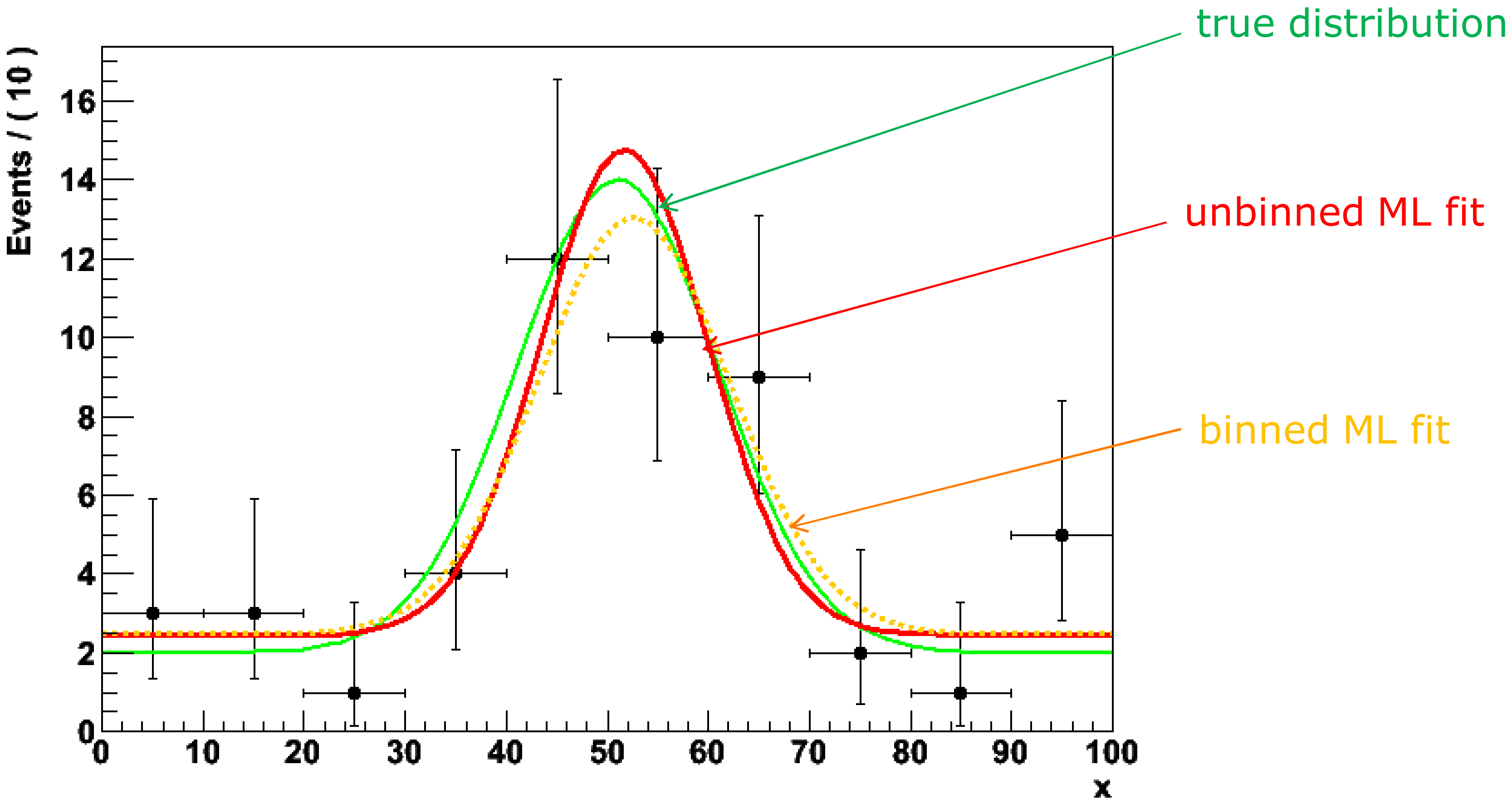
χ^2 fit

unbinned ML fit

binned ML fit

Example of binned vs unbinned ML fit

- Lowering number of bins and number of events…



true distribution

unbinned ML fit

binned ML fit

- Proper way to study bias, precision is with toy MC study → at the end of this module

Wouter Verkerke, NIKHEF

Maximum Likelihood or χ^2 – What should you use?

- χ^2 fit is fastest, easiest
  - Works fine at high statistics
  - Gives absolute goodness-of-fit indication
  - Make (incorrect) Gaussian error assumption on low statistics bins
  - Has bias proportional to 1/N
  - Misses information with feature size < bin size

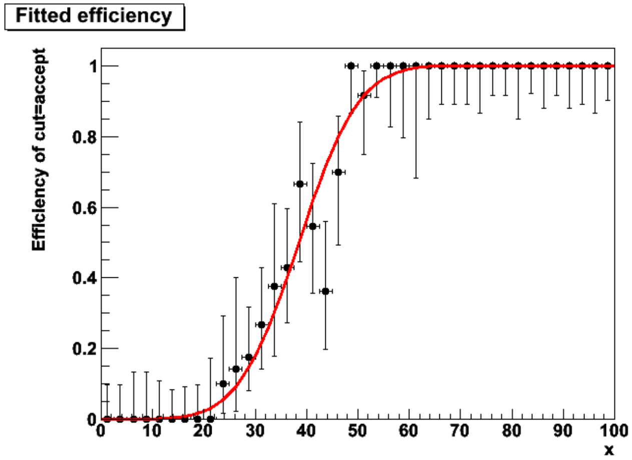
- Full Maximum Likelihood estimators most robust
  - No Gaussian assumption made at low statistics
  - No information lost due to binning
  - Gives best error of all methods (especially at low statistics)
  - No intrinsic goodness-of-fit measure, i.e. no way to tell if 'best' is actually 'pretty bad'
  - Has bias proportional to 1/N
  - Can be computationally expensive for large N

- Binned Maximum Likelihood in between
  - Much faster than full Maximum Likihood
  - Correct Poisson treatment of low statistics bins
  - Misses information with feature size < bin size
  - Has bias proportional to 1/N

$$-\ln L(p)_{\text{binned}} = \sum_{\text{bins}} n_{\text{bin}} \ln F(\vec{x}_{\text{bin-center}};\vec{p})$$

You can (almost) always avoid χ^2 fits

- Case study: Fit for efficiency function
  - Have some simulation sample: need to parameterize which fraction of events passes as function of observable x



- 'Traditional χ^2 approach'
  - Make histogram of Npassed/Ntotal
  - Fit parameterized efficiency function to histogram
  - Tricky question: what errors to use? √N is wrong.

    Can use binomial errors $V(r) = np(1-p) \quad \Rightarrow \quad \sigma = \sqrt{np(1-p)}$

    However still quite approximate: true errors will be asymmetric (i.e. no upward error on bin with Npass=10, Ntotal=10)

You can (almost) always avoid χ^2 fits

- MLE approach
  - Realize that your dataset has two observables (x,c), where c is a discrete observable with states ‘accept’ and ‘reject’
  - Corresponding probability density function:

$$
\begin{aligned}
F(c\,|\,x,\vec{p}) = \quad & \theta(c = accept)\cdot\varepsilon(x,\vec{p}) + \\
& \theta(c = reject)(1-\varepsilon(x,\vec{p}))
\end{aligned}
$$

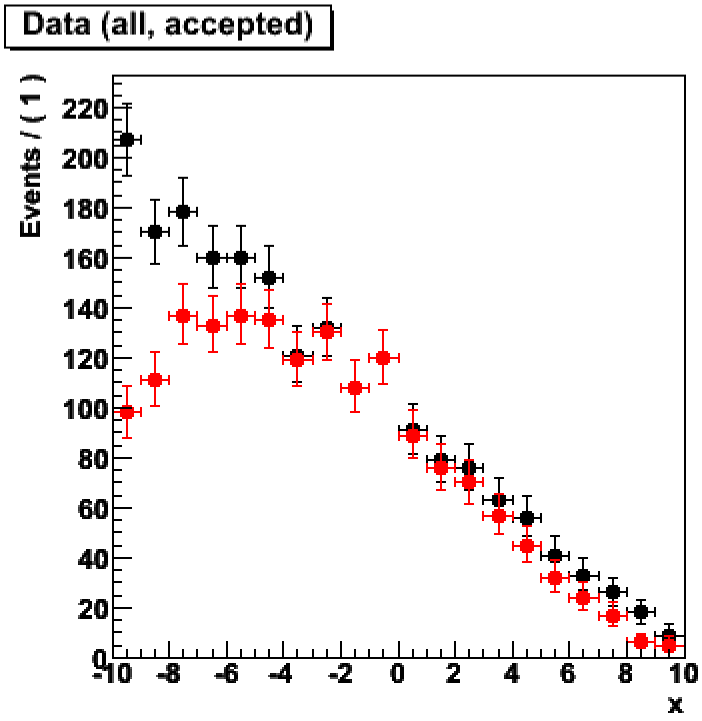
  - Clearly unit-normalized over c for each value of (x,p) (ε must be between 0 and 1 for all (x,p))
  - Write –log(L) as usual, using above p.d.f. and minimize

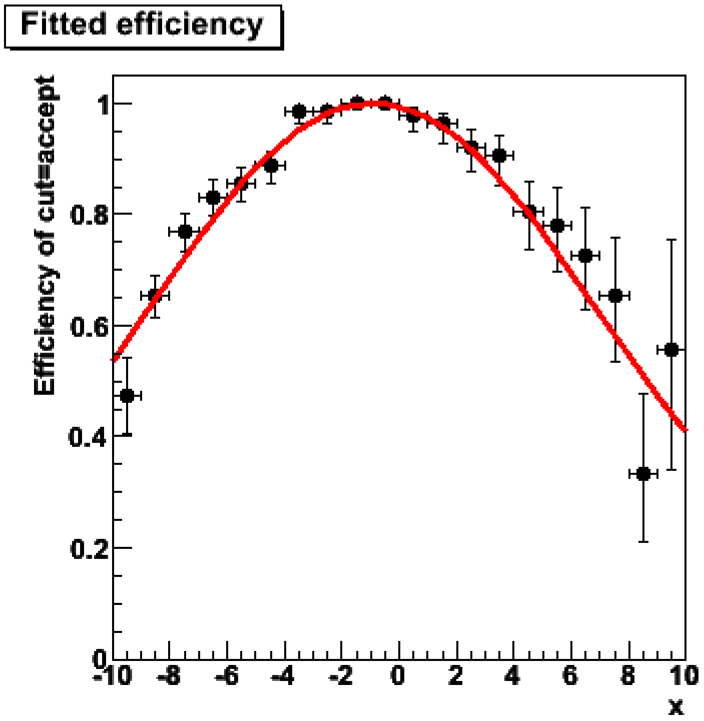
$$
-\ln L(\vec{p}) = -\sum_i \ln F(x_i, c_i; \vec{p})
$$

  - Result: estimation of e(x,p) using correct binomial/poisson assumption on distribution of observables.
  - Fit can also be performed unbinned

You can (almost) always avoid χ^2 fits

- Example of unbinned MLE fit for efficiency





Weighted data

- Sometimes input data is weighted
- Examples:
  - Certain Next-to-leading order event generator for LHC physics produce simulated events with weights +1 and -1.
  - You've subtracted a distribution of background events from a sideband in data (also results in events with weight +1 and -1)
  - You work with reweighted data samples for a variety of reasons (e.g. not enough data was available for one background sample, rescale available events with some non-unit weight to match available amounts of other samples)
- How to deal with event weights in χ^2, MLE parameter estimation
- χ^2 fit of histograms with weighted data are straightforward

From C.L.T

$$y_i = \sum_i w_i$$

From C.L.T

$$\chi^2 = \sum_i \left(\frac{y_i - f(\vec{x}_i;\vec{p})}{\sigma_i} \right)^2$$

$$\sigma_i = \sqrt{1 \Big/ \sum_i w_i^2}$$

  - NB: You may no longer be able to interpret $\hat{\sigma}(p) \equiv \sqrt{\hat{V}(p)}$ as a Gaussian error (i.e. 68% contained in 1σ)

Weighted data – χ^2 vs MLE

- Adding event weights to –log(L) straightforward, but does not yield correct estimates on parameter variance

$$-\ln L(\vec{p})_{\text{weighted}} = -\sum_i \overset{\text{Event weight}}{w_i} \ln F(\vec{x}_i;\vec{p})$$

  - **Variance estimate on parameters will be proportional to** $\sum_i w_i$
  - If $\sum_i w_i < N$ errors will be too small, if $\sum_i w_i > N$ errors will be too large!

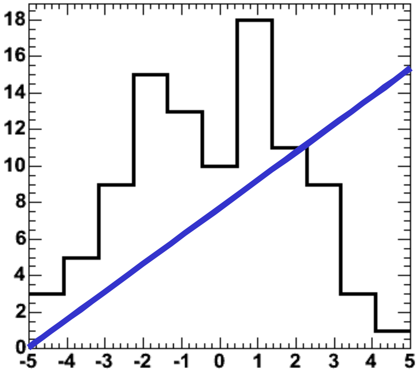
- No clean solution available that retains all good properties of MLE, but it is possible to perform sum-of-weights-like correction to covariance matrix to correct for effect of on-unit weights

$$V' = VC^{-1}V$$

  - where V is the cov. matrix calculated from a –log(L) with event weights *w*, and C is the cov. matrix calculated from a –log(L) with event weights ***w²***
  - It is easy to see that in the case of 1 parameter this is equivalent to $\sigma_i = \sqrt{1 \big/ \sum_i w_i^2}$

Hypothesis testing – Goodness of fit

- Hypothesis testing and goodness-of-fit
  - Reminder:
    classical hypothesis test compares data to two hypothesis H_0 and H_1 (e.g background-only vs signal+background).
    Type-I error = claiming signal when you should *not* have
    Type-II error = *not* claiming signal when you should have
  - If there is no alternate (H0) hypothesis, hypothesis test is called 'goodness-of-fit' test. NB: Can only quantify Type-I error thus question "which g.o.f. test is *best*" (e.g. χ^2, Kolmogorov) is ill posed



'Not a good fit'

Estimating and interpreting Goodness-Of-Fit

- Most common test: **the χ^2 test**

$$\chi^2 = \sum_i \left(\frac{y_i - f(\vec{x}_i; \vec{p})}{\sigma_i} \right)^2$$

  - If f(x) describes data then $\chi^2 \approx$ N, if χ^2 >> N something is wrong

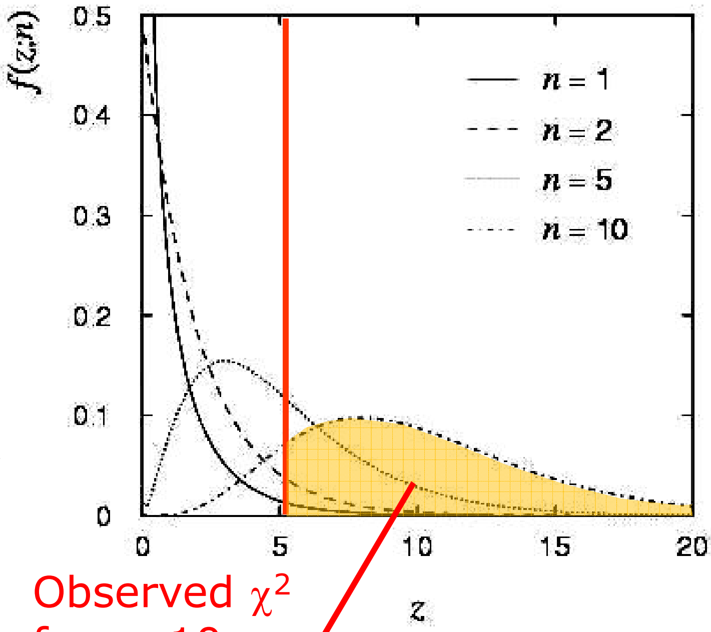
- How to quantify meaning of 'large χ^2'?
  - What you really want to know: the probability that a function which does genuinely describe the data on N points would give a χ^2 probability as large or larger than the one you already have.
  - For large N, sqrt(2χ^2) has a Gaussian distribution with mean sqrt(2N-1) and σ=1 → 'Easy'
  - How to make a well calibrated statement for intermediate N

How to quantify meaning of ‘large χ^2’

- Probability distr. for χ^2 is given by

$$\chi^2 = \sum_i \left(\frac{y_i - \mu_i}{\sigma_i} \right)^2$$

$$p(\chi^2, N) = \frac{2^{-N/2}}{\Gamma(N/2)} \chi^{N-2} e^{-\chi^2/2}$$



Observed χ^2 for n=10

P = integral over shaded area

$$P(\chi^2; N) = \int_{\chi^2}^{\infty} p(\chi^{2\prime}; N)\, d\chi^{2\prime}$$

- Good news: Integral of χ^2 pdf is analytically calculable!

Goodness-of-fit – χ^2

- Example for χ^2 probability
  - Suppose you have a function **f(x;p)** which gives a χ^2 of 20 for 5 points (histogram bins).
  - Not impossible that **f(x;p)** describes data correctly, just unlikely
  - How unlikely? $$\int_{20}^{\infty} p(\chi^2, 5)\, d\chi^2 = 0.0012$$
- Note: If function has been fitted to the data
  - Then you need to account for the fact that parameters have been adjusted to describe the data

$$N_{\mathrm{d.o.f.}} = N_{\mathrm{data}} - N_{\mathrm{params}}$$

- Practical tips
  - To calculate the probability in ROOT '`TMath::Prob(chi2,ndf)`'

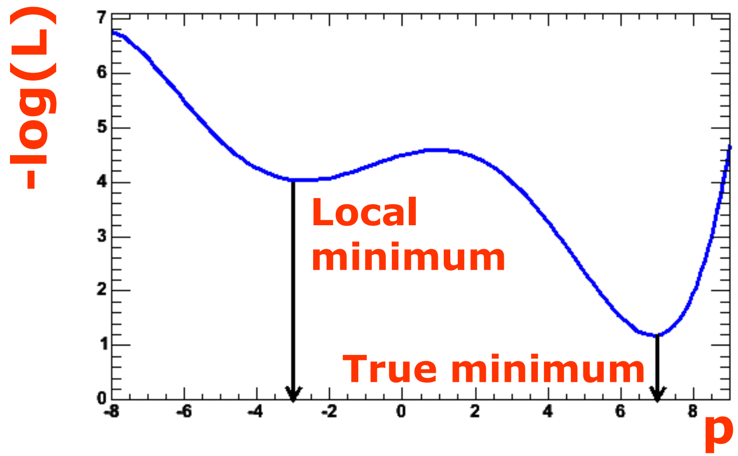
Practical estimation – Numeric χ^2 and -log(L) minimization

- For most data analysis problems minimization of χ^2 or –log(L) cannot be performed analytically
  - Need to rely on numeric/computational methods
  - In >1 dimension generally a difficult problem!

- But no need to worry – Software exists to solve this problem for you:
  - Function minimization workhorse in HEP many years: MINUIT
  - MINUIT does function minimization and error analysis
  - It is used in the PAW,ROOT fitting interfaces behind the scenes
  - It produces a lot of useful information, that is sometimes overlooked
  - Will look in a bit more detail into MINUIT output and functionality next

Numeric χ^2/-log(L) minimization – Proper starting values

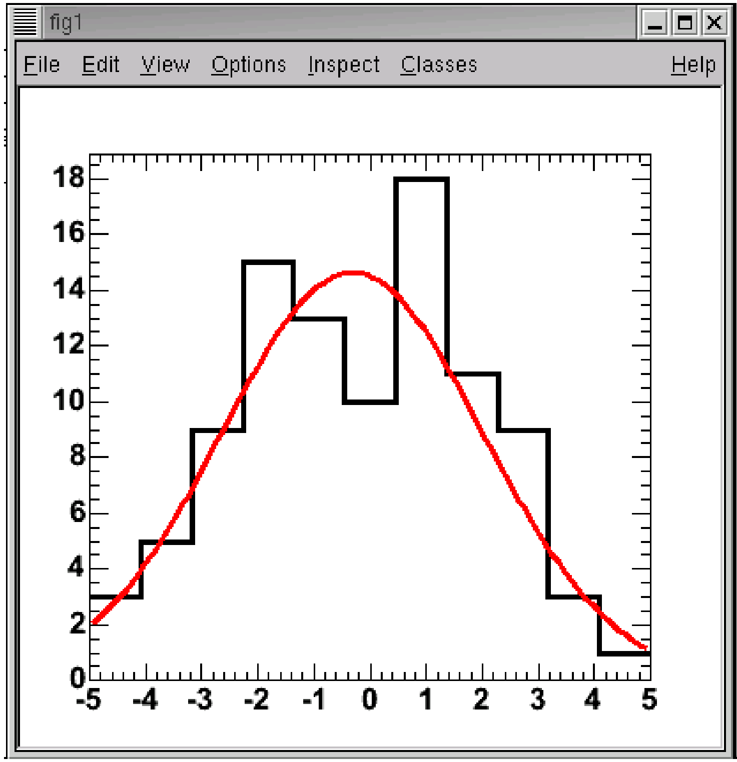
- For all but the most trivial scenarios it is not possible to automatically find reasonable starting values of parameters
  - This may come as a disappointment to some…
  - So you need to supply good starting values for your parameters

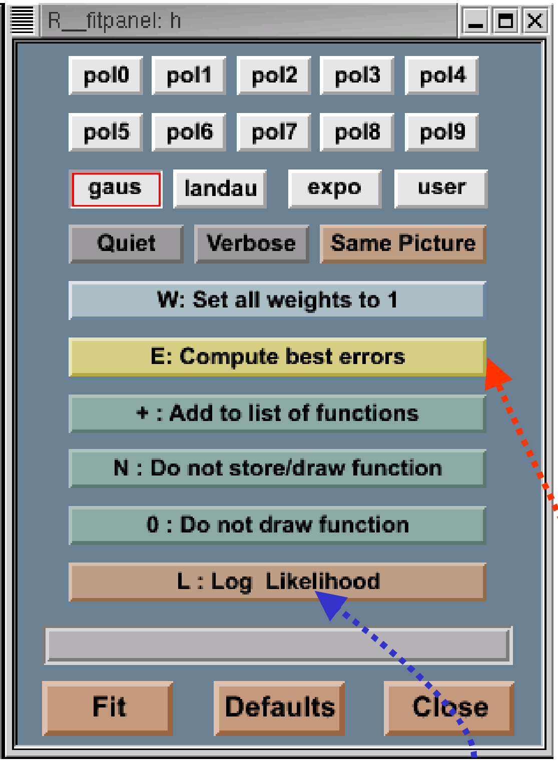


Reason: There may exist multiple (local) minima in the likelihood or χ^2

  - Supplying good initial uncertainties on your parameters helps too
  - Reason: Too large error will result in MINUIT coarsely scanning a wide region of parameter space. It may accidentally find a far away local minimum

Example of interactive fit in ROOT





- What happens in MINUIT behind the scenes

   1) Find minimum in –log(L) or χ^2 – MINUIT function MIGRAD

   2) Calculate errors on parameters – MINUIT function HESSE

   3) Optionally do more robust error estimate – MINUIT function MINOS

Wouter Verkerke, UCSB

Minuit function MIGRAD

- Purpose: find minimum

```
**********
**    13 **MIGRAD        1000              1
**********
(some output omitted)
MIGRAD MINIMIZATION HAS CONVERGED.
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                     EDM=2.36773e-06     STRATEGY= 1       ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP          FIRST
 NO.   NAME      VALUE            ERROR          SIZE       DERIVATIVE
  1   mean        8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
  2   sigma       3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                              ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25     NPAR=  2    ERR DEF=0.5
 1.049e-01  3.338e-04
 3.338e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1       2
       1  0.00430   1.000   0.004
       2  0.00430   0.004   1.000
```

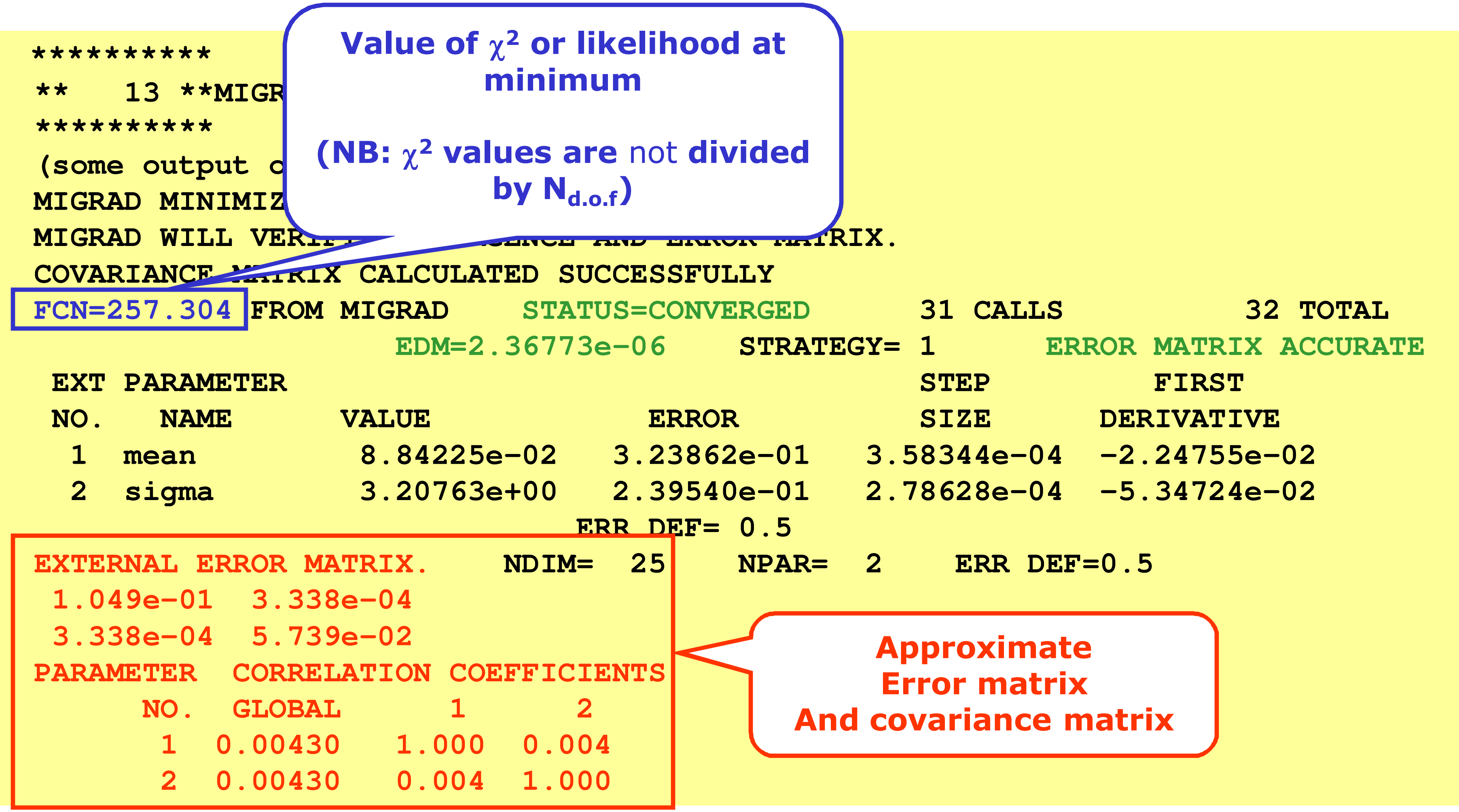
Progress information, watch for errors here

Parameter values and approximate errors reported by MINUIT

Error definition (in this case 0.5 for a likelihood fit)

Minuit function MIGRAD

- Purpose: find minimum



```
**********
**   13 **MIGR
**********
(some output o
MIGRAD MINIMIZ
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                     EDM=2.36773e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                  STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE       DERIVATIVE
  1  mean         8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
  2  sigma        3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                                    ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  3.338e-04
 3.338e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1       2
       1  0.00430   1.000   0.004
       2  0.00430   0.004   1.000
```

Value of χ^2 or likelihood at minimum

(NB: χ^2 values are not **divided by $N_{d.o.f}$)**

Approximate Error matrix And covariance matrix

Minuit function MIGRAD

- Purpose: find minimu

Status:
Should be ‘converged’ but can be ‘failed’

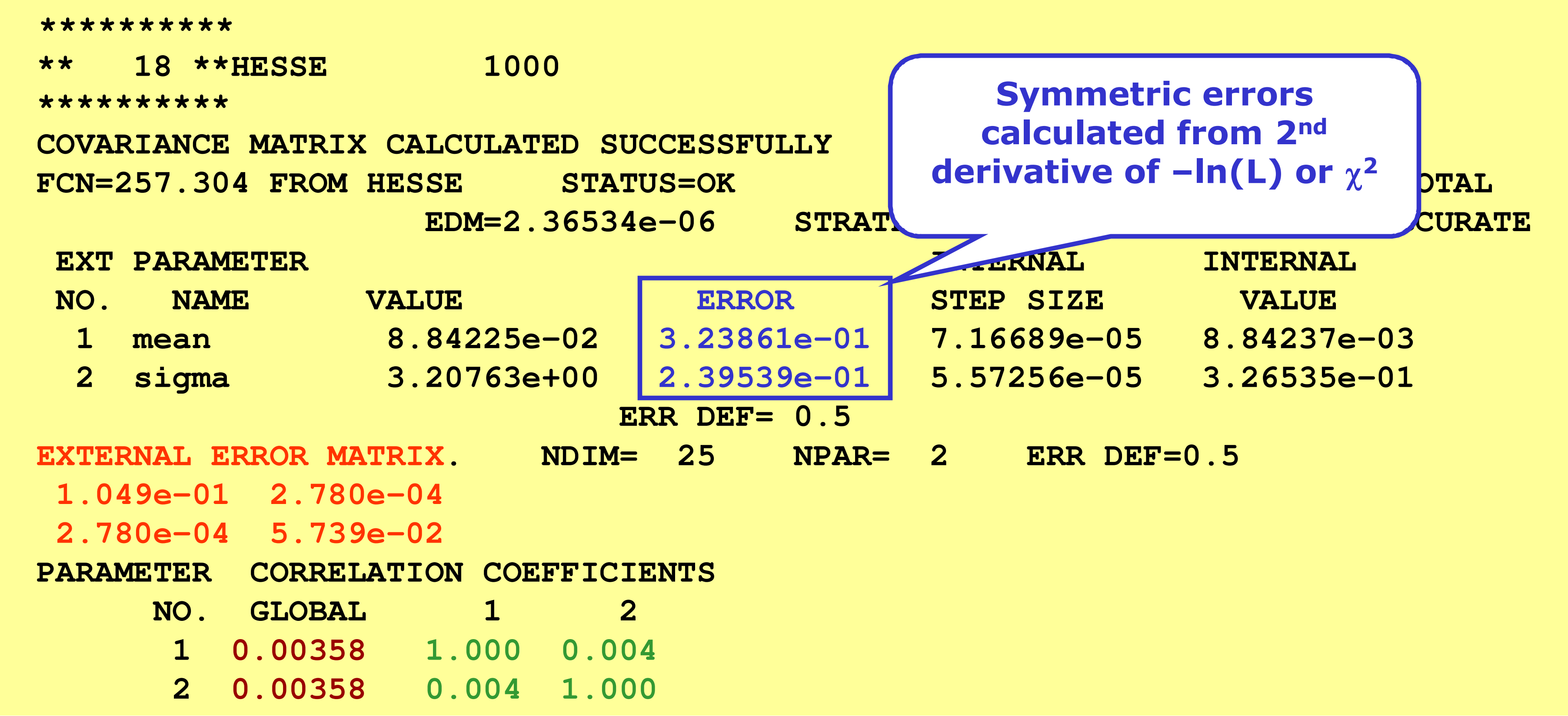
Estimated Distance to Minimum
should be small O(10^{-6})

Error Matrix Quality
should be ‘accurate’, but can be ‘approximate’ in case of trouble

```
**********
**    13 **MIGRAD        1000
**********
(some output omitted)
MIGRAD MINIMIZATION HAS CONVERGED
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                     EDM=2.36773e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE       DERIVATIVE
  1  mean         8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
  2  sigma        3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  3.338e-04
 3.338e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1       2
       1  0.00430   1.000   0.004
       2  0.00430   0.004   1.000
```

Minuit function HESSE

- Purpose: calculate error matrix from $\frac{d^2L}{dp^2}$

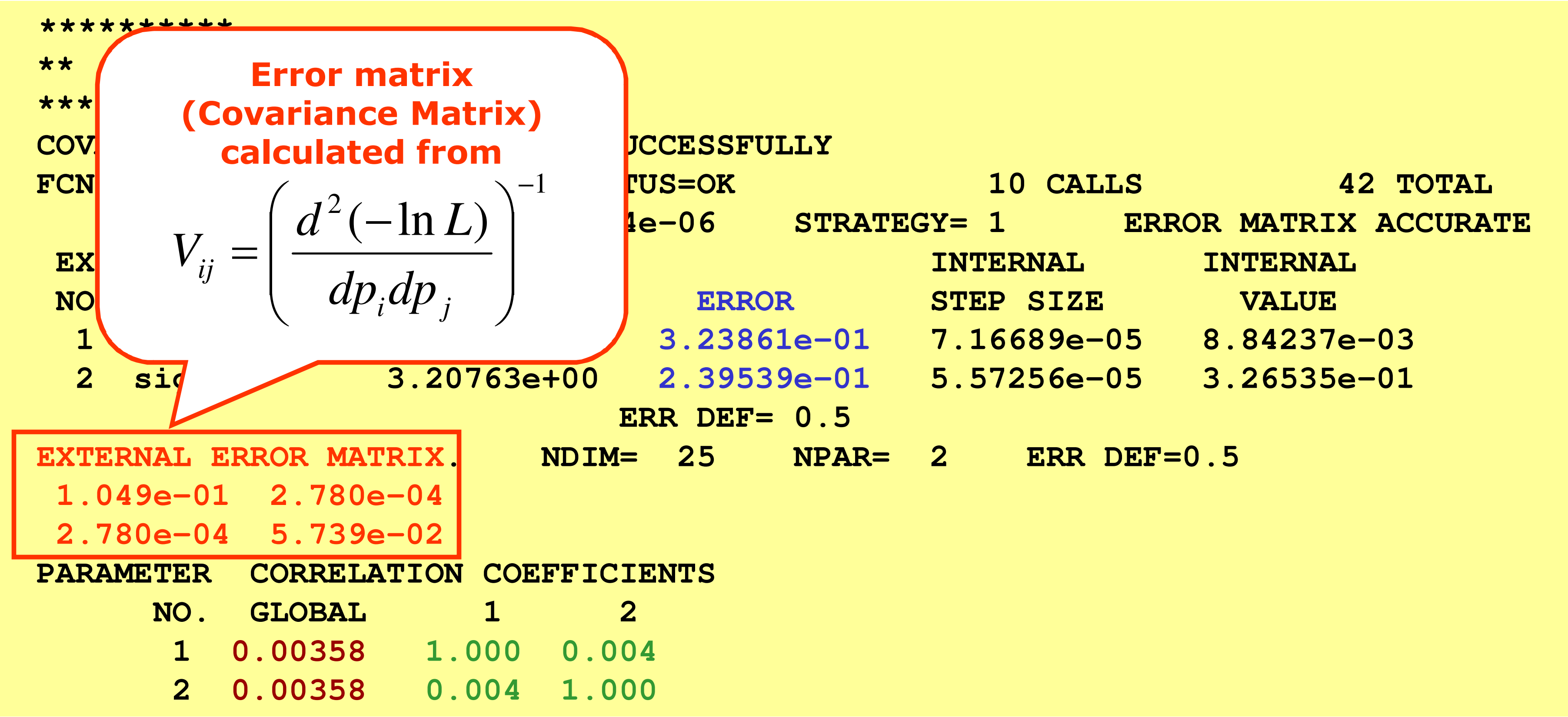


```
**********
**   18 **HESSE         1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK                              OTAL
                    EDM=2.36534e-06    STRAT                              CURATE
 EXT PARAMETER                                     ERNAL         INTERNAL
 NO.   NAME        VALUE          ERROR          STEP SIZE       VALUE
  1   mean          8.84225e-02   3.23861e-01   7.16689e-05   8.84237e-03
  2   sigma         3.20763e+00   2.39539e-01   5.57256e-05   3.26535e-01
                                ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  2.780e-04
 2.780e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2
       1  0.00358   1.000  0.004
       2  0.00358   0.004  1.000
```

Symmetric errors calculated from 2nd derivative of –ln(L) or χ^2

Minuit function HESSE

- Purpose: calculate error matrix from $\frac{d^2L}{dp^2}$



```
 **********
 **
 ***
 COV                                UCCESSFULLY
 FCN                                TUS=OK               10 CALLS            42 TOTAL
                                    4e-06     STRATEGY= 1       ERROR MATRIX ACCURATE
  EX                                                    INTERNAL          INTERNAL
  NO                                    ERROR           STEP SIZE          VALUE
   1                                    3.23861e-01     7.16689e-05     8.84237e-03
   2   si               3.20763e+00     2.39539e-01     5.57256e-05     3.26535e-01
                                   ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.       NDIM=  25     NPAR=  2     ERR DEF=0.5
  1.049e-01   2.780e-04
  2.780e-04   5.739e-02
 PARAMETER   CORRELATION COEFFICIENTS
       NO.   GLOBAL       1       2
        1   0.00358     1.000   0.004
        2   0.00358     0.004   1.000
```

Error matrix (Covariance Matrix) calculated from

$$V_{ij} = \left(\frac{d^2(-\ln L)}{dp_i dp_j} \right)^{-1}$$

Minuit function HESSE

- Purpose: calculate error matrix from $\frac{d^2L}{dp^2}$

```
**********
**   18 **HESSE        1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK             10 CALLS          42 TOTAL
                    EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR       STEP SIZE       VALUE
  1  mean         8.84225e-02                               8.84237e-03
  2  sigma        3.20763e+00                               3.26535e-01

EXTERNAL ERROR MATRIX.    NDIM                           F=0.5
 1.049e-01  2.780e-04
 2.780e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENT
      NO.  GLOBAL      1      2
       1  0.00358   1.000  0.004
       2  0.00358   0.004  1.000
```

Correlation matrix ρ_{ij} calculated from

$$V_{ij} = \sigma_i \sigma_j \rho_{ij}$$

Minuit function HESSE

- Purpose: calculate error matrix from $\frac{d^2L}{dp^2}$

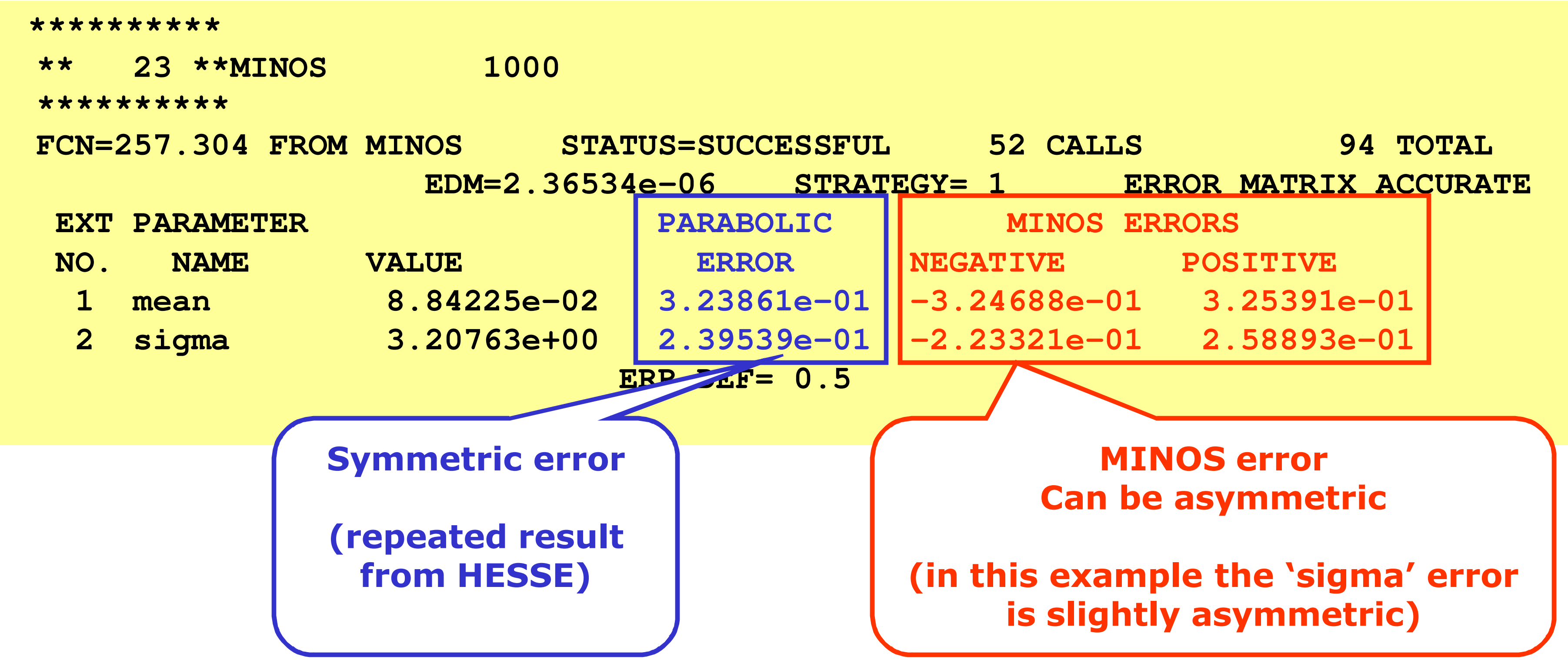
```
**********
**   18 **HESSE        1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK            10 CALLS          42 TOTAL
                    EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                  INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR         STEP SIZE       VALUE
  1  mean                                       7.16689e-05   8.84237e-03
  2  sigma                                      5.57256e-05   3.26535e-01

EXTERNAL ERROR                                  2    ERR DEF=0.5
 1.049e-01  2.
 2.780e-04  5.739e-
PARAMETER  CORRELA    N COEFFICIENTS
      NO.  GLOBAL      1      2
       1  0.00358   1.000  0.004
       2  0.00358   0.004  1.000
```

Global correlation vector: correlation of each parameter with *all other* parameters

Minuit function MINOS

- MINOS errors are calculated by ‘hill climbing algorithm’.
  - In one dimension find points where ΔL=+0.5.
  - In >1 dimension find contour with ΔL=+0.5. Errors are defined by bounding box of contour.
  - In >>1 dimension very time consuming, but more in general more robust.
- Optional – activated by option “E” in ROOT or PAW



```
**********
 **   23 **MINOS        1000
 **********
 FCN=257.304 FROM MINOS     STATUS=SUCCESSFUL     52 CALLS          94 TOTAL
                     EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                  PARABOLIC         MINOS ERRORS
  NO.   NAME      VALUE            ERROR      NEGATIVE      POSITIVE
   1  mean        8.84225e-02   3.23861e-01  -3.24688e-01   3.25391e-01
   2  sigma       3.20763e+00   2.39539e-01  -2.23321e-01   2.58893e-01
                               ERR DEF= 0.5
```

Symmetric error

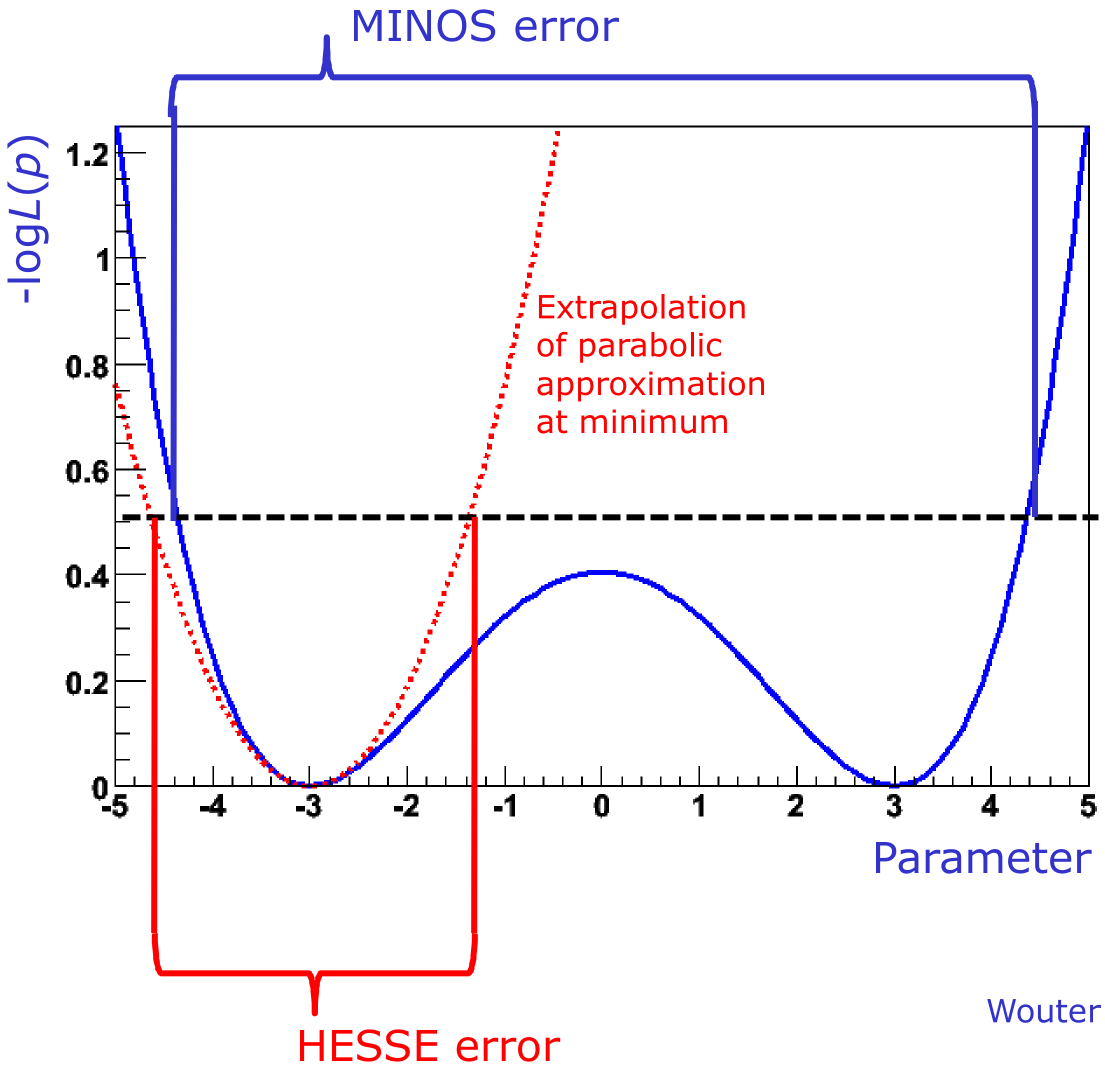
(repeated result from HESSE)

MINOS error
Can be asymmetric

(in this example the ‘sigma’ error is slightly asymmetric)

Illustration of difference between HESSE and MINOS errors

- ‘Pathological’ example likelihood with multiple minima and non-parabolic behavior



MINOS error

Extrapolation of parabolic approximation at minimum

HESSE error

Practical estimation – Fit converge problems

- Sometimes fits don't converge because, e.g.
  - MIGRAD unable to find minimum
  - HESSE finds negative second derivatives (which would imply negative errors)

- Reason is usually numerical precision and stability problems, but
  - The underlying cause of fit stability problems is usually by **highly correlated parameters** in fit

- HESSE correlation matrix in primary investigative tool

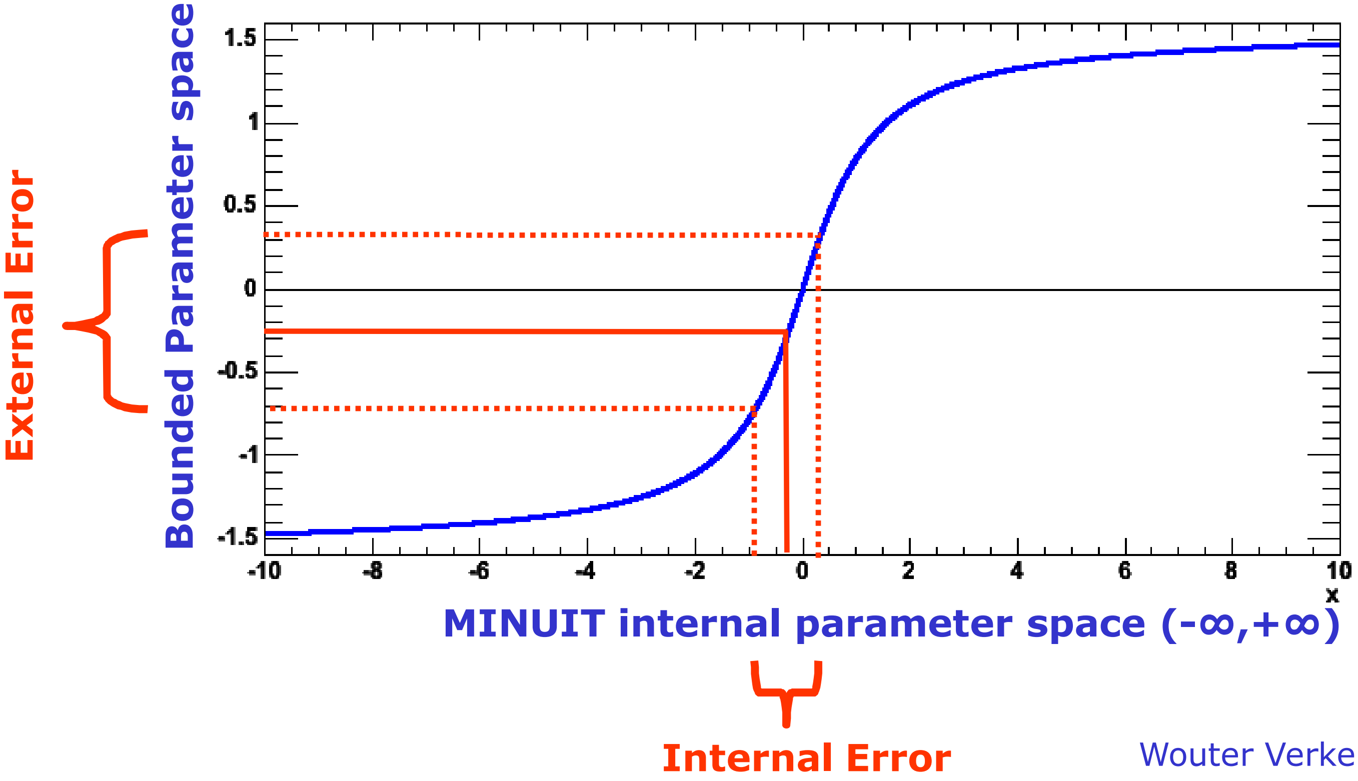
```
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2
       1  0.99835   1.000  0.998
       2  0.99835   0.998  1.000
```

Signs of trouble…

  - In limit of 100% correlation, the usual point solution becomes a line solution (or surface solution) in parameter space. Minimization problem is no longer well defined

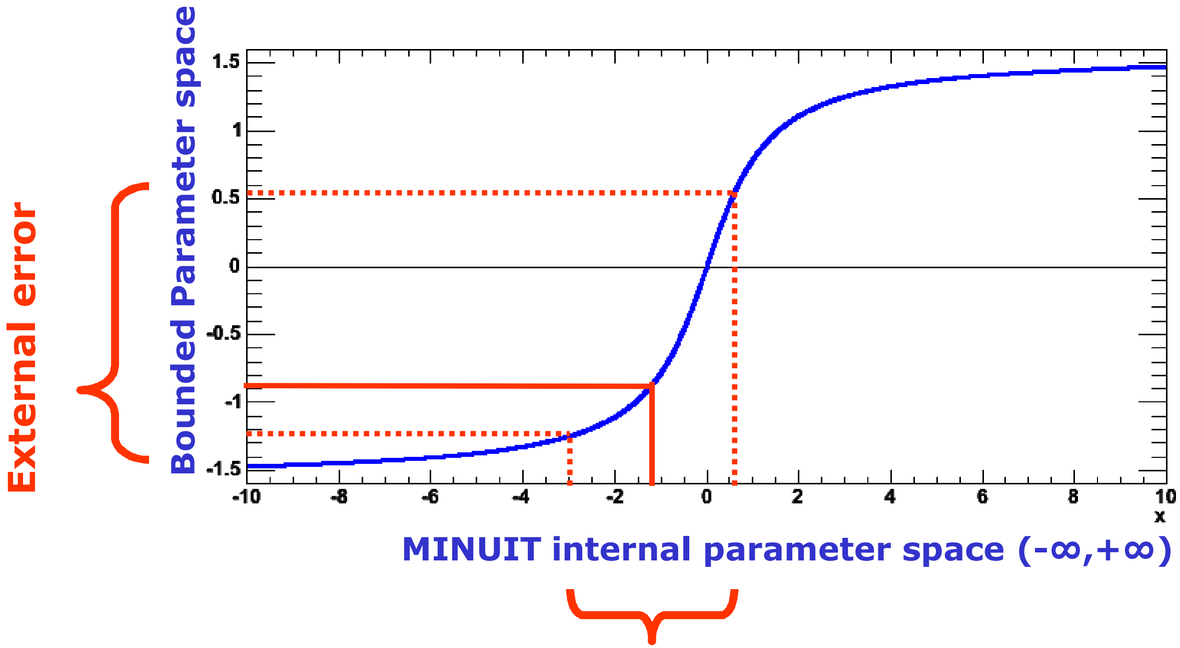
Practical estimation – Bounding fit parameters

- Sometimes is it desirable to bound the allowed range of parameters in a fit
  - Example: a fraction parameter is only defined in the range [0,1]
  - MINUIT option ‘B’ maps finite range parameter to an internal infinite range using an arcsin(x) transformation:



Practical estimation – Bounding fit parameters

- If fitted parameter values is close to boundary, errors will become asymmetric (and possible incorrect)



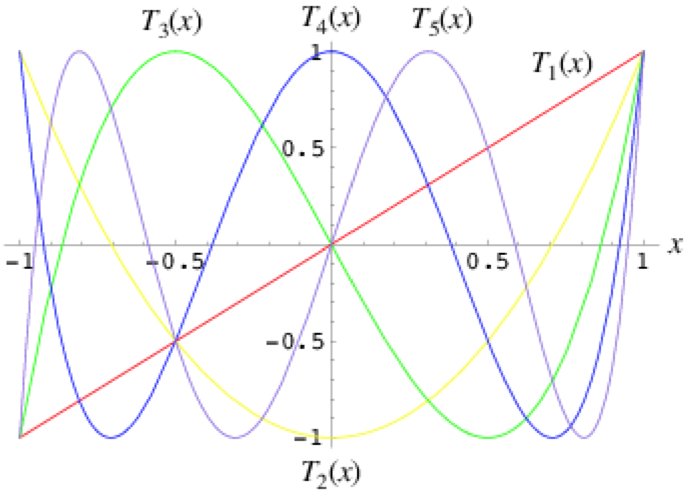
Internal error

- So be careful with bounds!
  - If boundaries are imposed to avoid region of instability, look into other parameterizations that naturally avoid that region
  - If boundaries are imposed to avoid 'unphysical', but statistically valid results, consider not imposing the limit and dealing with the 'unphysical' interpretation in a later stage

Mitigating fit stability problems -- Polynomials

- **Warning:** Regular parameterization of polynomials $a_0+a_1x+a_2x^2+a_3x^3$ nearly always results in strong correlations between the coefficients a_i.
  - *Fit stability problems, inability to find right solution common at higher orders*
- **Solution:** Use existing parameterizations of polynomials that have (mostly) uncorrelated variables
  - ***Example: Chebychev polynomials***

$$\left.\begin{aligned}
T_0(x) &= 1\\
T_1(x) &= x\\
T_2(x) &= 2x^2-1\\
T_3(x) &= 4x^3-3x\\
T_4(x) &= 8x^4-8x^2+1\\
T_5(x) &= 16x^5-20x^3+5x\\
T_6(x) &= 32x^6-48x^4+18x^2-1.
\end{aligned}\right\} \longrightarrow$$



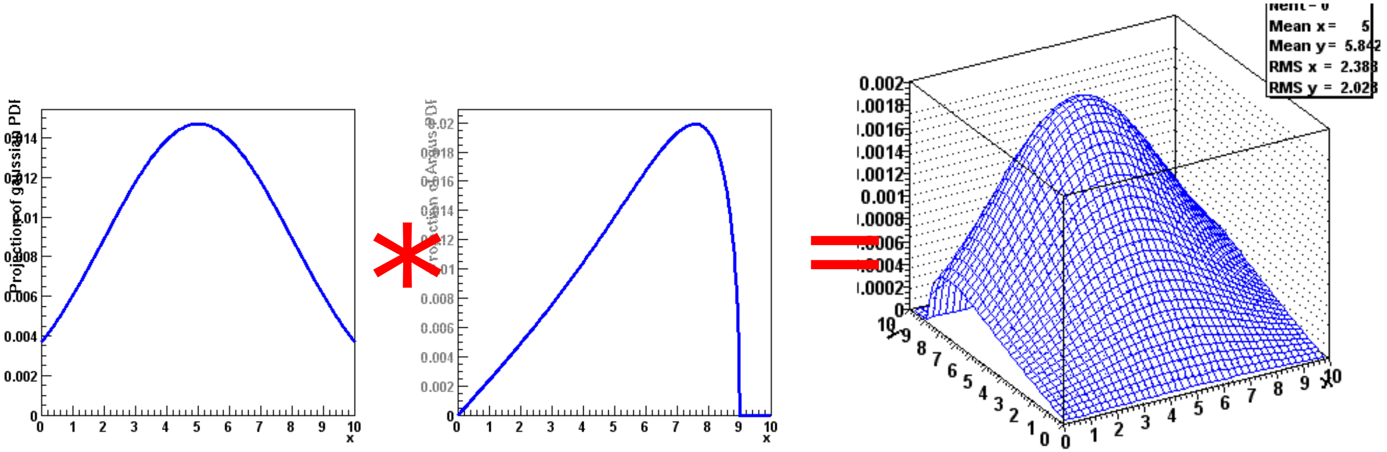
Extending models to more than one dimension

- If you have data with many observables, there are two common approaches
  - Compactify information with test statistic (see previous section)
  - Describe full N-dimensional distribution with a p.d.f.

- Choice of approach largely correlated with understanding of correlation between observables and amount of information contained in correlations
  - No correlation between observables → 'Big fit' and 'Compactification' work equally well.
  - Important correlations that are poorly understood → Compactification preferred. Approach:
    1. Compactify all-but-one observable (ideally uncorrelated with the compactified observables)
    2. Cut on compactification test statistic to reduce backgrounds
    3. Fit remaining observable → Estimate from data remaining amount of background (smallest systematic uncertainty due to poor understanding of test statistic and its inputs)
  - Important correlations that are well understood → Big fit preferred

Extending models to more than one dimension

- Bottom line: N-dim models used when either *no correlations* or *well understood correlations*
- Constructing multi-dimensional models without correlations is easy
  - Just multiply N 1-dimensional p.d.f.s.

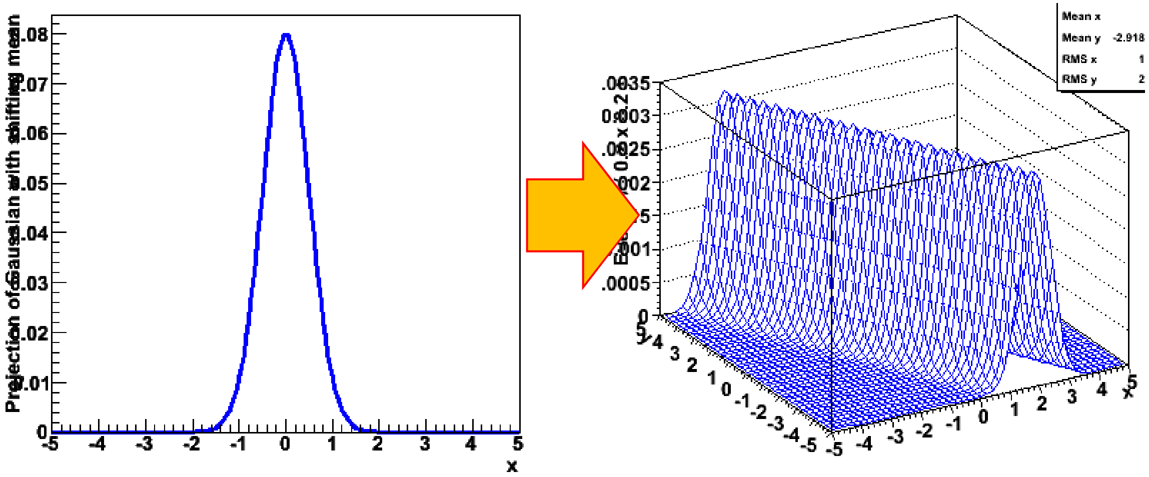


  - No complex issues with p.d.f. normalization: if 1-dim p.d.f.s are normalized then product is also by construction

Writing multi-dimensional models with correlations

- Formulating N-dim models *with* correlations may seem daunting, but it really isn’t so difficult.
  - Simplest approach: start with one-dimensional model, replace one parameter p with a function p’(y) of another observable
  - Yields correction distribution of x for every given value of y

$F(x,m,s)$ $F'(x,[a_0+a_1y],s)$



  - NB: Distribution of y probably *not* correct…

Writing multi-dimensional models with correlations

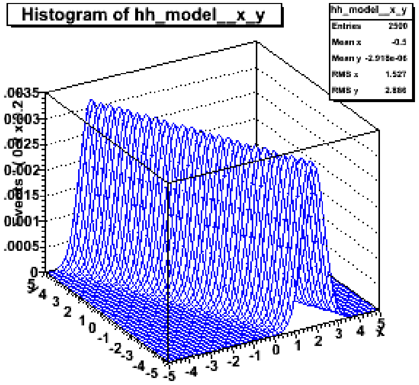
- Solution: see F’(x,y,p) as a *conditional p.d.f.* F’(x|y)
  - Difference is in normalization

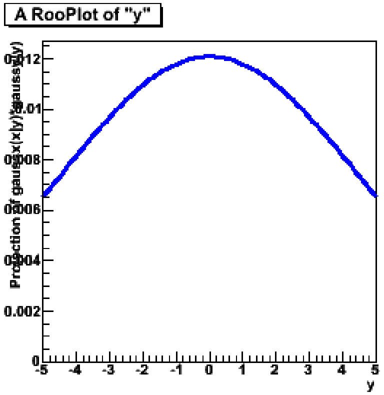
$$\int F(x,y)dxdy \equiv 1 \qquad \int F(x\,|\,y)dx \equiv 1$$ for *each* value of y

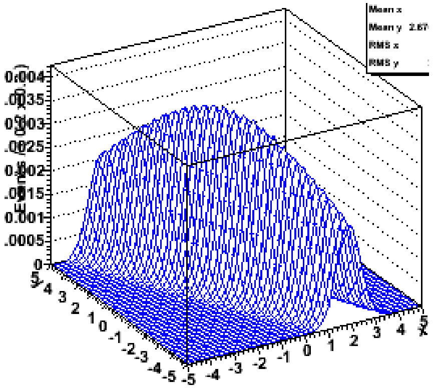
  - Then multiply with a separate p.d.f describing distribution in y

$$M(x,y) = F'(x\,|\,y)\cdot G(y)$$

F’(x|y) ***** **G(y)** **=** **M(x,y)**





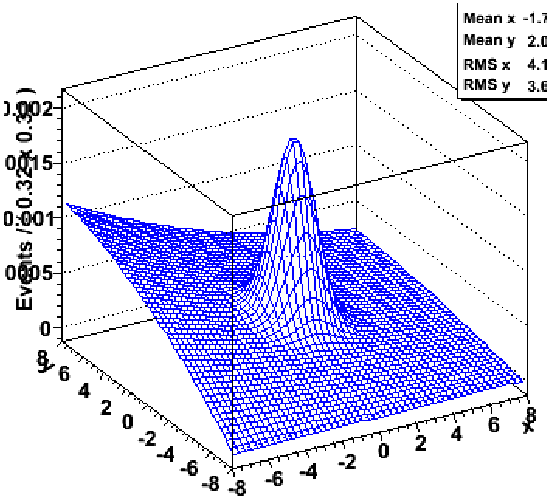


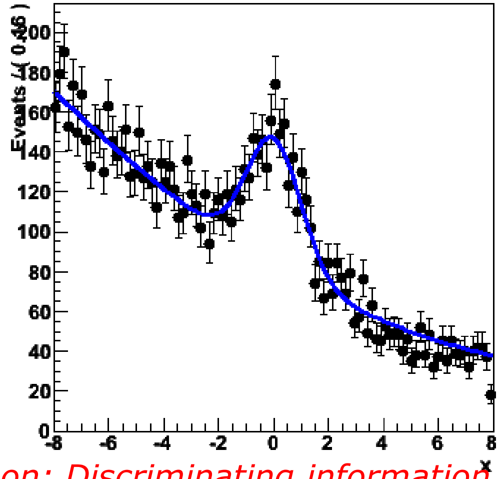
- Almost ***all*** modeling issues with correlations can be treated this way
  - Iteration 1) Exponential Missing ET distr. of ‘background’ is independent of Transverse mass
  - Iteration 2) Slope depends linearly on MT → write conditional pdf F(ET|MT)
  - Iteration 3) Multiply F(ET|MT) with empirical shape for MT

Visualization of multi-dimensional models

- Visualization of multi-dimensional models presents some additional challenges w.r.t. 1-D
- Can show 2D,3D distribution
  - Graphically appealing, but not so useful as you cannot overlay model on data and judge goodness-of-fit
  - Prefer to project on one dimension (there will be multiple choices)
  - But plain projection discards a lot of information contained in both model and data

Significance of signal less apparent

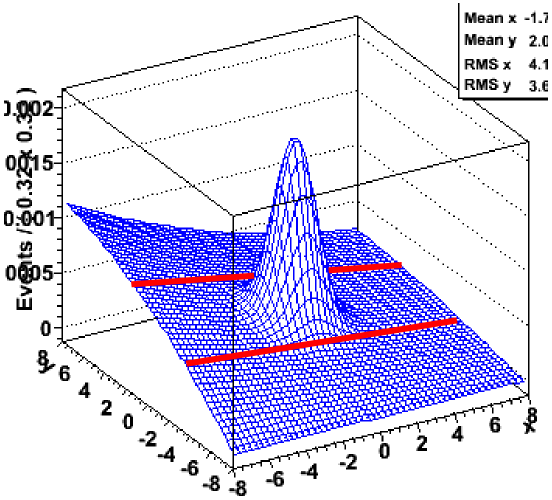




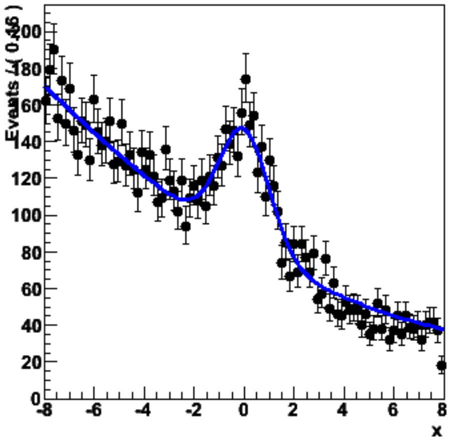
*Reason: Discriminating information in **y** observable in both data and model is ignored*

Visualizing signal projections of N-dim models

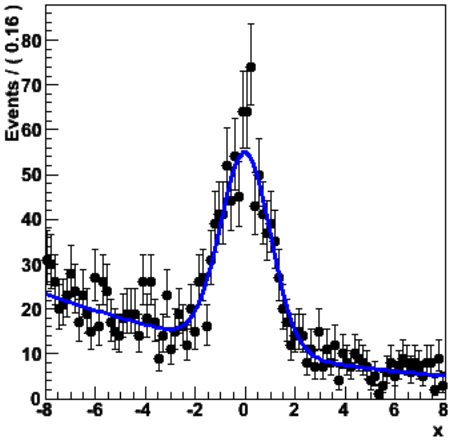
- Simplest solution, only show model and data in "signal range" of observable y
  - Significance shown in "range projection" much more in line with that of 2D distribution



y∈[-10,10]



y∈[-2,2]

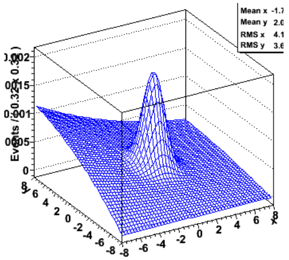


- Easy to define a "signal range" simple model above. How about 6-dimensional model with non-trivial shape?
  - Need *generic algorithm* → Likelihood ratio plot

Likelihood ratio plots

- Idea: use information on S/(S+B) ratio in projected observables to define a cut

- Example: generalize previous toy model to 3 dimensions



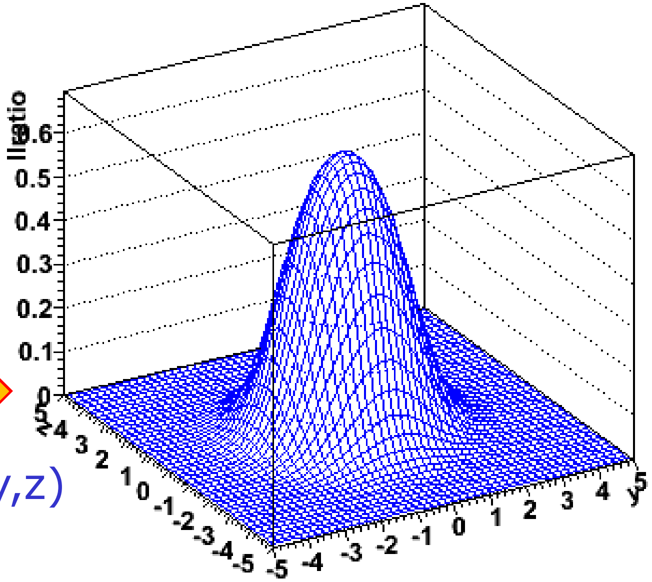
- Express information on S/(S+B) ratio of model in terms of integrals over model components

$$LR(x,y,z)=\frac{S(x,y,z)}{\left[S(x,y,z)+B(x,y,z)\right]}$$

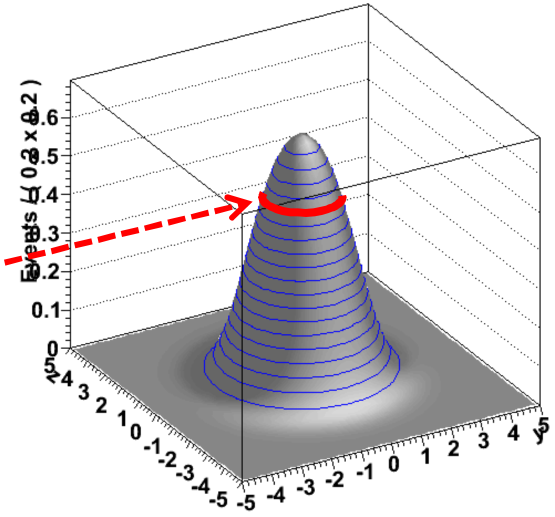
Integrate over x

$$LR(y,z)=\frac{\int S(x,y,z)dx}{\int\left[S(x,y,z)+B(x,y,z)\right]dx}$$

Plot LR vs (y,z)



Likelihood ratio plots

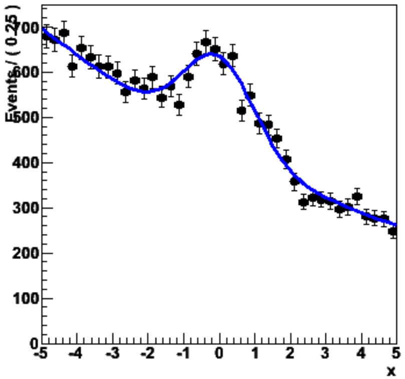


- Decide on s/(s+b) purity contour of LR(y,z)
  - Example s/(s+b) > 50%
- Plot both data and model with corresponding cut.
  - For data: calculate LR(y,z) for each event, plot only event with LR>0.5
  - For model: using Monte Carlo integration technique:

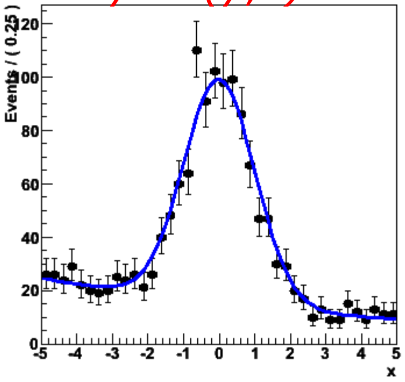
$$\int_{LR(y,z)>0.5} M(x,y,z)dydz \approx \frac{1}{N}\sum_{D(y,z)} M(x,y_i,z_i)$$

Dataset with values of (y,z) sampled from p.d.f and filtered for events that meet LR(y,z)>0.5

All events



Only LR(y,z)>0.5

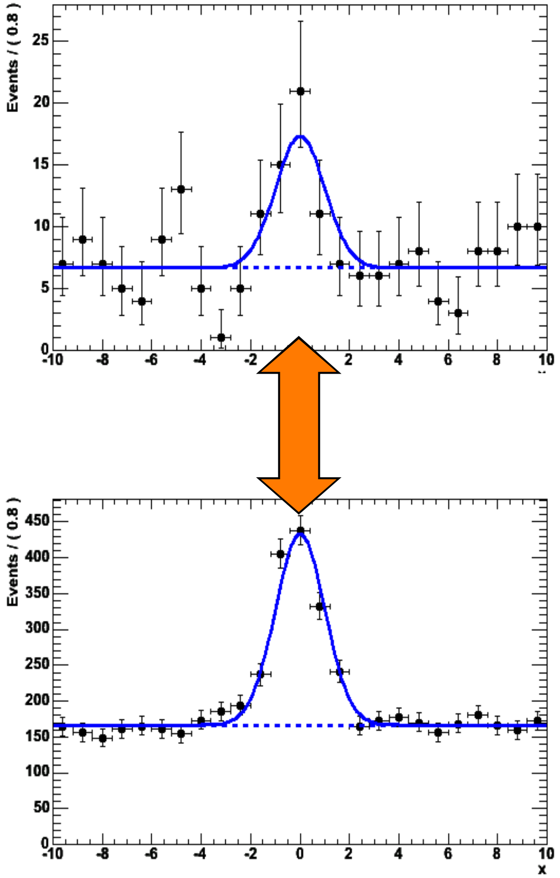


Multidimensional fits – Goodness-of-fit determination

- Goodness-of-fit determination of >1 D models is difficult
  - Standard χ^2 test does not work very will in N-dim because of natural occurrence of large number of empty bins
  - Simple equivalent of (unbinned) Kolmogorov test in >1-D does not exist
- This area is still very much a work in progress
  - Several new ideas proposed but sometimes difficult to calculate, or not universally suitable
  - Some examples
    - Cramer-von Mises (close to Kolmogorov in concept)
    - Anderson-Darling
    - ‘Energy’ tests
  - **No magic bullet here, “best” generally an ill-posed question**
  - Some references to recent progress:
    - PHYSTAT2001/2003/2005

Practical fitting – Error propagation between samples

- Common situation: you want to fit a small signal in a large sample
  - Problem: small statistics does not constrain shape of your signal very well
  - Result: errors are large



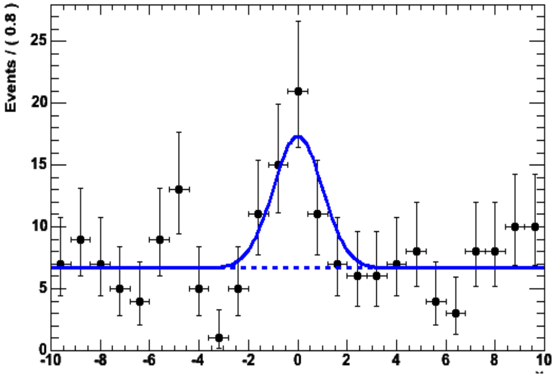
- Idea: Constrain shape of your signal from a fit to a control sample
  - Larger/cleaner data or MC sample with similar properties

- Needed: a way to propagate the information from the control sample fit (parameter values *and* errors) to your signal fit

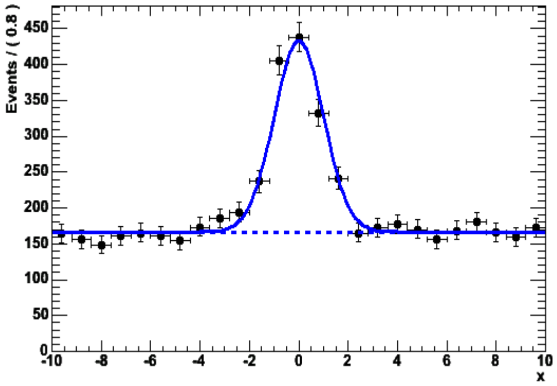
Practical fitting – Simultaneous fit technique

- given data $D_{sig}(x)$ and model $F_{sig}(x;a,\boldsymbol{b})$ and
  data $D_{ctl}(x)$ and model $F_{ctl}(x;\boldsymbol{b},c)$
  - Construct $-\log[L_{sig}(a,\boldsymbol{b})]$ and $-\log[L_{ctl}(\boldsymbol{b},c)]$ and

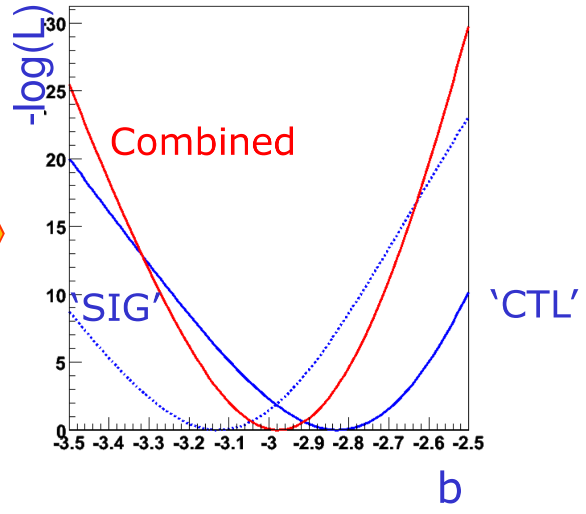
$D_{sig}(x)$, $F_{sig}(x;a,\boldsymbol{b})$



$D_{ctl}(x)$, $F_{ctl}(x;\boldsymbol{b},c)$



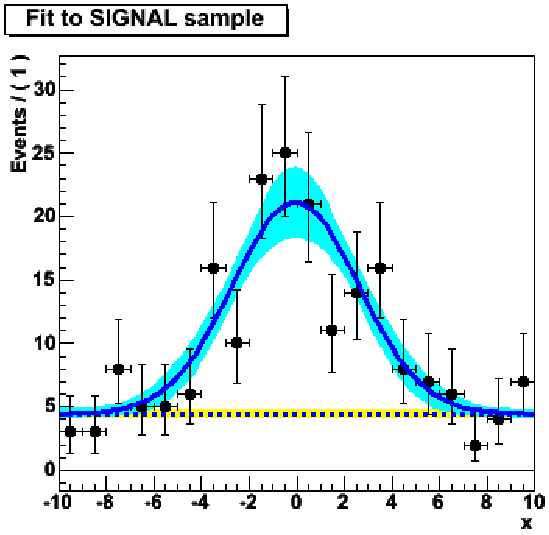
Likelihood view



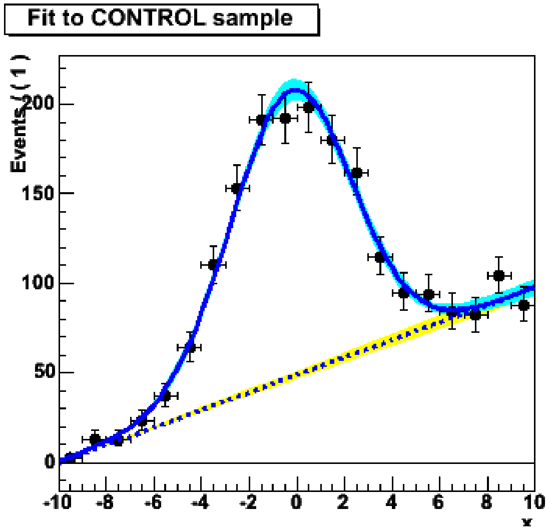
- Minimize $-\log L(a,\boldsymbol{b},c) = -\log L(a,\boldsymbol{b}) + -\log L(\boldsymbol{b},c)$
  - Errors, correlations on common param. ***b*** automatically propagated

Practical fitting – Simultaneous fit technique

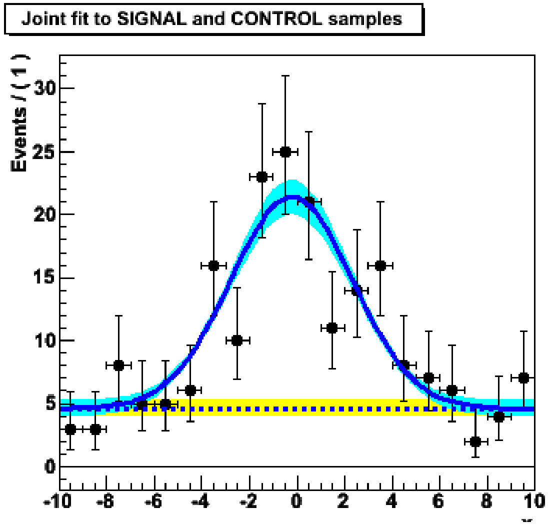
- Simultaneous fit with visualization of error



Fit to SIGNAL sample



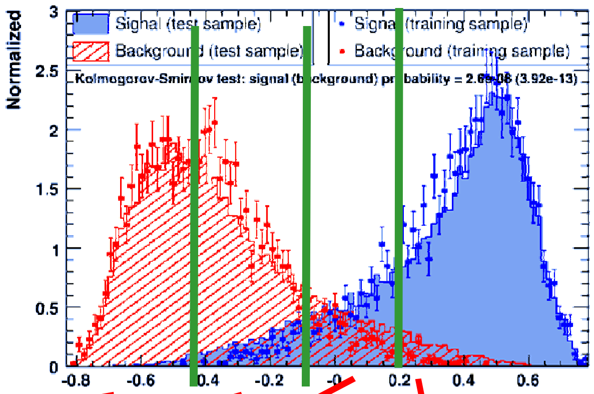
Fit to CONTROL sample

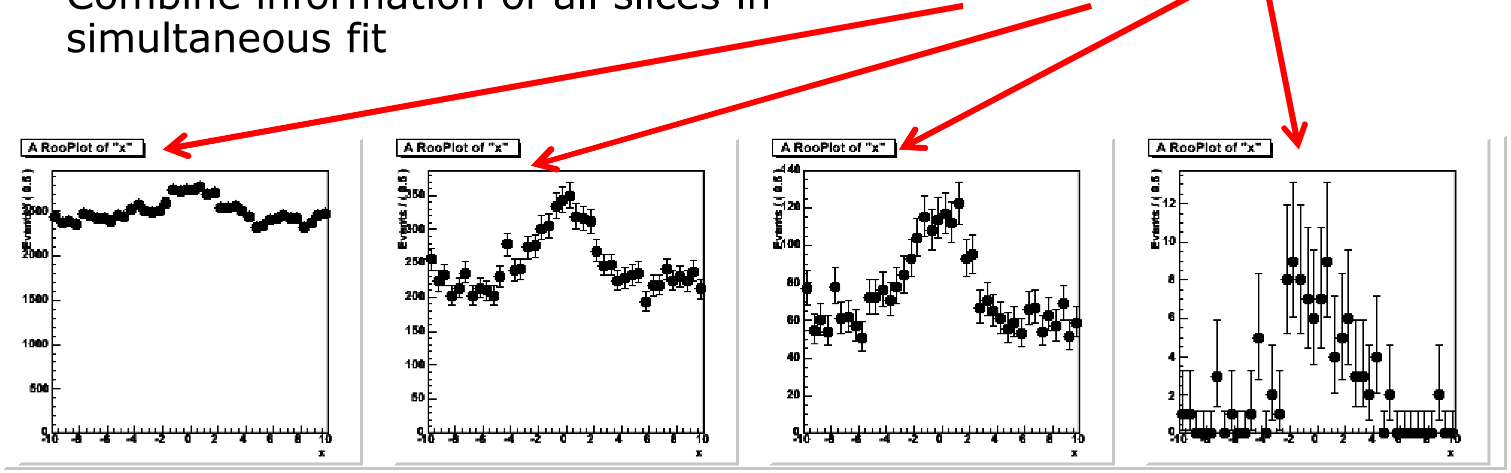


Joint fit to SIGNAL and CONTROL samples

Another application of simultaneous fits

- You can also use simultaneous fits to samples of the same type (“signal samples”) with different purity
- Go back to example of NN with one observable left out
  - Fit xN after cut on N(x)
  - But instead of just fitting data with N(x)>α, slice data in bins of N(x) and fit *each bin.*
  - Now you exploit all data instead of just most pure data. Still no uncontrolled systematic uncertainty as purity is measured from data in each slide
  - Combine information of all slices in simultaneous fit





Practical Estimation – Verifying the validity of your fit

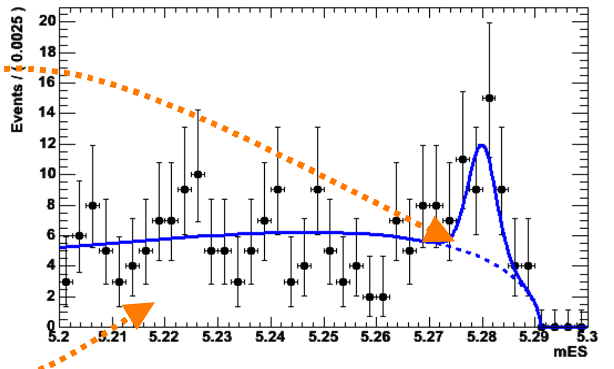
- How to validate your fit? – You want to demonstrate that
  1) Your fit procedure gives on average the correct answer **'no bias'**
  2) The uncertainty quoted by your fit is an accurate measure for the statistical spread in your measurement **'correct error'**

- **Validation is important for low statistics fits**
  - **Correct behavior not obvious a priori due to intrinsic ML bias proportional to 1/N**

- Basic validation strategy – A simulation study
  1) Obtain a large sample of simulated events
  2) Divide your simulated events in O(100-1000) samples with the same size as the problem under study
  3) Repeat fit procedure for each data-sized simulated sample
  4) Compare average value of fitted parameter values with generated value → **Demonstrates (absence of) bias**
  5) Compare spread in fitted parameters values with quoted parameter error → **Demonstrates (in)correctness of error**

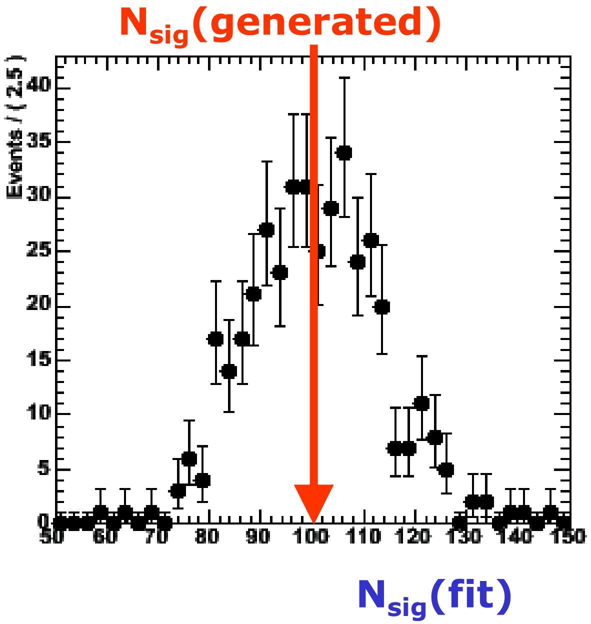
Fit Validation Study – Practical example

- Example fit model in 1-D (B mass)
  - Signal component is Gaussian centered at B mass
  - Background component is Argus function (models phase space near kinematic limit)



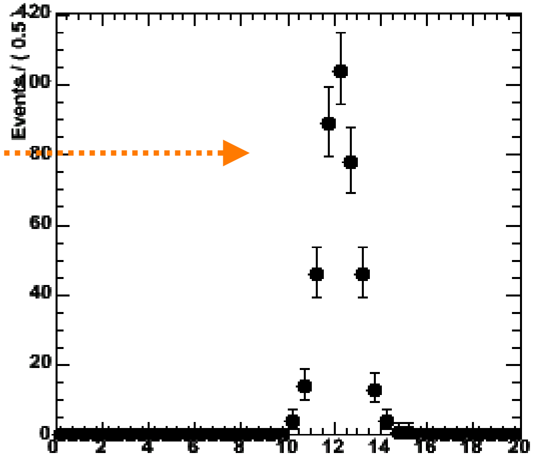
$$F(m; N_{\mathrm{sig}}, N_{\mathrm{bkg}}, \vec{p}_S, \vec{p}_B) = N_{\mathrm{sig}} \cdot G(m; p_S) + N_{\mathrm{bkg}} \cdot A(m; p_B)$$

- Fit parameter under study: ***N*_sig**
  - Results of simulation study: 1000 experiments with **N_{SIG}(gen)=100, N_{BKG}(gen)=200**
  - Distribution of N_{sig}(fit) ....▶
  - This particular fit looks unbiased…



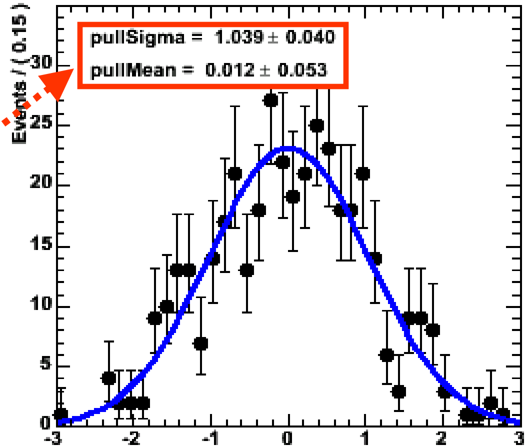
Fit Validation Study – The pull distribution

- What about the validity of the error?
  - Distribution of error from simulated experiments is difficult to interpret…
  - We don't have equivalent of N_{sig}(generated) for the error



$\sigma(N_{sig})$

- Solution: look at the ***pull distribution***
  - Definition: $$\mathrm{pull}(N_{\mathrm{sig}}) = \frac{N_{sig}^{fit} - N_{sig}^{true}}{\sigma_N^{fit}}$$
  - Properties of pull:
    - **Mean is 0 if there is no bias**
    - **Width is 1 if error is correct**
  - In this example: no bias, correct error within statistical precision of study



$\mathrm{pull}(N_{sig})$

Fit Validation Study – Low statistics example

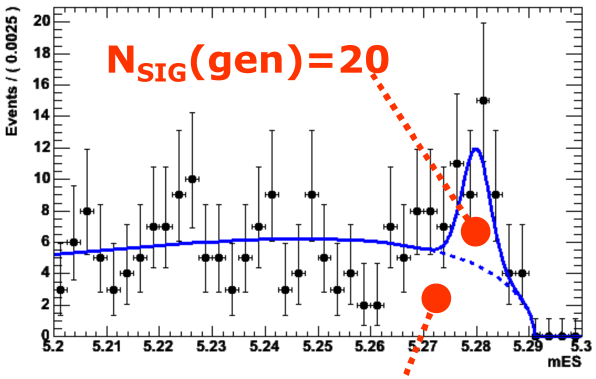
- Special care should be taken when fitting small data samples
  - Also if fitting for small signal component in large sample

- Possible causes of trouble
  - χ^2 estimators may become approximate as Gaussian approximation of Poisson statistics becomes inaccurate
  - ML estimators may no longer be efficient
    → error estimate from 2nd derivative may become inaccurate
  - Bias term proportional to 1/N of ML and χ^2 estimators may no longer be small compared to 1/sqrt(N)

- In general, absence of bias, correctness of error can not be assumed. How to proceed?
  - Use unbinned ML fits only – most robust at low statistics
  - **Explicitly verify the validity of your fit**

Demonstration of fit bias at low N – pull distributions

- Low statistics example:
  - Scenario as before but now with 200 bkg events and **only 20 signal events** (instead of 100)



N_{SIG}(gen)=20

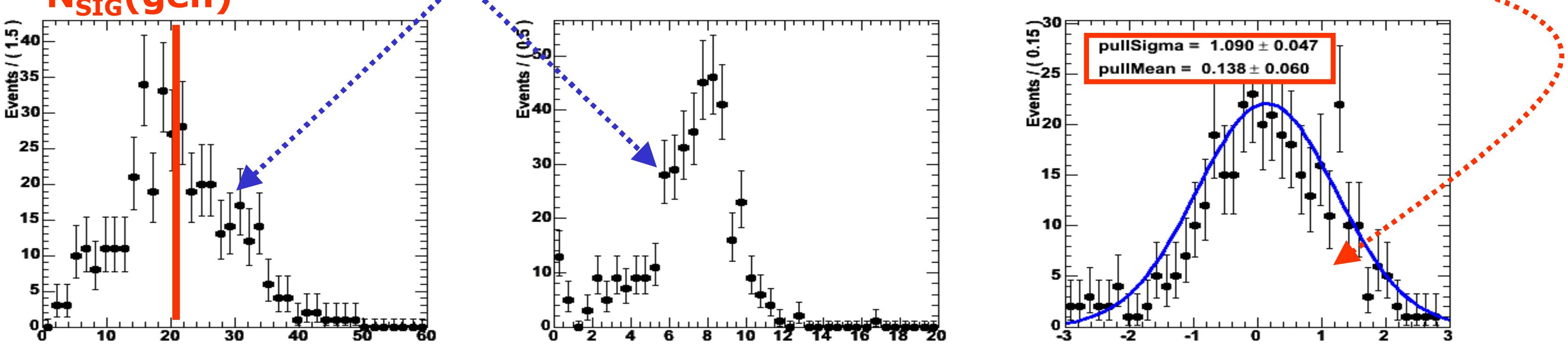
N_{BKG}(gen)=200

- Results of simulation study

Distributions become asymmetric at low statistics

→ Fit is positively biased!

N_{SIG}(gen)



N_{SIG}(fit)

$\sigma(N_{SIG})$

pull(N_{SIG})

- *Absence of bias, correct error at low statistics not obvious!*
  - *Small yields are typically overestimated*

Fit Validation Study – How to obtain 10.000.000 simulated events?

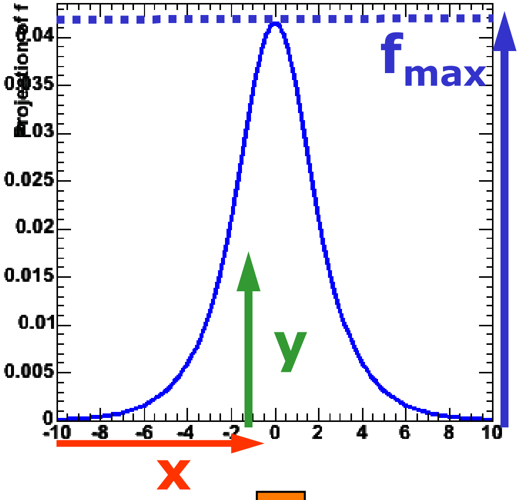
- Practical issue: usually you need very large amounts of simulated events for a fit validation study
  - Of order 1000x number of events in your fit, easily >1.000.000 events
  - Using data generated through a full GEANT-based detector simulation can be prohibitively expensive

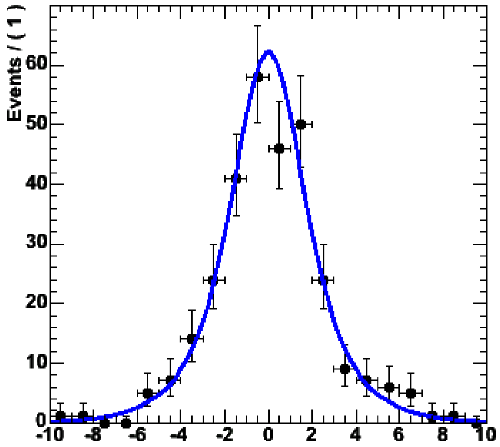
- Solution: Use events sampled directly from your fit function
  - Technique named '*Toy Monte Carlo*' sampling
  - Advantage: Easy to do and very fast
  - Good to determine fit bias due to low statistics, choice of parameterization, boundary issues etc
  - Cannot be used to test assumption that went into model (e.g. absence of certain correlations). Still need full GEANT-based simulation for that.

Toy MC generation – Accept/reject sampling

- *How to sample events directly from your fit function?*
- Simplest: accept/reject sampling

1) Determine maximum of function f_{max}
2) Throw random number x
3) Throw another random number y
4) **If $y<f(x)/f_{max}$ keep x, otherwise return to step 2)**





– PRO: Easy, always works

– CON: It can be inefficient if function is strongly peaked. Finding maximum empirically through random sampling can be lengthy in >2 dimensions

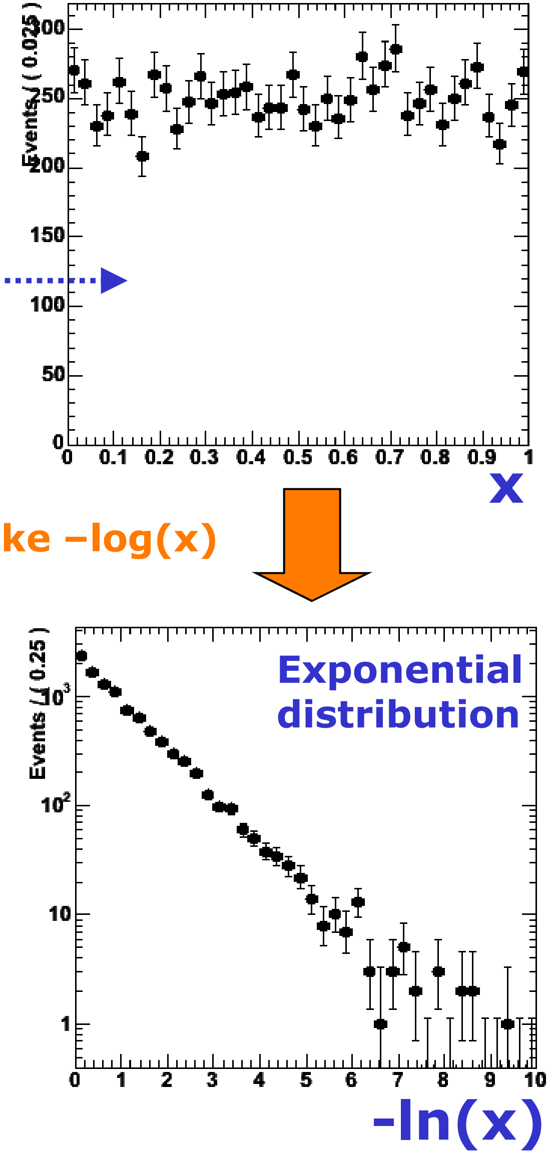
Toy MC generation – Inversion method

- Fastest: function inversion

1) Given f(x) find inverted function F(x) so that f(F(x)) = x

2) Throw uniform random number x

3) Return F(x)

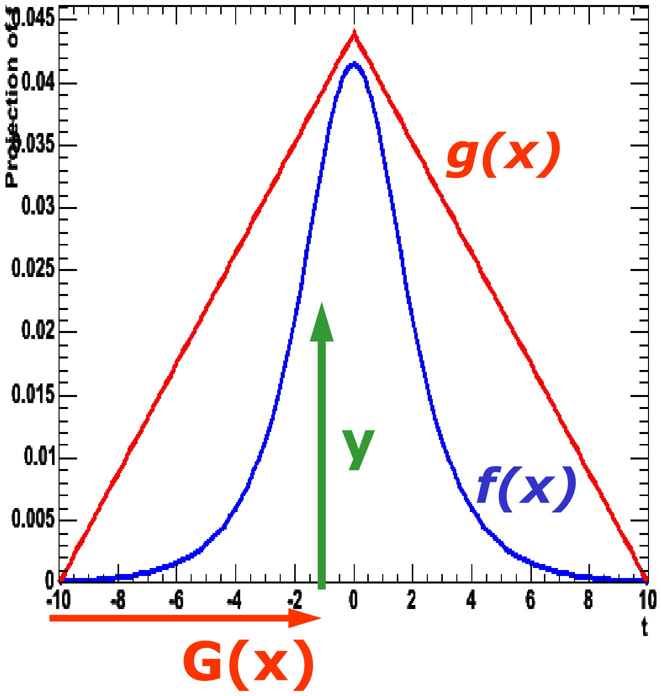


- PRO: Maximally efficient
- CON: Only works for invertible functions

Toy MC Generation in a nutshell

- Hybrid: Importance sampling

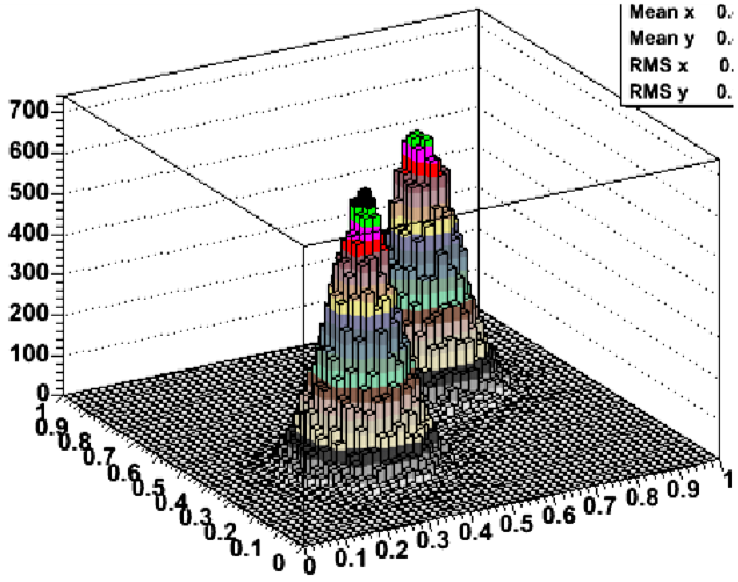
1) Find 'envelope function' g(x) that is invertible into G(x) and that fulfills g(x)>=f(x) for all x
2) Generate random number x from G using inversion method
3) Throw random number 'y'
4) If y<f(x)/g(x) keep x, otherwise return to step 2

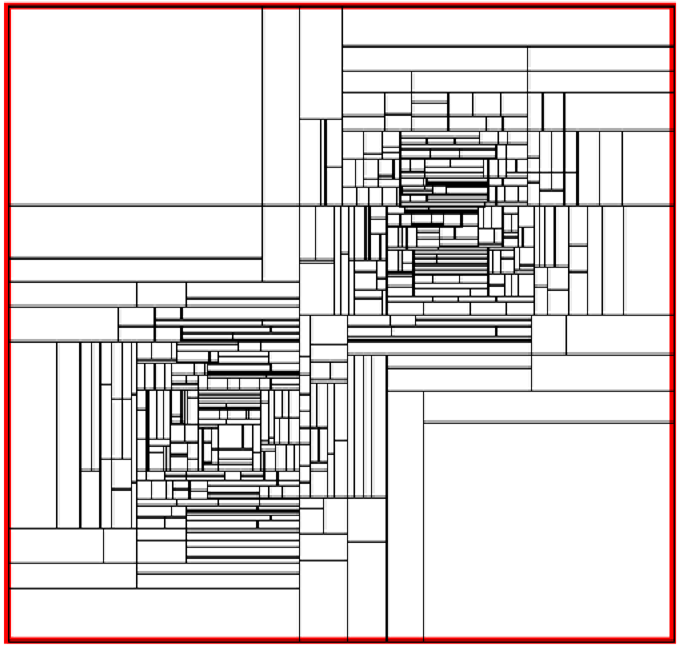


- PRO: Faster than plain accept/reject sampling
  Function does not need to be invertible
- CON: Must be able to find invertible envelope function

Toy MC Generation in a nutshell

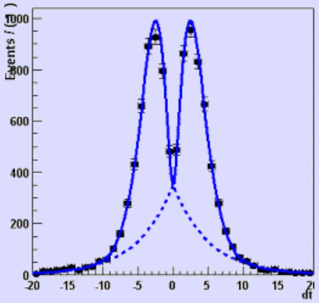
- General algorithms exists that can construct empirical envelope function
  - Divide observable space recursively into smaller boxes and take uniform distribution in each box
  - Example shown below from FOAM algorithm

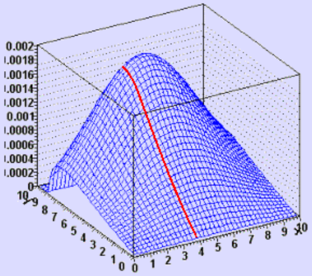


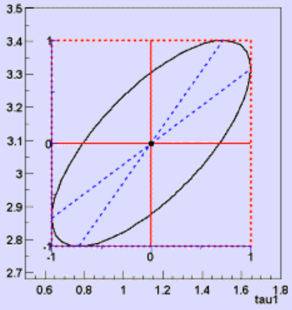


(Software Advertisement #2)

RooFit





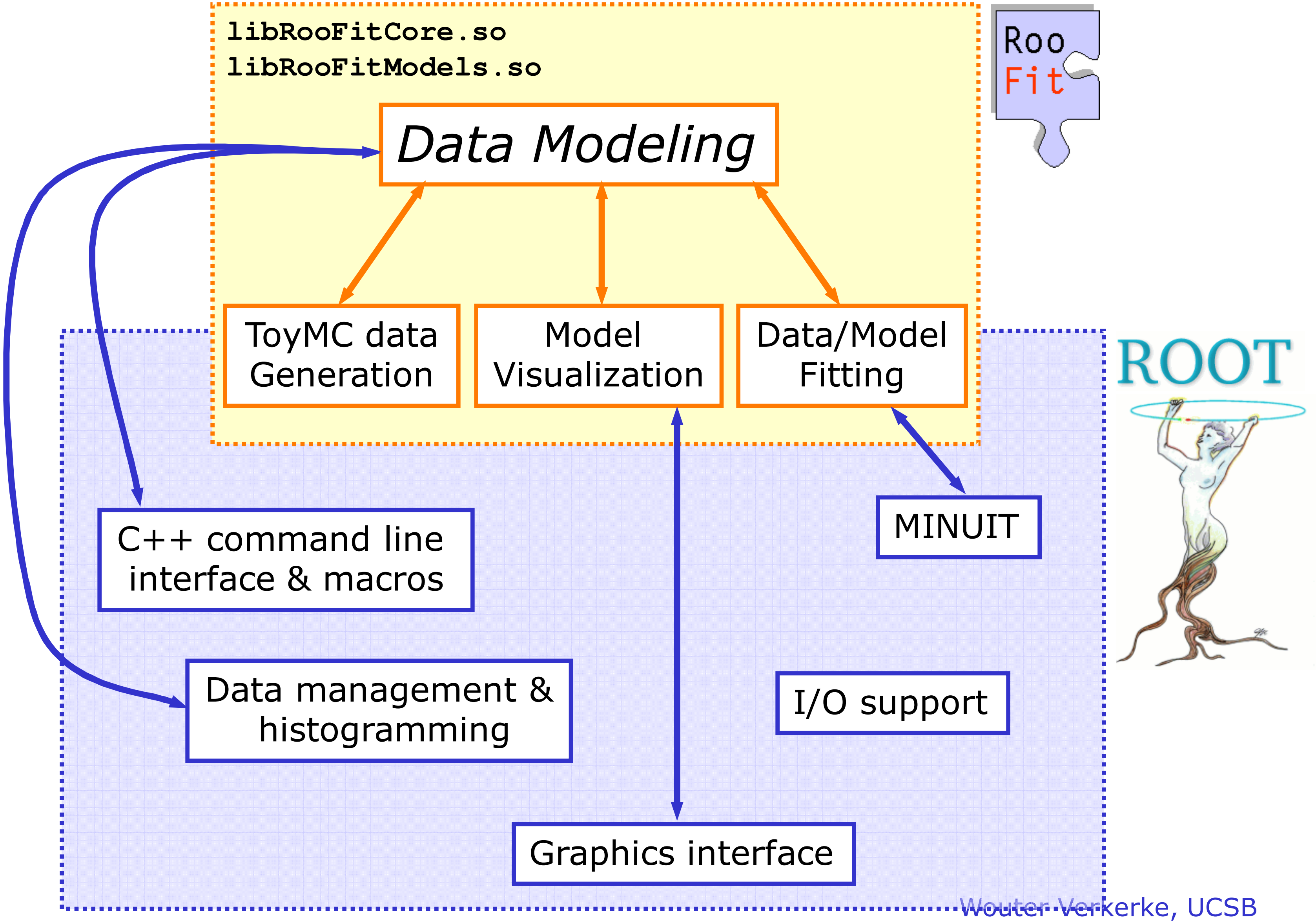


RooFit

A general purpose tool kit for data modeling

Wouter Verkerke (UC Santa Barbara)
David Kirkby (UC Irvine)

Implementation – Add-on package to ROOT



Data modeling – OO representation

- Mathematical objects are represented as C++ objects

Mathematical concept		RooFit class
variable	x, p	`RooRealVar`
function	$f(\vec{x})$	`RooAbsReal`
PDF	$F(\vec{x};\vec{p},\vec{q})$	`RooAbsPdf`
space point	$\vec{x}$	`RooArgSet`
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	`RooRealIntegral`
list of space points	$\vec{x}_k$	`RooAbsData`

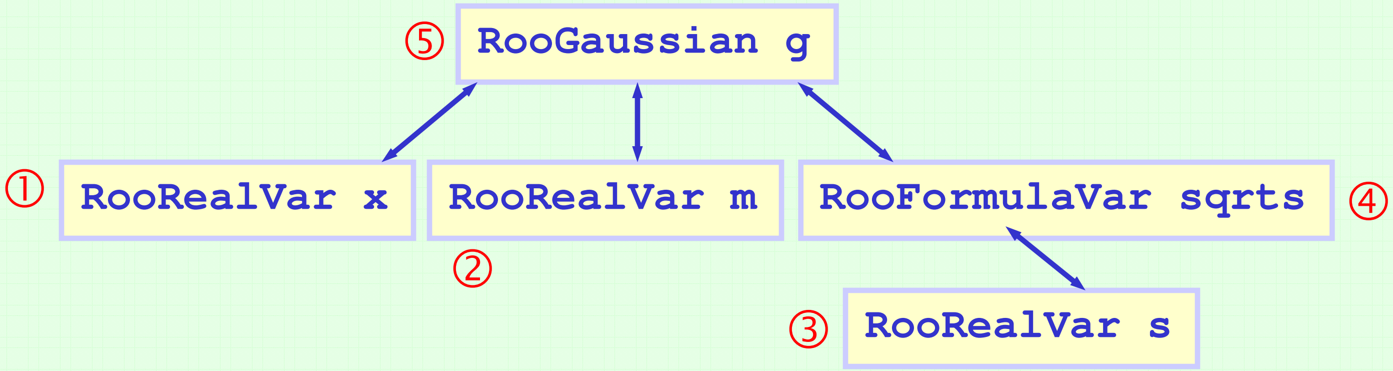
Data modeling – Constructing composite objects

- Straightforward correlation between mathematical representation of formula and RooFit code

Math

$$G(x,m,\sqrt{s})$$

RooFit diagram

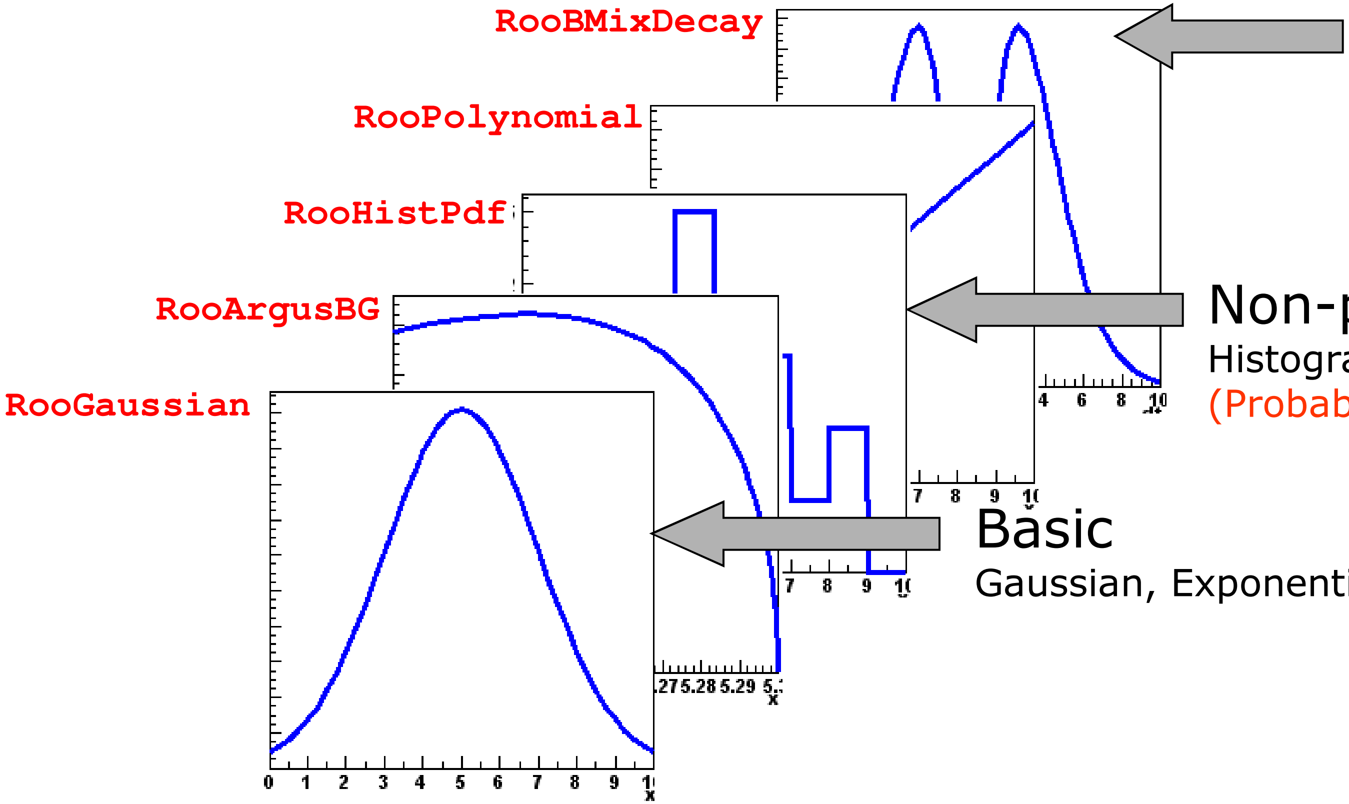


RooFit code

```
① RooRealVar x(“x”,”x”,-10,10) ;
② RooRealVar m(“m”,”mean”,0) ;
③ RooRealVar s(“s”,”sigma”,2,0,10) ;
④ RooFormulaVar sqrts(“sqrts”,”sqrt(s)”,s) ;
⑤ RooGaussian g(“g”,”gauss”,x,m,sqrts) ;
```

Model building – (Re)using standard components

- RooFit provides a collection of compiled standard PDF classes



Physics inspired
ARGUS,Crystal Ball,
Breit-Wigner, Voigtian,
B/D-Decay,….

Non-parametric
Histogram, KEYS
(Probability Density Estimate)

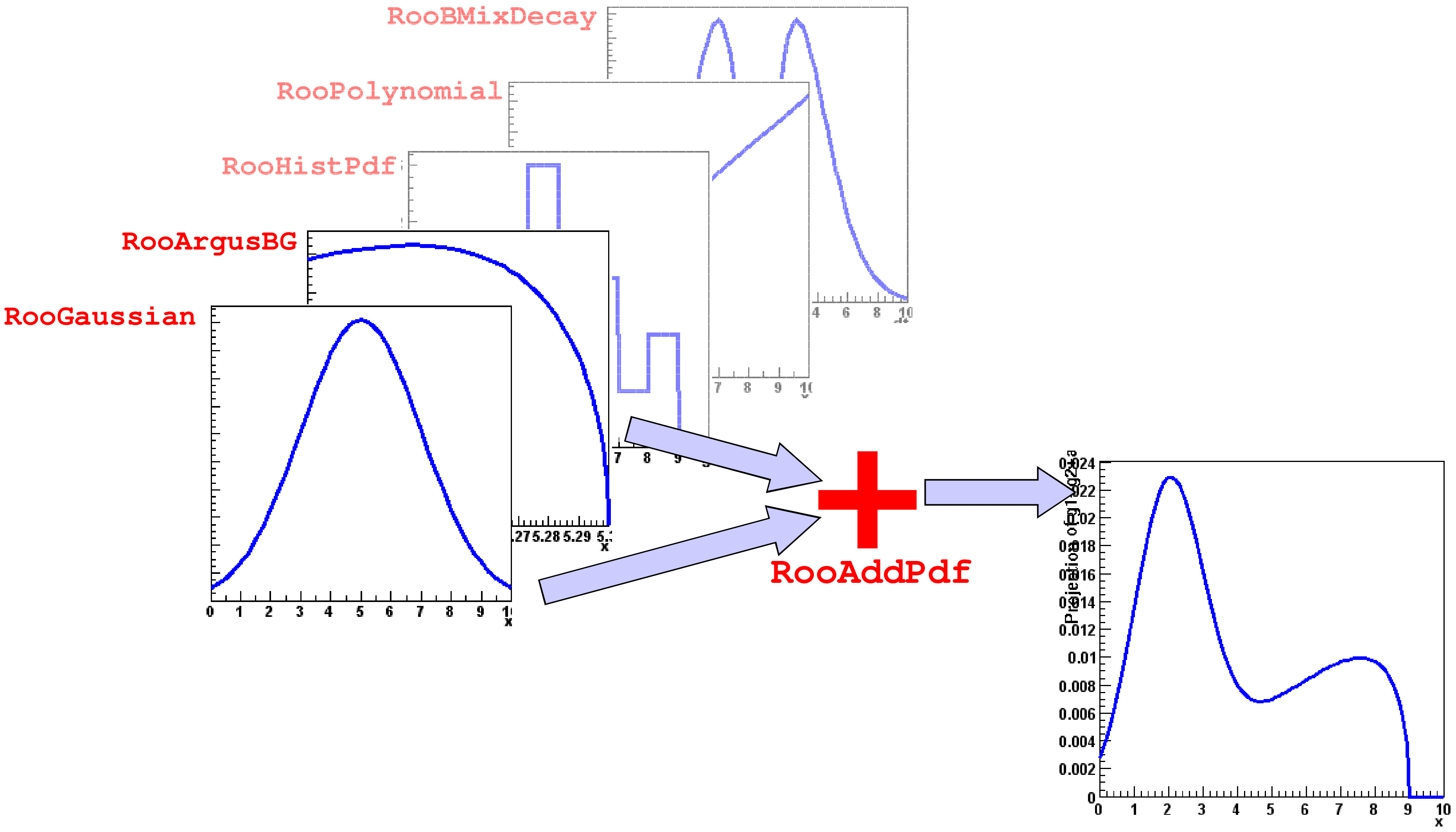
Basic
Gaussian, Exponential, Polynomial,…

PDF Normalization

- By default RooFit uses numeric integration to achieve normalization
- Classes can optionally provide (partial) analytical integrals
- Final normalization can be hybrid numeric/analytic form

Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes



RooBMixDecay

RooPolynomial

RooHistPdf

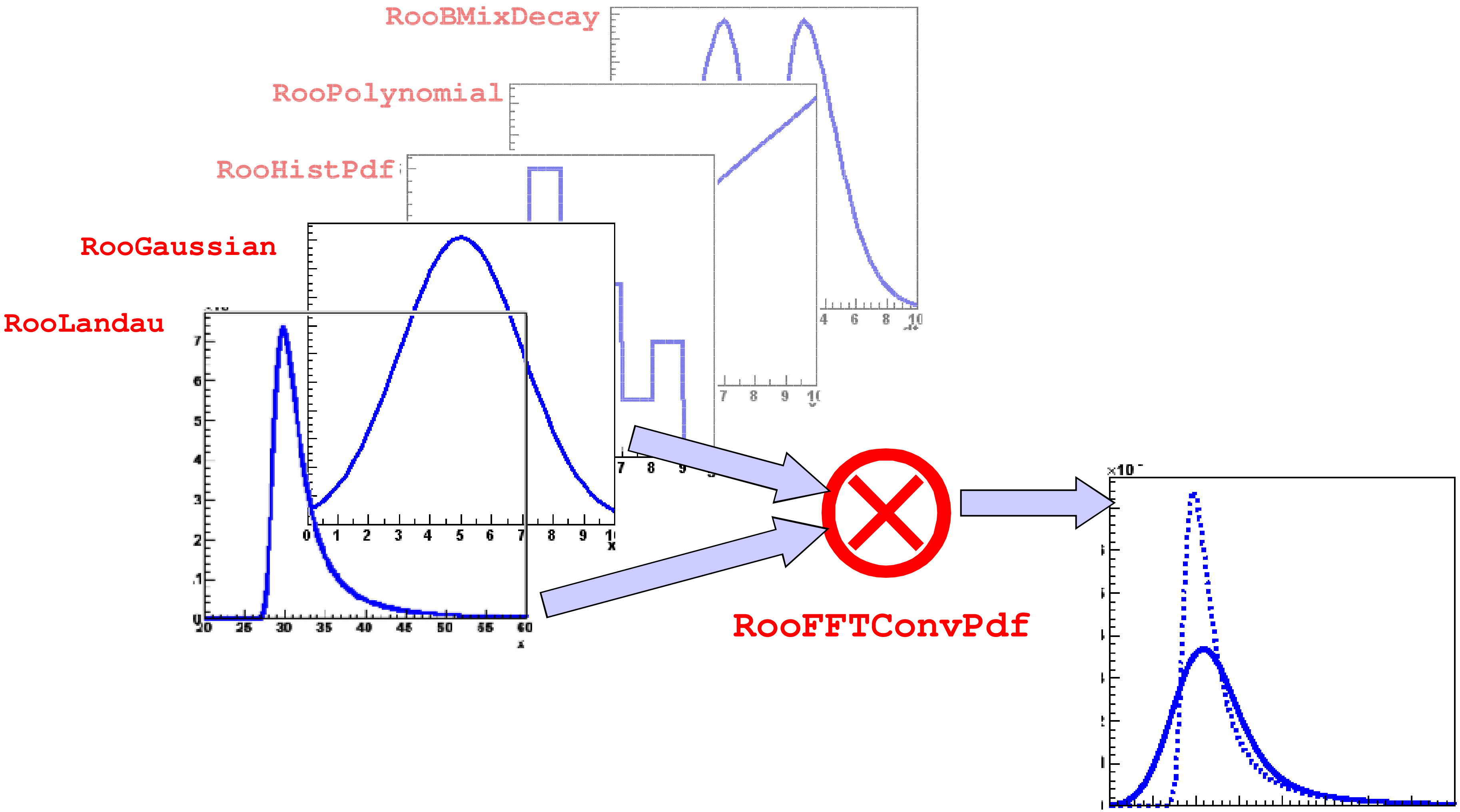
RooArgusBG

RooGaussian

RooAddPdf

Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes



RooBMixDecay

RooPolynomial

RooHistPdf

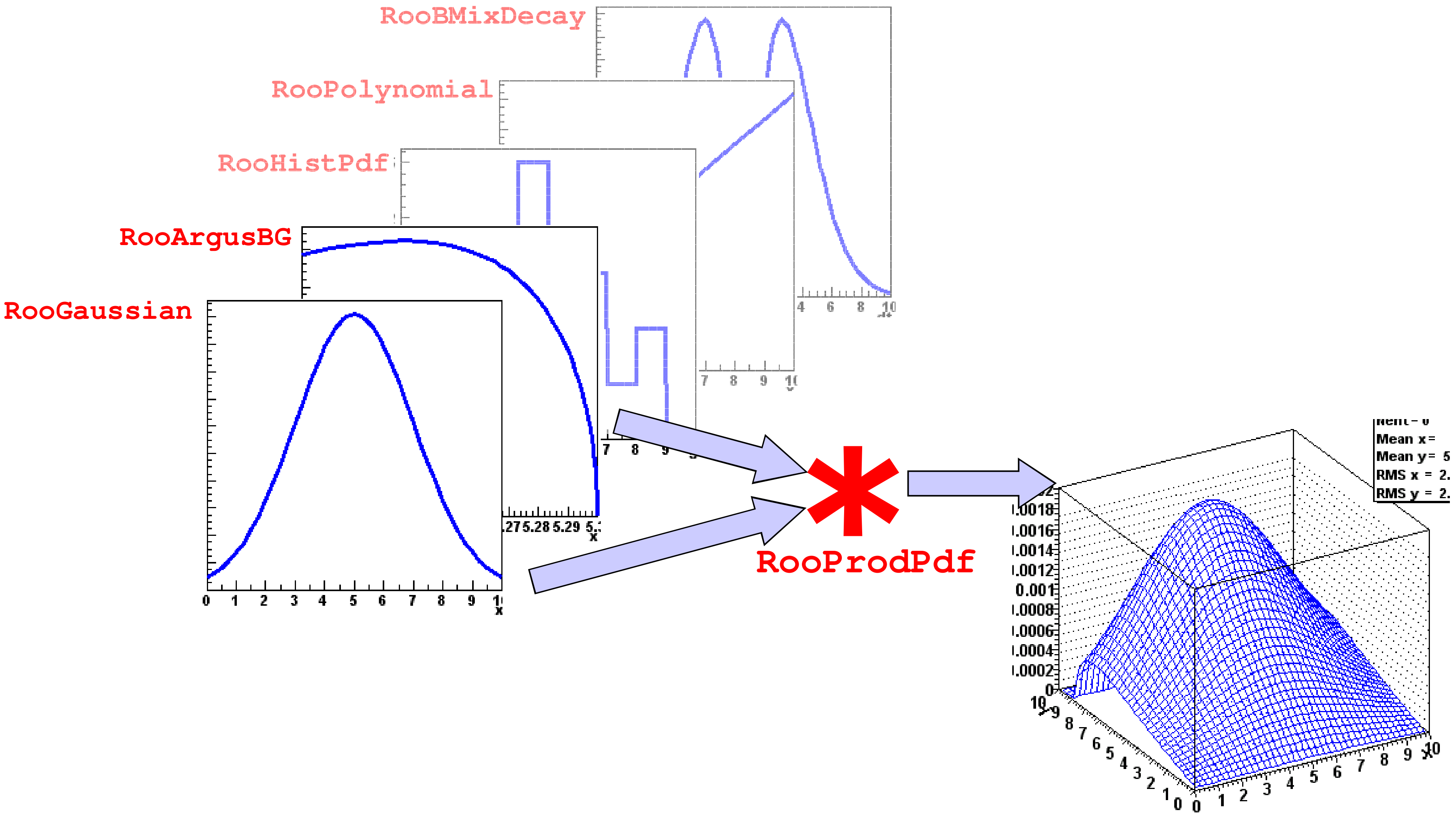
RooGaussian

RooLandau

RooFFTConvPdf

Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes



RooBMixDecay

RooPolynomial

RooHistPdf

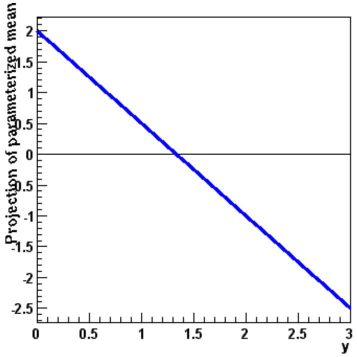
RooArgusBG

RooGaussian

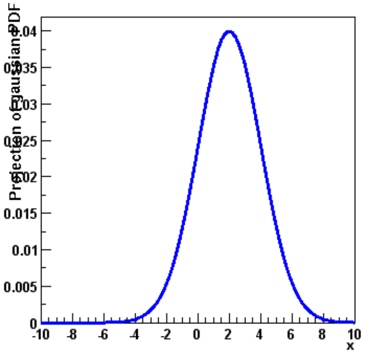
RooProdPdf

Model building – (Re)using standard components

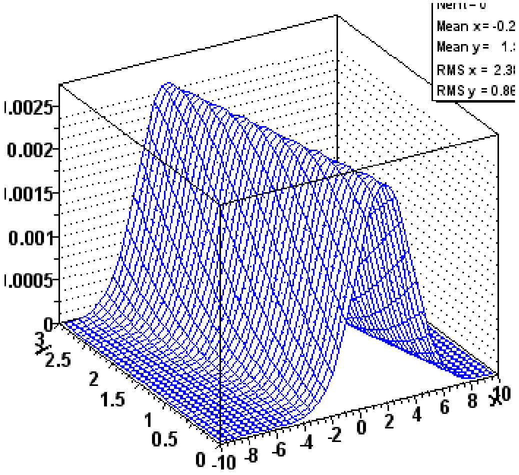
- Building blocks are *flexible*
  - Function *variables can be functions* themselves
  - Just plug in *anything* you like
  - Universally supported by core code
    (PDF classes don’t need to implement special handling)



$m(y;a_0,a_1)$



$g(x;m,s)$



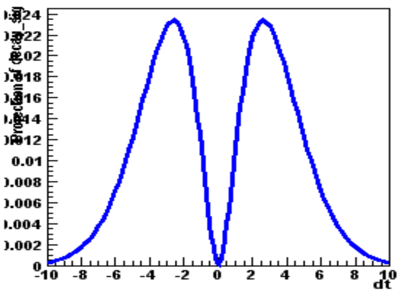
$g(x,y;a_0,a_1,s)$

```
RooPolyVar  m("m",y,RooArgList(a0,a1)) ;
RooGaussian g("g","gauss",x,m,s) ;
```

Wouter Verkerke, UCSB

Model building – Expression based components

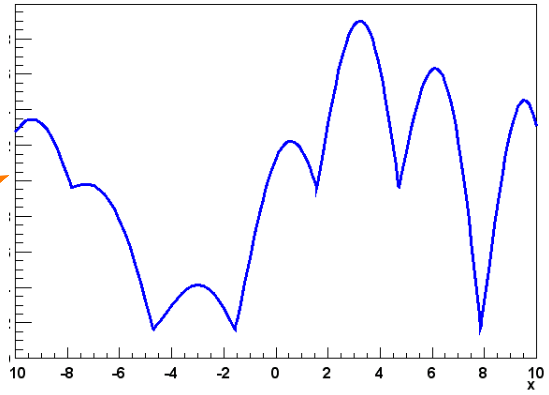
- **`RooFormulaVar`** – Interpreted real-valued function
  - Based on ROOT **`TFormula`** class
  - Ideal for modifying parameterization of existing compiled PDFs



RooBMixDecay(t,tau,w,…)

RooFormulaVar w(“w”,”1-2*D”,D) ;

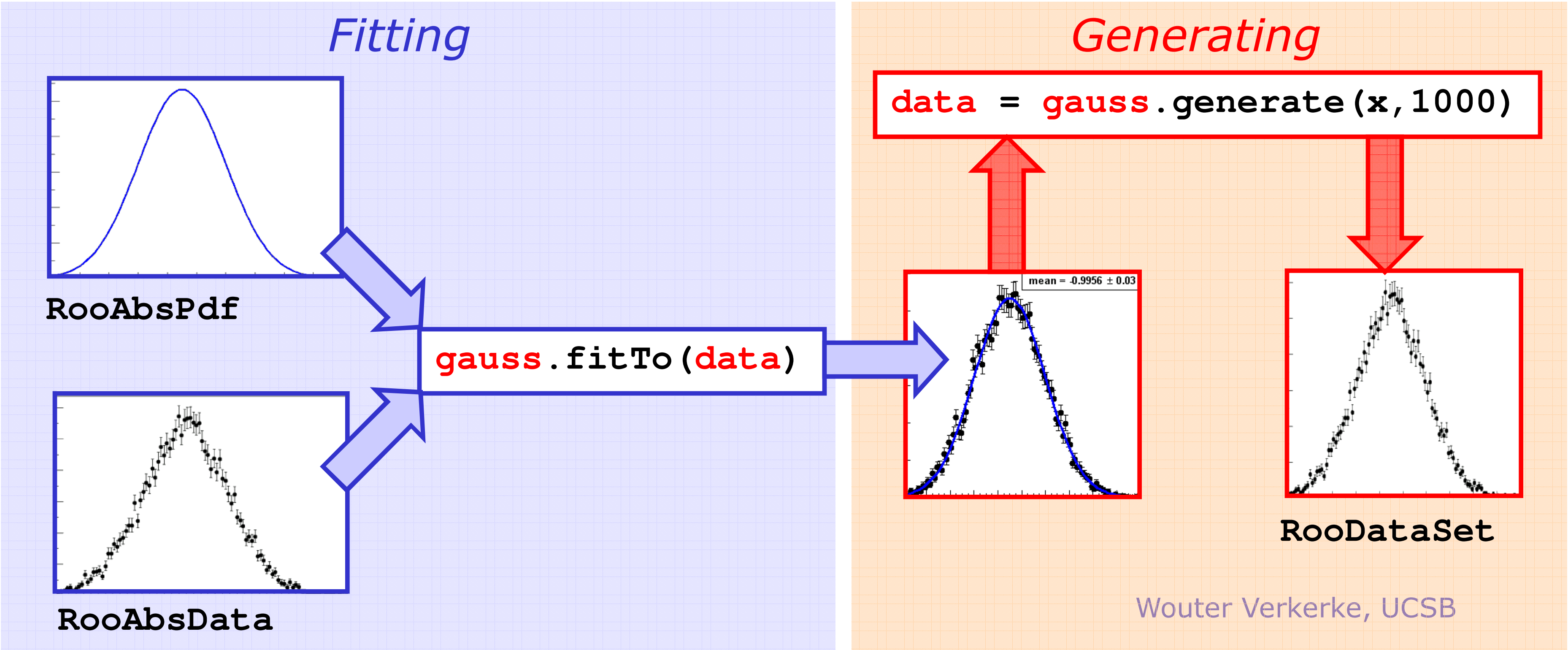
- **`RooGenericPdf`** – Interpreted PDF
  - Based on ROOT **`TFormula`** class
  - User expression doesn’t need to be normalized
  - Maximum flexibility



```
RooGenericPdf f("f","1+sin(0.5*x)+abs(exp(0.1*x)*cos(-1*x))",x)
```

Using models - Overview

- *All* RooFit models provide *universal and complete fitting* and Toy Monte Carlo *generating* functionality
  - Model complexity only limited by available memory and CPU power
    - models with >16000 components, >1000 fixed parameters and>80 floating parameters have been used (published physics result)
  - Very easy to use – Most operations are one-liners



Fitting

RooAbsPdf

RooAbsData

```
gauss.fitTo(data)
```

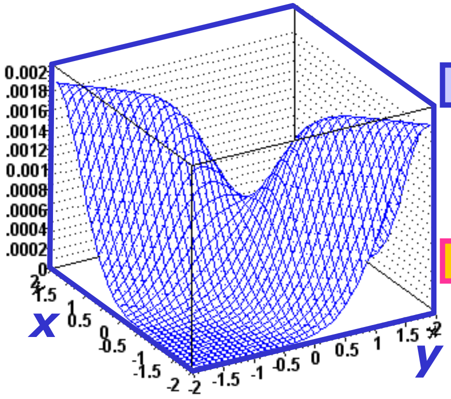
Generating

```
data = gauss.generate(x,1000)
```

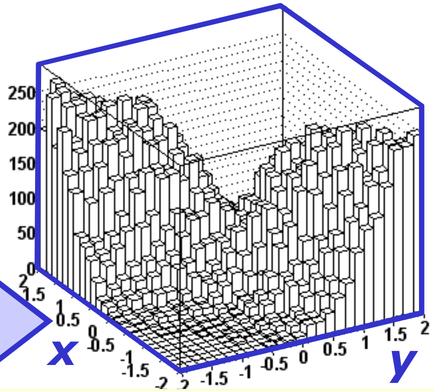
RooDataSet

Using models – Toy MC Generation

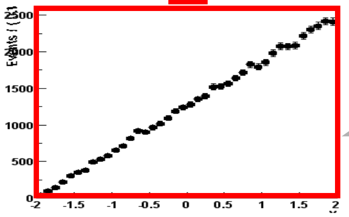
- Generate "Toy" Monte Carlo samples from *any* PDF
  - Sampling method used by default, but PDF components can advertise alternative (more efficient) generator methods
  - No limit to number of dimensions, discrete-valued dimensions also supported

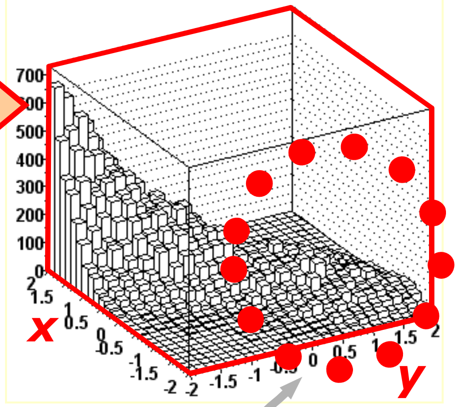


```
data=pdf.generate(x,y,1000)
```



```
data=pdf.generate(x,ydata)
```

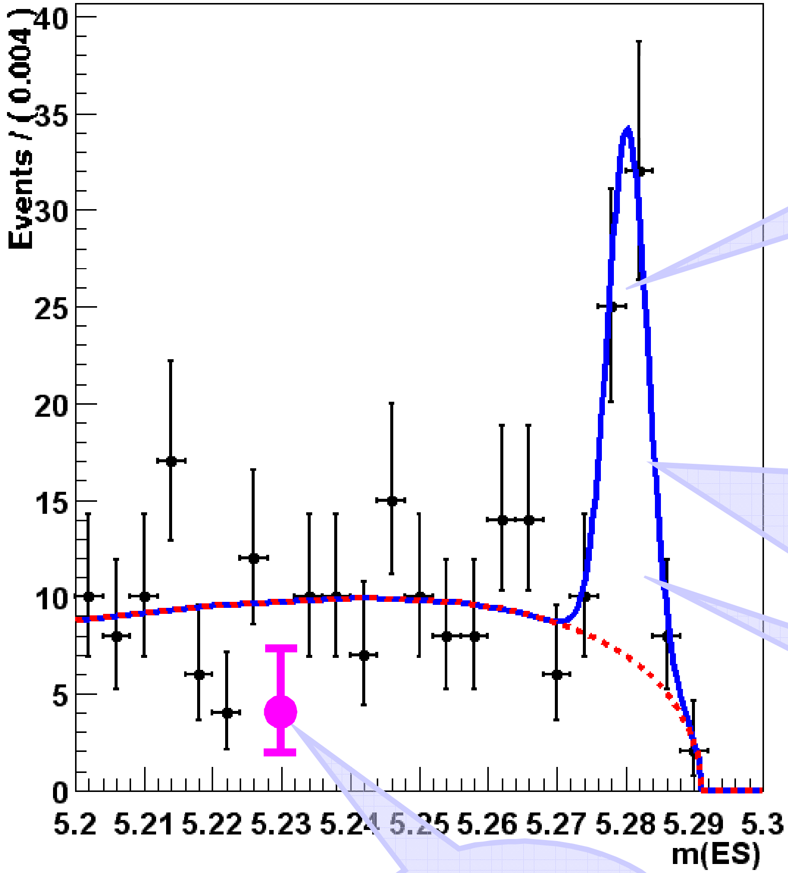




- Subset of variables can be taken from a prototype dataset
  - E.g. to more accurately model the statistical fluctuations in a particular sample.
  - ***Correlations*** with prototype observables ***correctly taken into account***

Using models – Plotting

- `RooPlot` – View of ≥1 datasets/PDFs projected on the same dimension



Curve always normalized to last plotted dataset in `frame`

For multi-dimensional PDFs: appropriate 1-dimensional projection is automatically created:

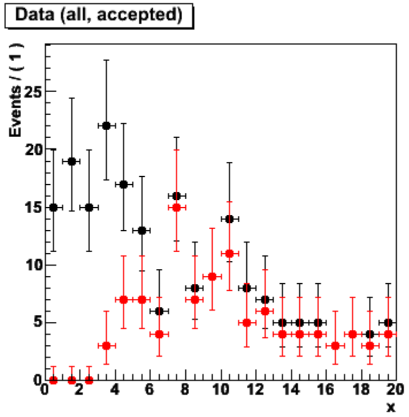
$$\text{Projection}[F](x) = N \cdot \frac{\int F(x, \vec{y})\, d\vec{y}}{\int F(x, \vec{y})\, dx\, d\vec{y}}$$

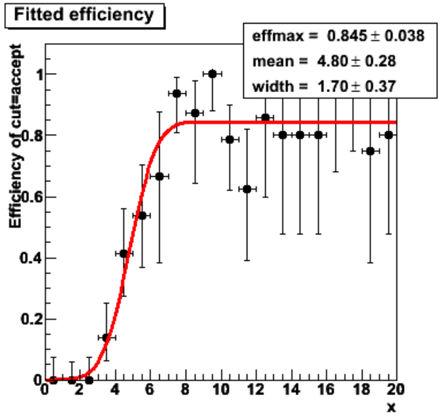
Poisson errors on histogram

Adaptive spacing of curve points to achieve 1‰ precision, regardless of data binning

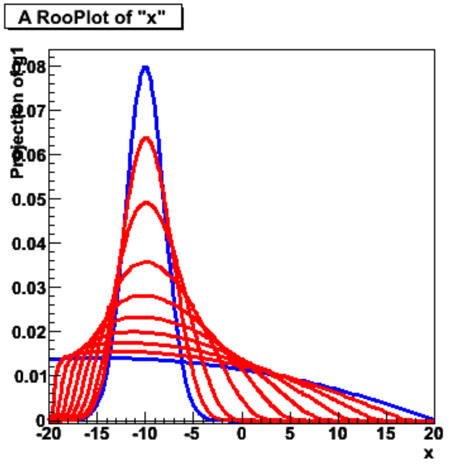
Many default solutions for standard problems

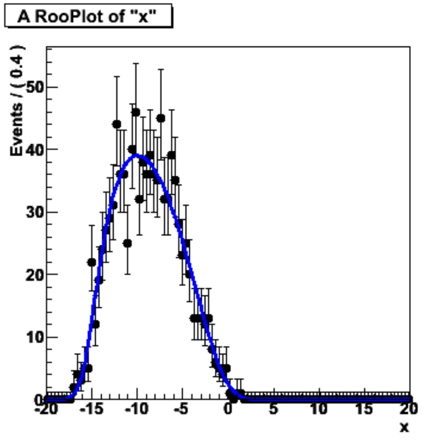
- Unbinned ML fit of efficiency curves
  - Example: trigger threshold





- Template interpolation
  - Example: morph polynomial into Gaussian
  - Realistic use case: interpolate full MC Higgs signal between masses of e.g. 160 GeV and 180 GeV





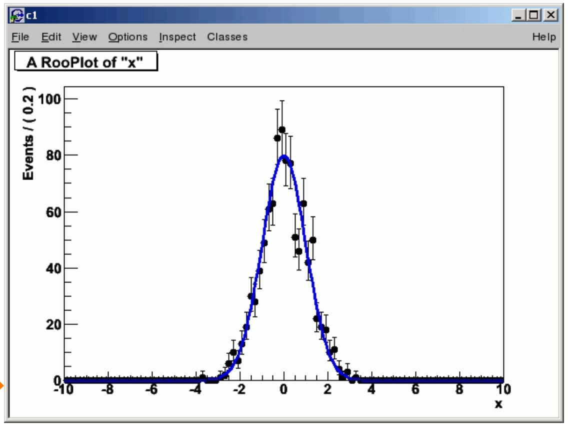
Persisting models in the workspace

- Using both model & p.d.f from file

```
TFile f(“myresults.root”) ;
RooWorkspace* w = f.Get(“w”) ;
```

Make plot of data and p.d.f

```
RooPlot* xframe = w->var(“x”)->frame() ;
w->data(“d”)->plotOn(xframe) ;
w->pdf(“g”)->plotOn(xframe) ;
```

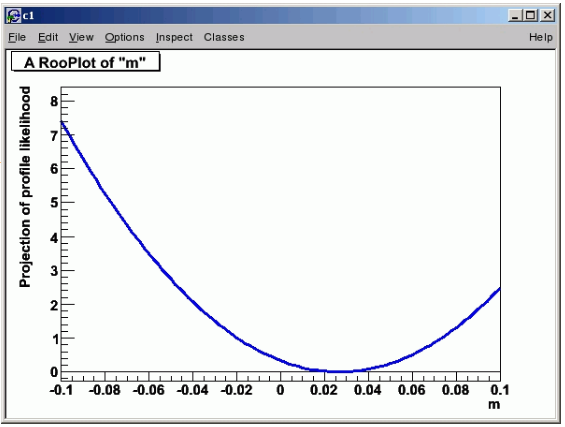


Construct likelihood & profile LH

```
RooNLLVar nll(“nll”,”nll”,*w->pdf(“g”),*w->data(“d”)) ;
RooProfileLL pll(“pll”,”pll”, nll,*w->var(“m”)) ;
```

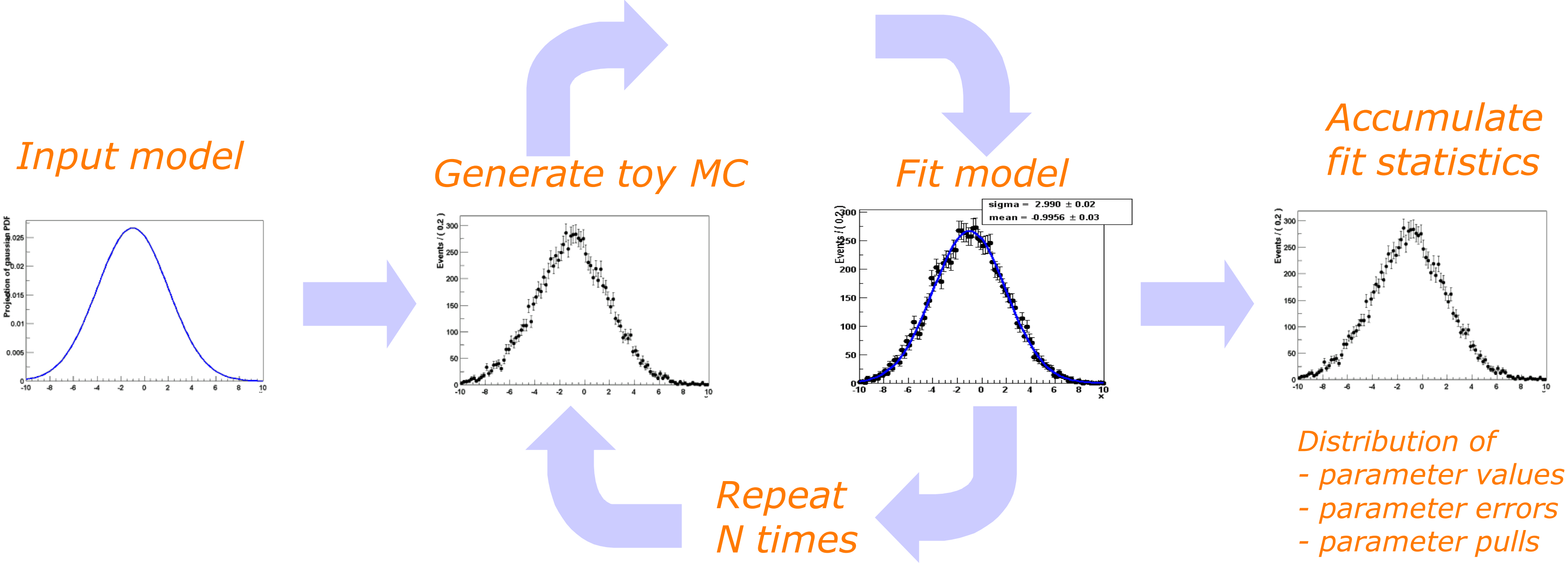
Draw profile LH

```
RooPlot* mframe = w->var(“m”)->frame(-1,1) ;
pll.plotOn(mframe) ;
mframe->Draw()
```


c1
File Edit View Options Inspect Classes Help
A RooPlot of "m"
Projection of profile likelihood
8
7
6
5
4
3
2
1
0
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08
0.1
m

Advanced features – Task automation

- Support for routine task automation, e.g. goodness-of-fit study



Input model

Generate toy MC

Fit model

Accumulate fit statistics

Repeat N times

Distribution of
- parameter values
- parameter errors
- parameter pulls

```
// Instantiate MC study manager
RooMCStudy mgr(inputModel) ;

// Generate and fit 100 samples of 1000 events
mgr.generateAndFit(100,1000) ;

// Plot distribution of sigma parameter
mgr.plotParam(sigma)->Draw()
```