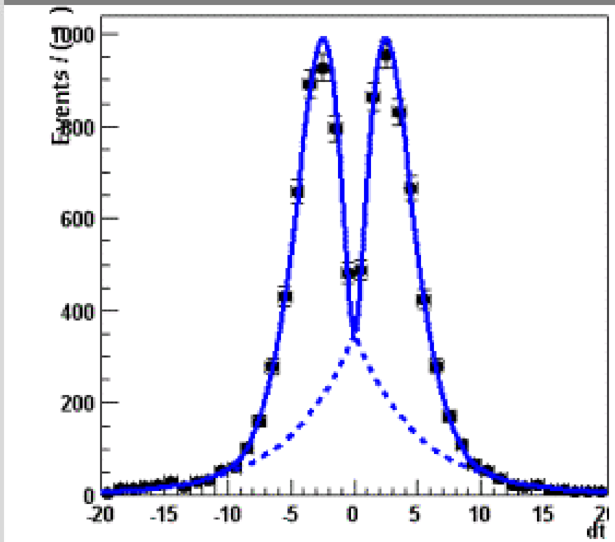
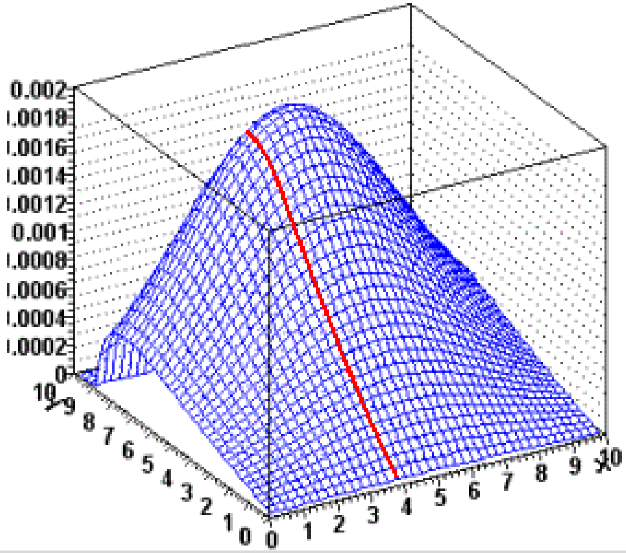
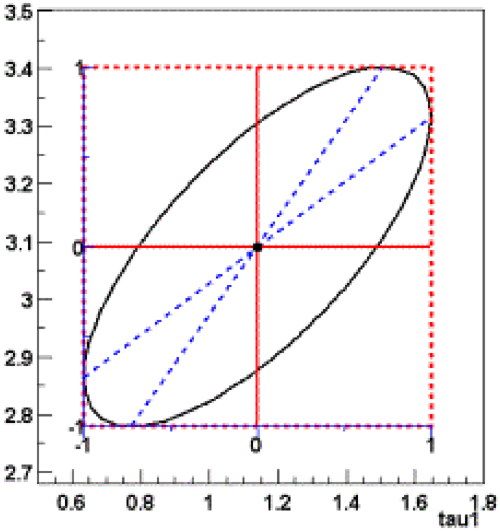
Introduction to RooFit

Lukas Hehn
KSETA PhD Workshop Freudenstadt, October 16th to 18th 2003

Lukas Hehn, Institut für Kernphysik, KIT







RooFit ...

- … is a library which provides a toolkit for data analysis
- … is included in ROOT framework
- … is used to model expected event distributions in physics analysis
- … can perform (un)binned maximum likelihood fits, produce plots and study goodness-of-fit with toy Monte Carlo samples
- … was originally developed for the BaBar collaboration @ Stanford Linear Accelerator Center

To use RooFit in ROOT CINT

- Load library as:

```
gSystem->Load("libRooFit") ;
using namespace RooFit ;
```

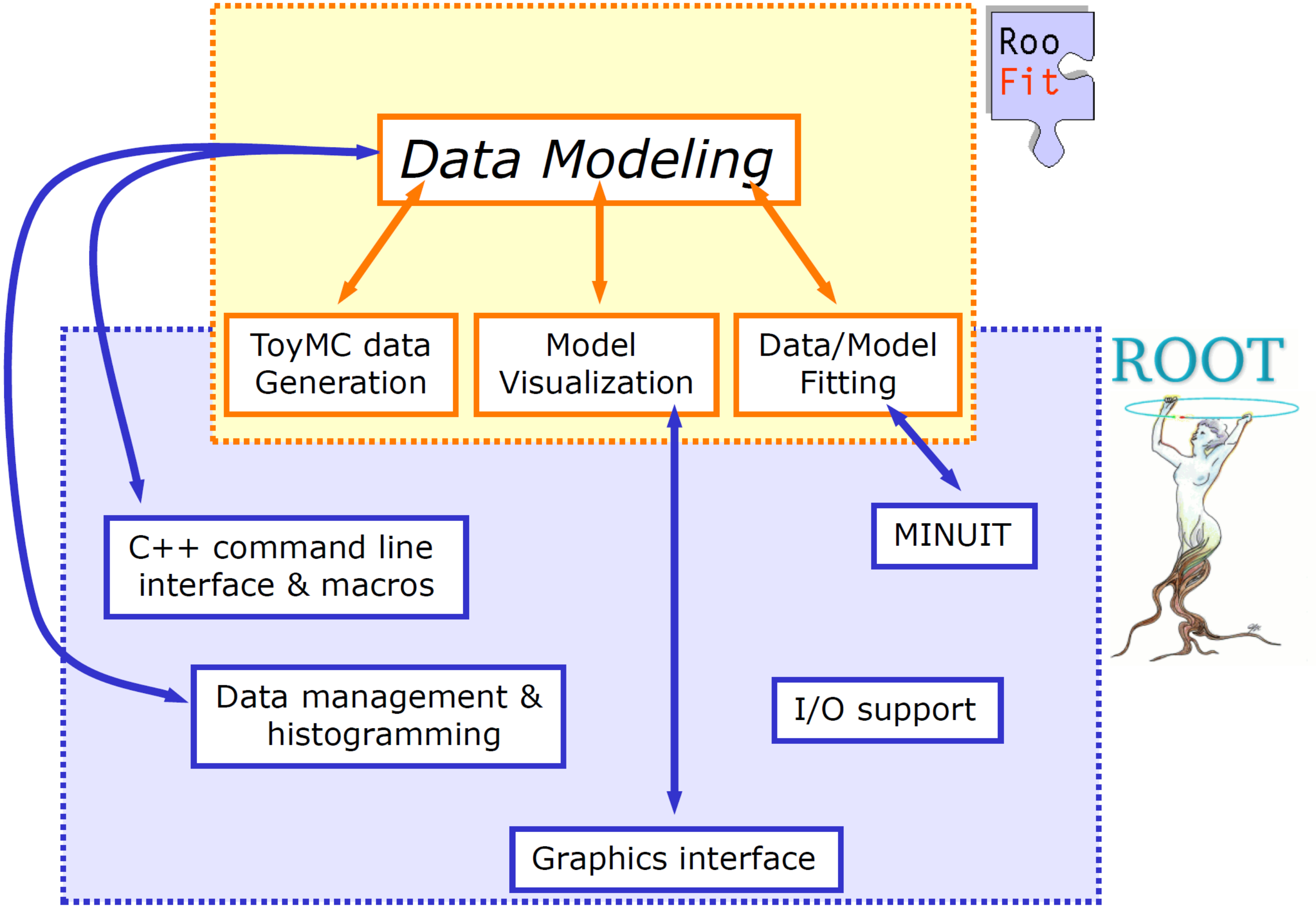
OR

- Load prepared macro file

```
.x path-to-file
```

RooFit & ROOT

- RooFit library comes with and depends on ROOT



Principles of maximum likelihood estimation

- you have a data set D(x) with observables x (i.e. x & y or Energy & time)
- possible to construct an estimator: Likelihood function ***L***

$$L(\vec{p})=\prod_{n=0}^{N} F(\vec{p},\vec{x}_n)\cdot \underbrace{Poisson(N_{\exp},N_{obs})}$$

(for extended ML only)

- with *probability density function* (PDF) F:

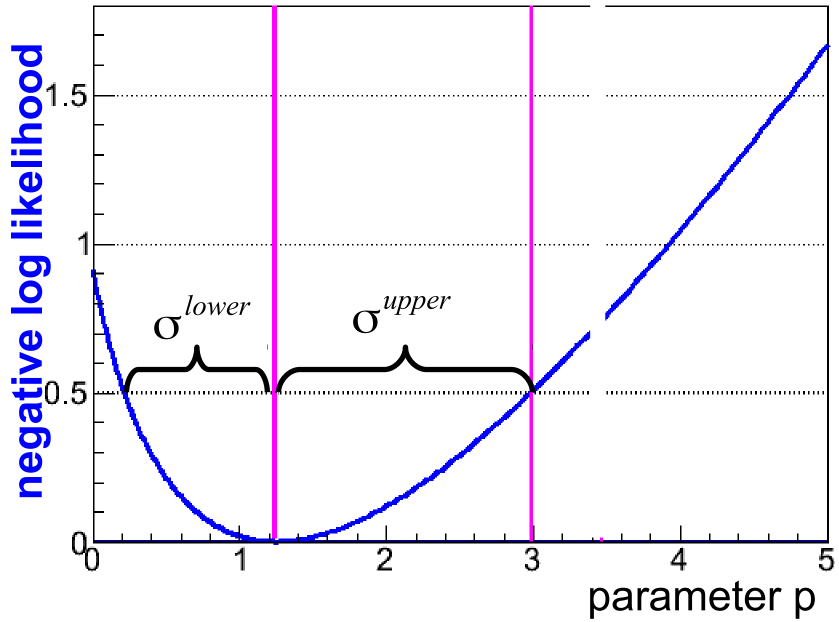
$$\int F(\vec{x};\vec{p})\,d\vec{x}\equiv 1, \qquad F(\vec{x};\vec{p})>0$$

- best fit parameters p given by maximizing likelihood ***L* or** minimizing *negative log likelihood* (***NLL***)

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

- estimator of the parameter variance:

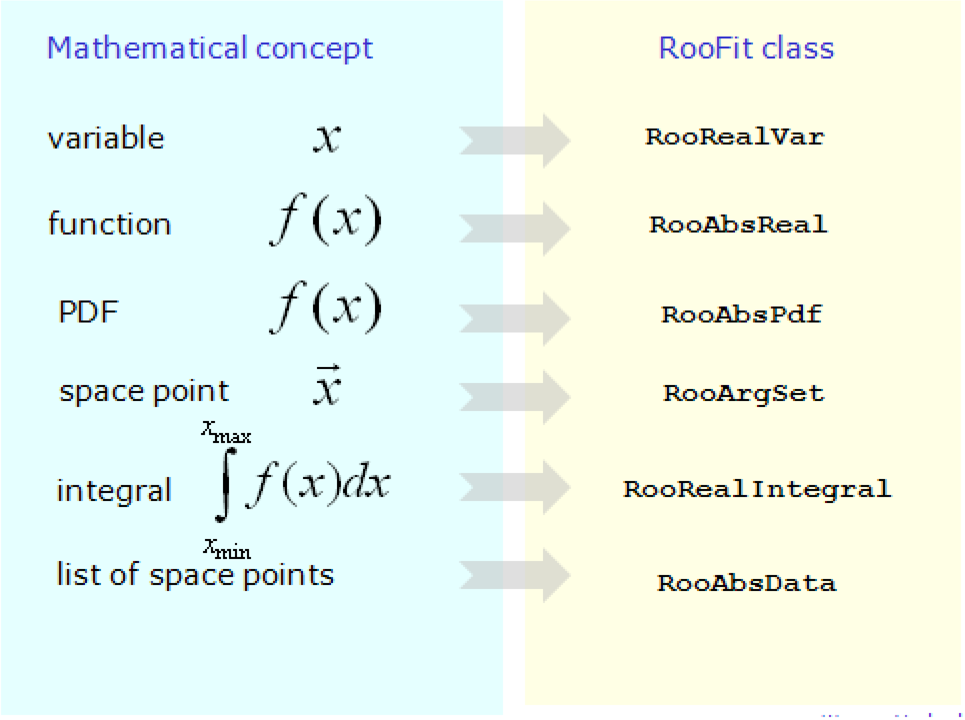
$$-\ln L(p\pm\sigma)=-\ln L_{min}+0.5$$



Principle of RooFit

you define everything with RooFit classes:

- your PDF-model
- your data and its observables
- the parameter in your PDF you want to fit (and all other parameters)
- the likelihood function you want to minimize



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

Available documentation

- Official Websites
  - http://root.cern.ch/drupal/content/roofit
  - http://roofit.sourceforge.net/
- **Class documentation:** http://root.cern.ch/root/html/ROOFIT_ROOFITCORE_Index.html
- **Tutorial macros** (83)
  - http://root.cern.ch/root/html/tutorials/roofit/index.html
  - `$ROOTSYS/tutorials/roofit`
- **User Manual** 134 pages from 2008
- **Conference Talk**: Strasbourg School of Statistics 2008 (200 slides) http://dx.doi.org/10.1051/epjconf/20100402005
- Conference Procedings: Wouter Verkerke, David Kirkby: *"The RooFit toolkit for data modeling"* (arXiv:0306116)
- Quick Start Guide: 24 pages from 2009 http://root.cern.ch/drupal/sites/default/files/roofit_quickstart_3.00.pdf

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

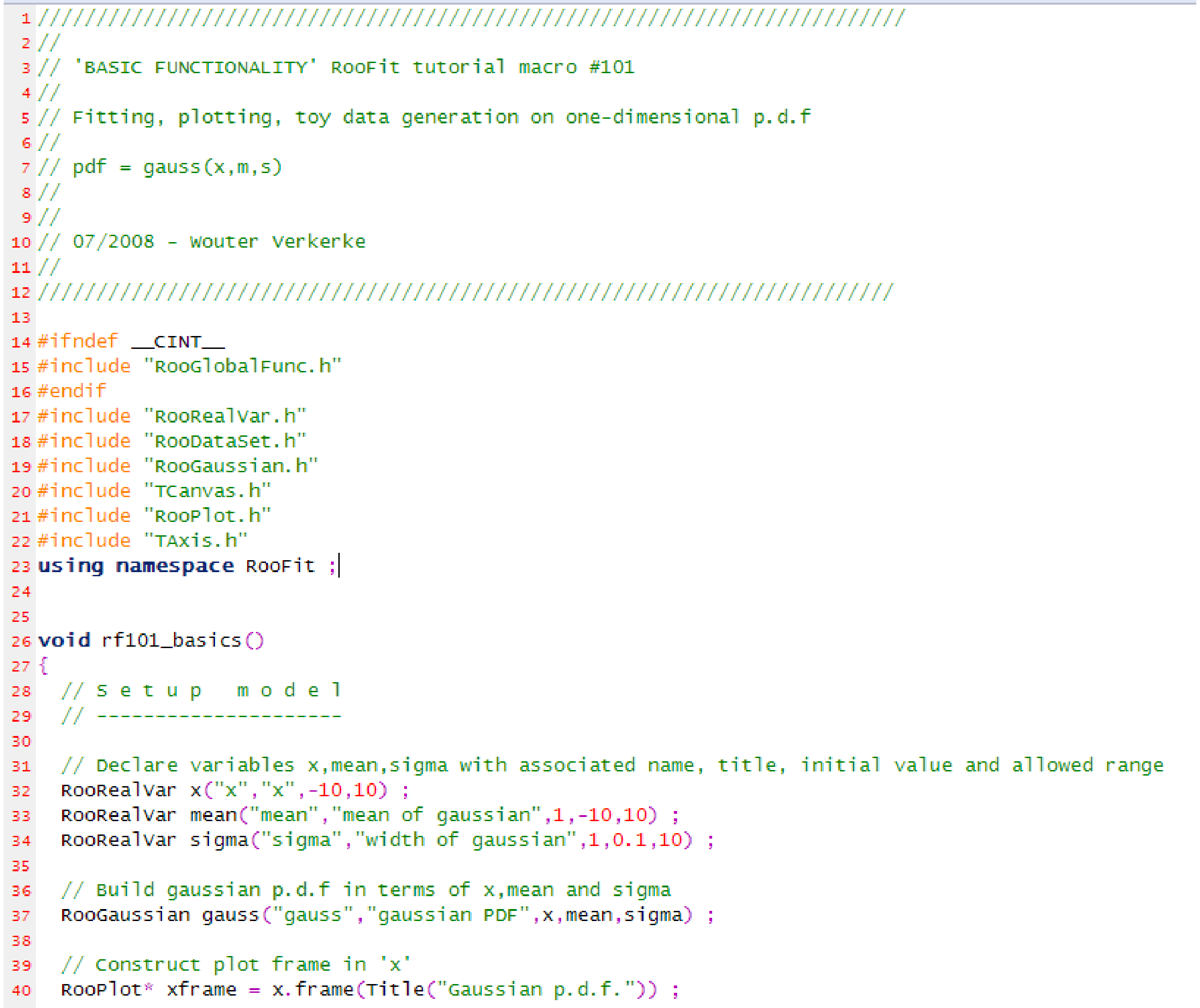
```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

Tutorial macro rf101_basics.C



```
///////////////////////////////////////////////////////////////////////////
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #101
//
// Fitting, plotting, toy data generation on one-dimensional p.d.f
//
// pdf = gauss(x,m,s)
//
//
// 07/2008 - Wouter Verkerke
//
//////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "TCanvas.h"
#include "RooPlot.h"
#include "TAxis.h"
using namespace RooFit ;


void rf101_basics()
{
  // S e t u p   m o d e l
  // ---------------------

  // Declare variables x,mean,sigma with associated name, title, initial value and allowed range
  RooRealVar x("x","x",-10,10) ;
  RooRealVar mean("mean","mean of gaussian",1,-10,10) ;
  RooRealVar sigma("sigma","width of gaussian",1,0.1,10) ;

  // Build gaussian p.d.f in terms of x,mean and sigma
  RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma) ;

  // Construct plot frame in 'x'
  RooPlot* xframe = x.frame(Title("Gaussian p.d.f.")) ;
```

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

1. Defining variables

- variables are defined as

```
RooRealVar("name", "title", value, minValue, maxValue, "unit")
```

construct with either a fixed value / or a range / or starting value + range

- observables (i.e. x, y, energy, time) and parameters of a PDF (i.e. mean, sigma, slope) are both variables
  → the data set "tells" a PDF what it's observable is
  → all other variables must be parameters
- when fitting a PDF model to data: all free floating (= not fixed) parameters are fitted
- you can later on define and exclude a parameter from being fitted by the method `RooRealVar.setValue(value)` **and** `RooRealVar.setConstant()`
- construct flexible variable:

```
RooFormulaVar
mean_shifted("mean_shifted","@0+@1",RooArgList(mean,shift))
```

ROOT TFormula expression RooRealVar's

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

2. About PDFs

- construction of PDF is one of the most important steps
- bad PDF → bad fit
- the PDF contains the parameters which are fitted:
  this can either be parameters defining the shape of a PDF (like decay constant, Gaussian width, …) or often fractions of different PDF components (i.e. signal vs. background component)
- PDFs are automatically normalized within RooFit

Build in PDFs

~20 predefined PDFs to build models from

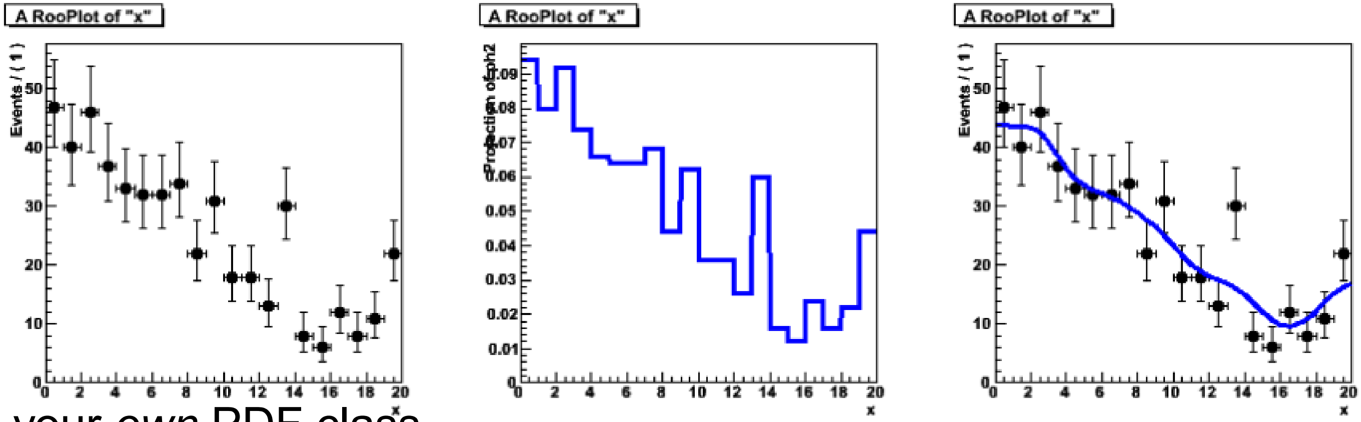
- Basic functions:
  - `RooGaussian`: normal Gaussian
  - `RooBifurGauss`: different width on low and high side of mean
  - `RooExponential`: standard exponential decay
  - `RooPolynomial`: standard polynoms
  - `RooChebychev`: Chebychev polynomials (recommended because of higher fit stability due to little correlation)
  - `RooPoisson`: Poisson distribution
- Physics inspired functions:
  - Landau (`RooLandau`), Breit-Wigner, Crystal Ball, …
- Specialized functions for B physics:
  - Decay distributions with mixing, CP violation, …

all one parameter less than normal because for a PDF → integral != 1

More on PDFs

- Other non-parametric functions:
  - **`RooHistPdf`**: from external ROOT histogram, optional interpolation for smoothing
  - **`RooKeysPdf`**: Kernel estimation, superposition of Gaussians on external unbinned data

Data (N=500)   RooHistPdf(data)   RooKeysPdf(data)



- Writing your *own* PDF class
  - from a formula expression:

```
RooGenericPdf gp(“gp”,”Generic PDF”,”exp(x*y+a)-b*x”,
RooArgSet(x,y,a,b)) ;
```

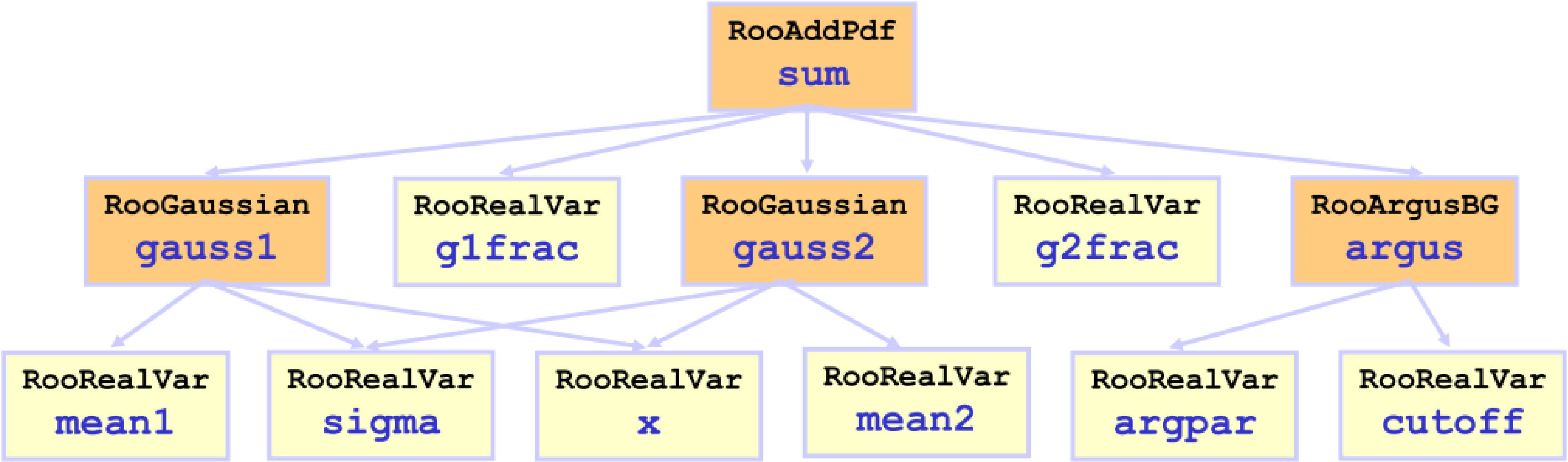
  - **`RooClassFactory`** to write and compile own C++ code for PDFs

Composite PDF models

- realistic models are often a sum of multiple PDFs, i.e.
  Gaussian signal + flat background
- class `RooAddPdf` adds *N* PDFs with (*N-1*) `RooRealVar` fraction coefficients

$$S = c_0 P_0 + c_1 P_1 + c_2 P_2 + \ldots + c_{n-1} P_{n-1} + \left(1 - \sum_{i=0,n-1} c_i\right) P_n$$

  - caveat: total PDF can become negative in some cases!
- all methods work normally on such a PDF (`fitTo()`, `plotOn()`, ...)
- exemplary tree view of such a PDF



RooAddPdf
sum

RooGaussian
gauss1

RooRealVar
g1frac

RooGaussian
gauss2

RooRealVar
g2frac

RooArgusBG
argus

RooRealVar
mean1

RooRealVar
sigma

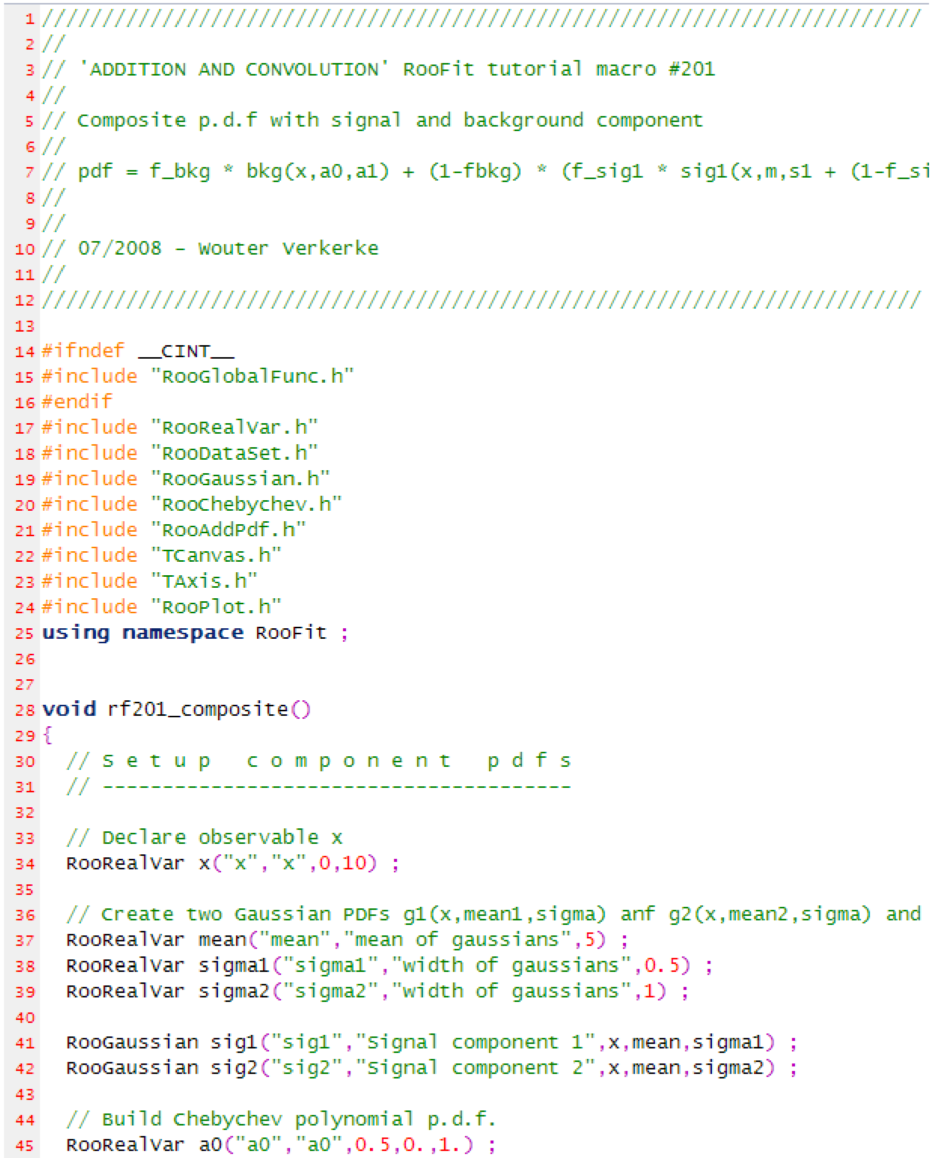
RooRealVar
x

RooRealVar
mean2

RooRealVar
argpar

RooRealVar
cutoff

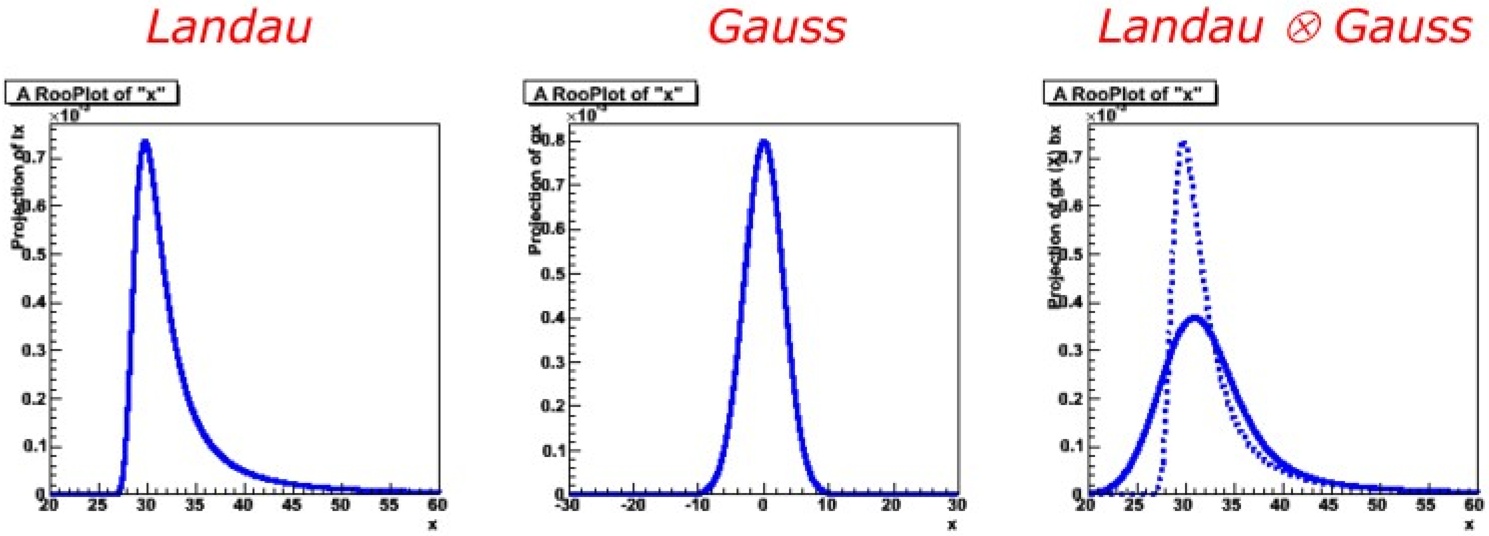
Tutorial macro rf201_composite.C



```
//////////////////////////////////////////////////////////////////////////
//
// 'ADDITION AND CONVOLUTION' RooFit tutorial macro #201
//
// Composite p.d.f with signal and background component
//
// pdf = f_bkg * bkg(x,a0,a1) + (1-fbkg) * (f_sig1 * sig1(x,m,s1 + (1-f_si
//
//
// 07/2008 - Wouter Verkerke
//
//////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooChebychev.h"
#include "RooAddPdf.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "RooPlot.h"
using namespace RooFit ;


void rf201_composite()
{
  // S e t u p   c o m p o n e n t   p d f s
  // ---------------------------------------

  // Declare observable x
  RooRealVar x("x","x",0,10) ;

  // Create two Gaussian PDFs g1(x,mean1,sigma) anf g2(x,mean2,sigma) and
  RooRealVar mean("mean","mean of gaussians",5) ;
  RooRealVar sigma1("sigma1","width of gaussians",0.5) ;
  RooRealVar sigma2("sigma2","width of gaussians",1) ;

  RooGaussian sig1("sig1","Signal component 1",x,mean,sigma1) ;
  RooGaussian sig2("sig2","Signal component 2",x,mean,sigma2) ;

  // Build Chebychev polynomial p.d.f.
  RooRealVar a0("a0","a0",0.5,0.,1.) ;
```

Convoluting PDFs $f(x) \otimes g(x) = \int_{-\infty}^{+\infty} f(x) g(x - x') dx'$

- typical for experiments: expected observable behaviour (*physics*) is smeared with a (Gaussian) resolution function (*detector*)
  → convolution of 2 different PDFs



Landau

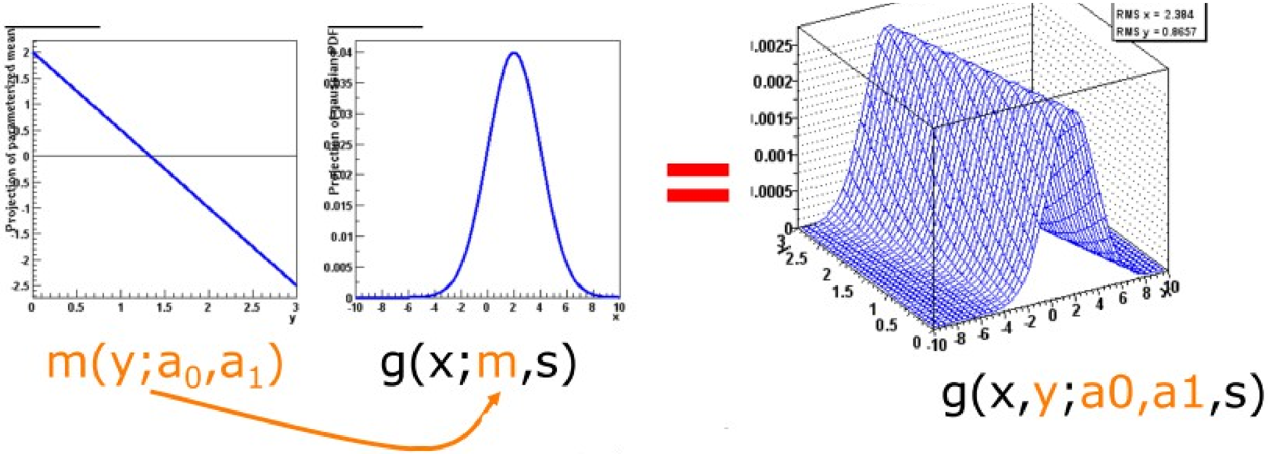
Gauss

Landau ⊗ Gauss

- RooFit offers several different methods to achieve this:
  - `RooNumConv`: brute force numeric convolution
  - `RooFFTConvPdf`: convolution based on fast fourier transformation (FFT)
  - (other predefined particle physics convolutions)

Multidimensional PDF models

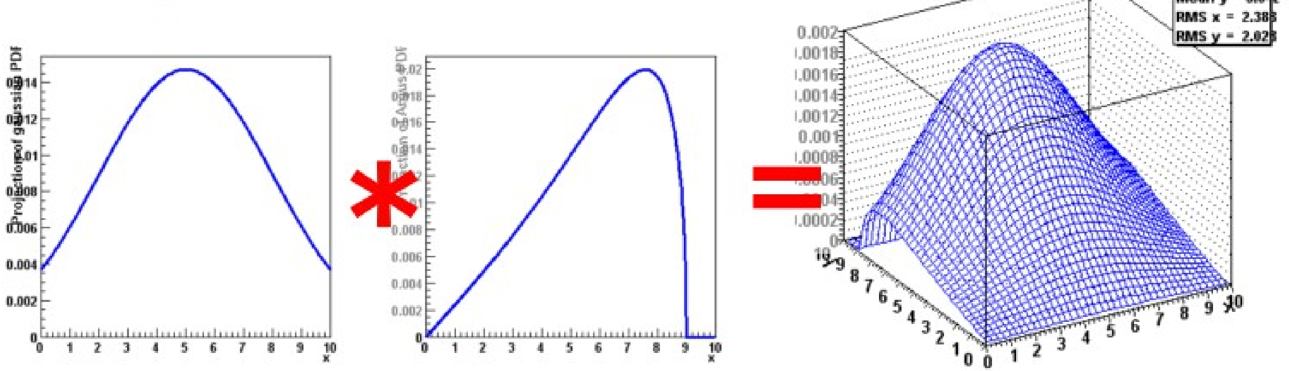
- replace parameter in 1D PDF with another PDF in another observable:



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

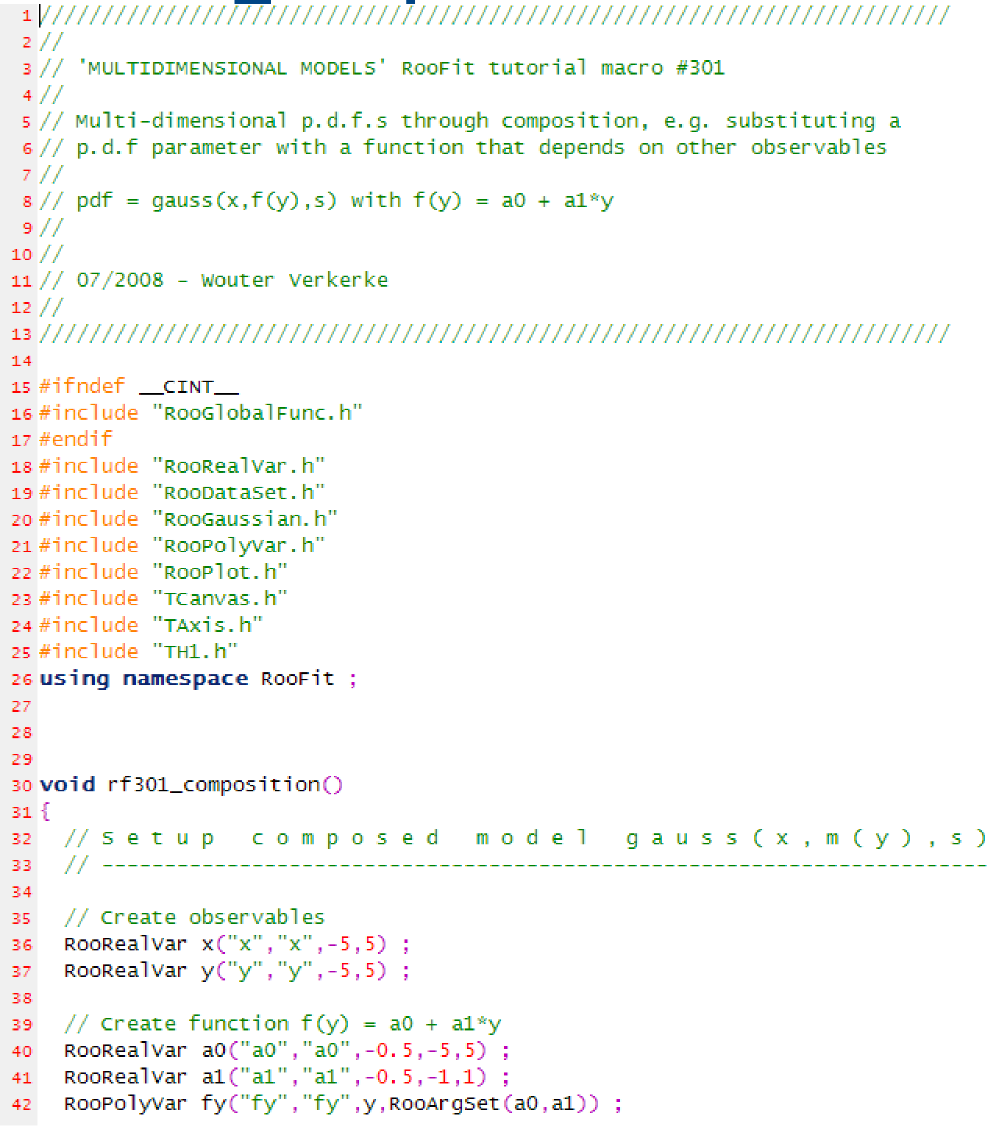
$g(x,y;a0,a1,s)$

- create model for more than 1 Observable (i.e. energy & time, x & y) with `RooProdPdf` class



- with `RooGenericPdf gp(“gp”,”sqrt(x+y)*sqrt(x-y)”,RooArSet(x,y)) ;`

Tutorial macro rf301_composition.C



```
/////////////////////////////////////////////////////////////////////////
//
// 'MULTIDIMENSIONAL MODELS' RooFit tutorial macro #301
//
// Multi-dimensional p.d.f.s through composition, e.g. substituting a
// p.d.f parameter with a function that depends on other observables
//
// pdf = gauss(x,f(y),s) with f(y) = a0 + a1*y
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooPolyVar.h"
#include "RooPlot.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "TH1.h"
using namespace RooFit ;



void rf301_composition()
{
  // S e t u p   c o m p o s e d   m o d e l   g a u s s ( x , m ( y ) , s )
  // -----------------------------------------------------------------------

  // Create observables
  RooRealVar x("x","x",-5,5) ;
  RooRealVar y("y","y",-5,5) ;

  // Create function f(y) = a0 + a1*y
  RooRealVar a0("a0","a0",-0.5,-5,5) ;
  RooRealVar a1("a1","a1",-0.5,-1,1) ;
  RooPolyVar fy("fy","fy",y,RooArgSet(a0,a1)) ;
```

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

3. Datasets

- class RooDataSet is an N-dimension collection of points with continuous RooRealVar or discrete RooCategory observables and optional weights
- for all testing purposes: method `generate(observable,#events)` works on all PDFs (including composite, product, convoluted, ...)
- internally stored as unbinned or binned data in a ROOT TTree object
- importing unbinned data
  - from ASCII files (values in tab seperated columns)

    ```
    RooRealVar x(“x”,”x”,-10,10) ;
    RooRealVar c(“c”,”c”,0,30) ;
    RooDataSet::read(“ascii.txt”,RooArgList(x,c)) ;
    ```

  - from ROOT TTrees

    ```
    RooDataSet data(“data”,”data”,inputTree,RooArgSet(x,c));
    ```

- importing binned data from ROOT THx histograms

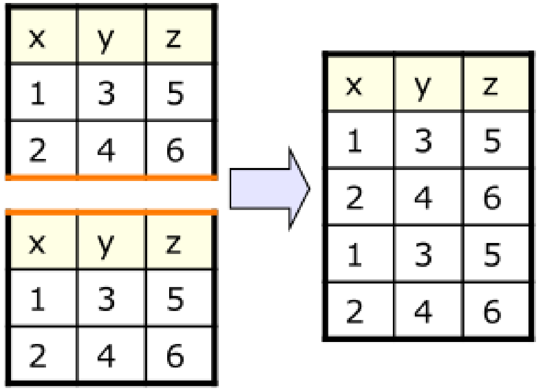
  ```
  RooDataHist bdata2(“bdata”,”bdata”,RooArgList(x,y),histo2d);
  ```

- manual filling with `dataset.add(RooArgSet(x,c))`

only values which are in observable range are imported

Operations on unbinned data sets

- Appending:
  `d1.append(d2)`

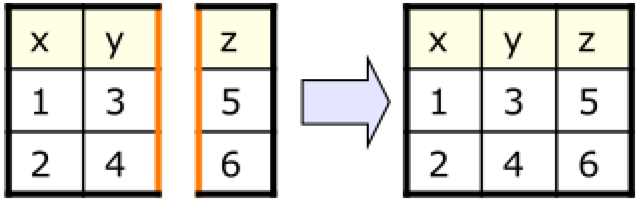


x	y	z
1	3	5
2	4	6

x	y	z
1	3	5
2	4	6

x	y	z
1	3	5
2	4	6
1	3	5
2	4	6

- Merging:
  `d1.merge(d2)`

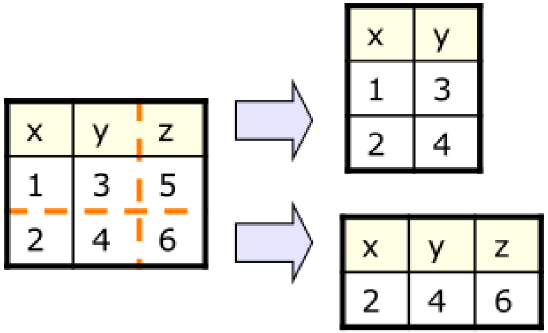


x	y
1	3
2	4

z
5
6

x	y	z
1	3	5
2	4	6

- Reducing
  i.e. `RooDataSet* d2 = d1.reduce(RooArgSet(x,y));`

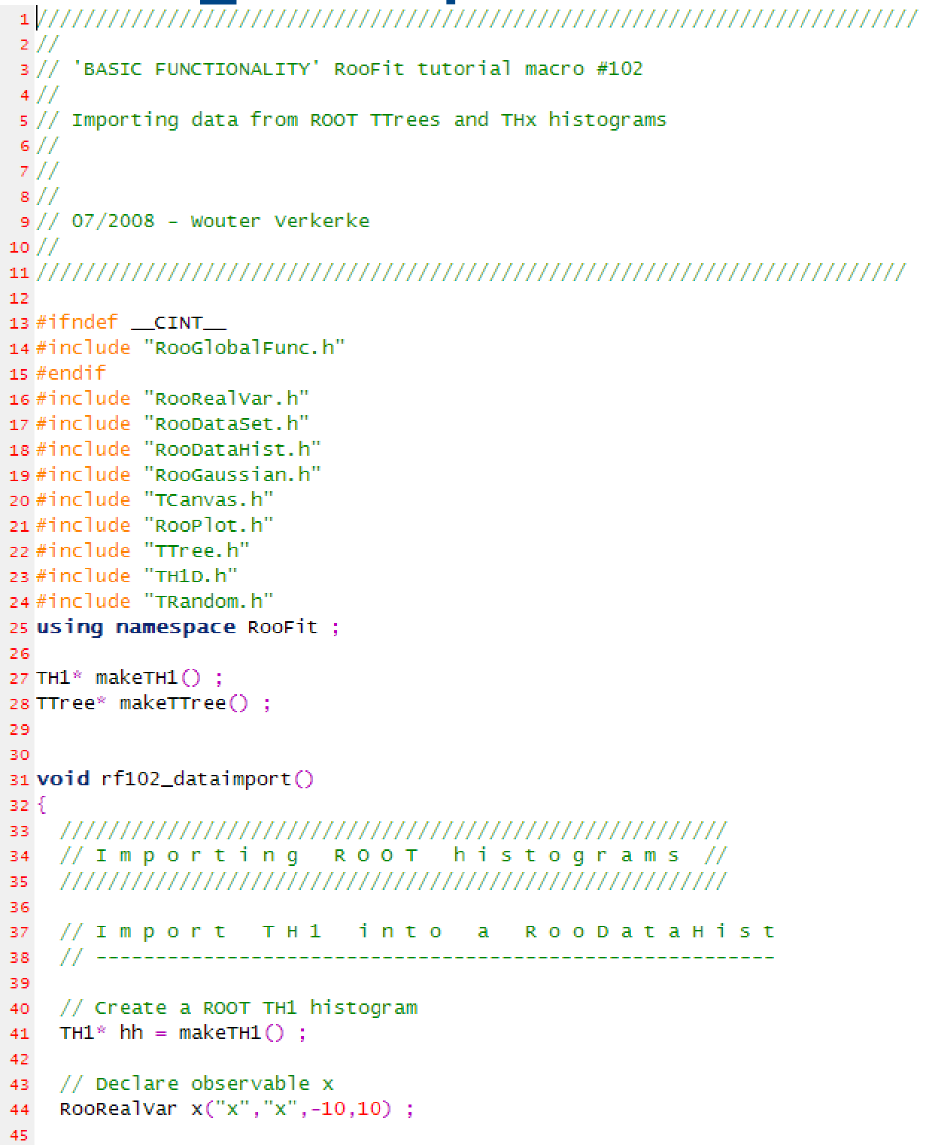


x	y	z
1	3	5
2	4	6

x	y
1	3
2	4

x	y	z
2	4	6

Tutorial macro rf102_dataimport.C



```
//////////////////////////////////////////////////////////////////////////
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #102
//
// Importing data from ROOT TTrees and THx histograms
//
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooDataHist.h"
#include "RooGaussian.h"
#include "TCanvas.h"
#include "RooPlot.h"
#include "TTree.h"
#include "TH1D.h"
#include "TRandom.h"
using namespace RooFit ;

TH1* makeTH1() ;
TTree* makeTTree() ;


void rf102_dataimport()
{
  ////////////////////////////////////////////////////////
  // Importing ROOT histograms //
  ////////////////////////////////////////////////////////

  // Import TH1 into a RooDataHist
  // -------------------------------------------------------

  // Create a ROOT TH1 histogram
  TH1* hh = makeTH1() ;

  // Declare observable x
  RooRealVar x("x","x",-10,10) ;

```

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

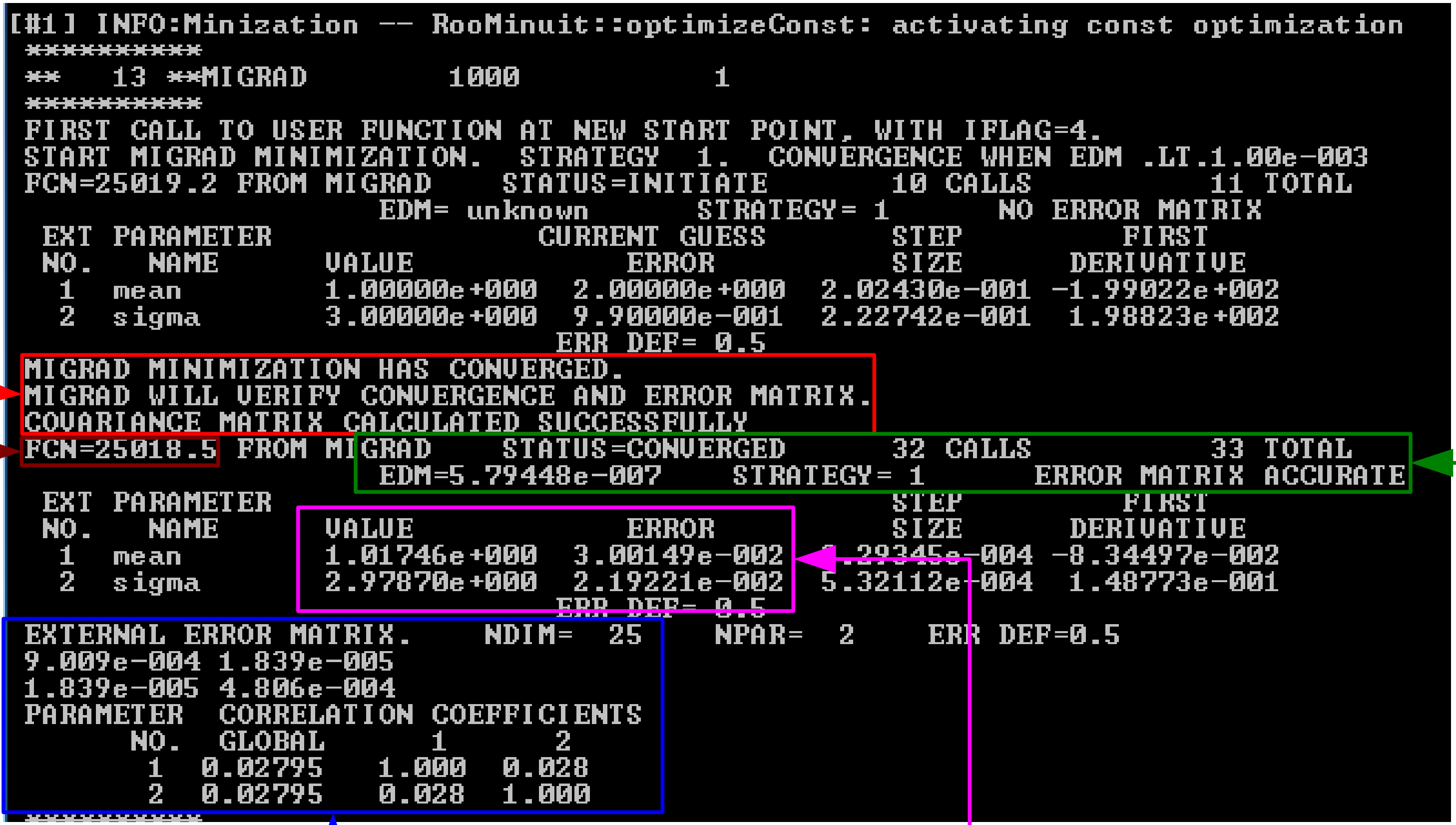
4. Fitting and accessing of results

RooNLLVar can be plotted like any RooRealVar

- 2 different ways of fitting a PDF model to data
  - automatic mode on a given pdf
    `pdf.fitTo(*data)`
  - manual mode:
    ```
    // Construct function object representing –log(L)
    RooNLLVar nll(“nll”,”nll”,pdf,data) ;
    ```

  -
    ```
    // Minimize nll w.r.t its parameters
    RooMinuit m(nll) ;
    m.migrad() ; // find min NLL
    m.hesse() ; // symmetric errors assuming parabola
    m.minos() ; // asymmetric errors from min NLL +0.5
    ```
- both methods accept fit-options (Extended-mode, # of CPU-Cores, fit range, etc)
- fitting is performed via interface with ROOT ***MINUIT*** package
- option “r” saves result in `RooFitResults` object
- further possibilities:
  - profile likelihood with class `RooProfileLL`
  - exporting likelihood function + PDF + data in `Workspace` object

Exemplary fit output



```
[#1] INFO:Minization -- RooMinuit::optimizeConst: activating const optimization
 **********
 **    13 **MIGRAD        1000           1
 **********
 FIRST CALL TO USER FUNCTION AT NEW START POINT, WITH IFLAG=4.
 START MIGRAD MINIMIZATION.  STRATEGY  1.  CONVERGENCE WHEN EDM .LT.1.00e-003
 FCN=25019.2 FROM MIGRAD    STATUS=INITIATE      10 CALLS          11 TOTAL
                     EDM= unknown      STRATEGY= 1      NO ERROR MATRIX
  EXT PARAMETER               CURRENT GUESS       STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean       1.00000e+000  2.00000e+000  2.02430e-001 -1.99022e+002
   2  sigma      3.00000e+000  9.90000e-001  2.22742e-001  1.98823e+002
                               ERR DEF= 0.5
 MIGRAD MINIMIZATION HAS CONVERGED.
 MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=25018.5 FROM MIGRAD    STATUS=CONVERGED     32 CALLS          33 TOTAL
                     EDM=5.79448e-007    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean       1.01746e+000  3.00149e-002  [illegible].29345e-004 -8.34497e-002
   2  sigma      2.97870e+000  2.19221e-002  5.32112e-004  1.48773e-001
                               ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 9.009e-004 1.839e-005
 1.839e-005 4.806e-004
 PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2
        1  0.02795   1.000  0.028
        2  0.02795   0.028  1.000
```

progress information

min NLL

status, distance to minimum (EDM)

fit values and errors

error & correlation matrix

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:

- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

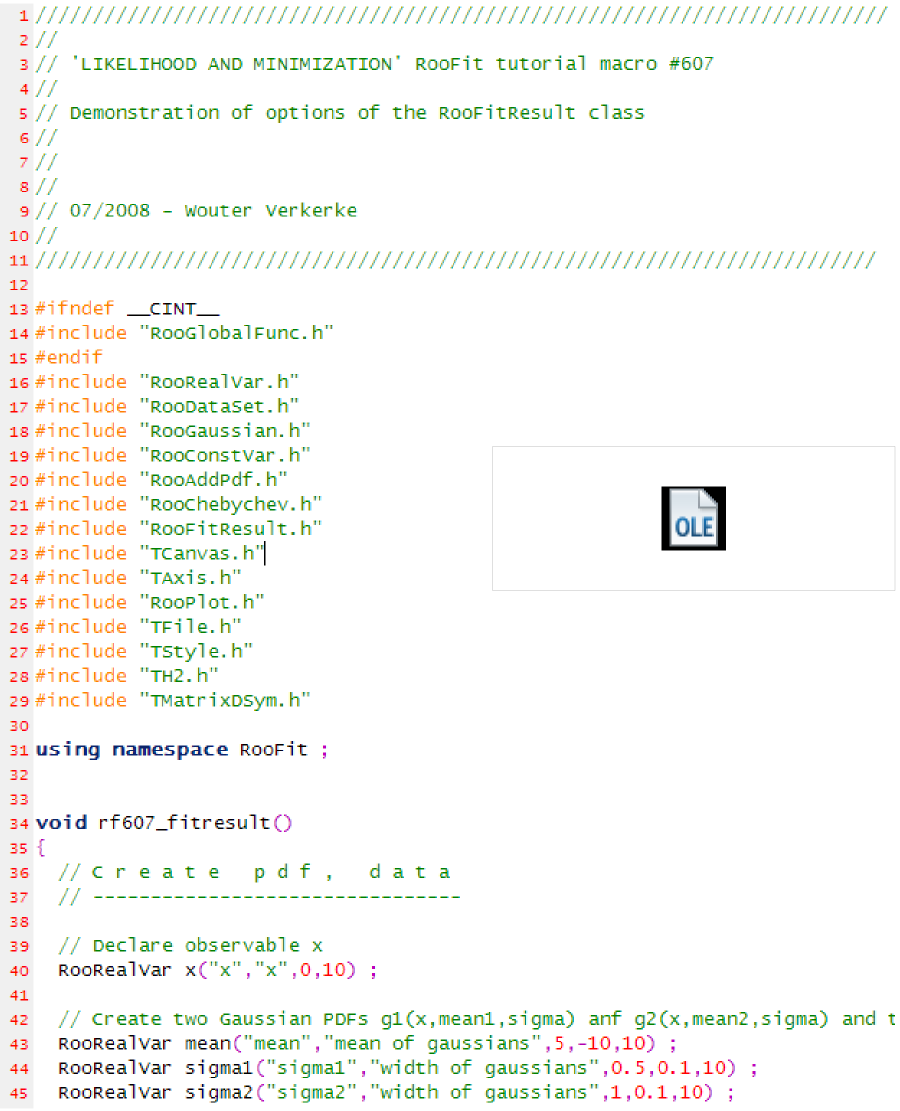
5. plot data and PDF

5. Plotting

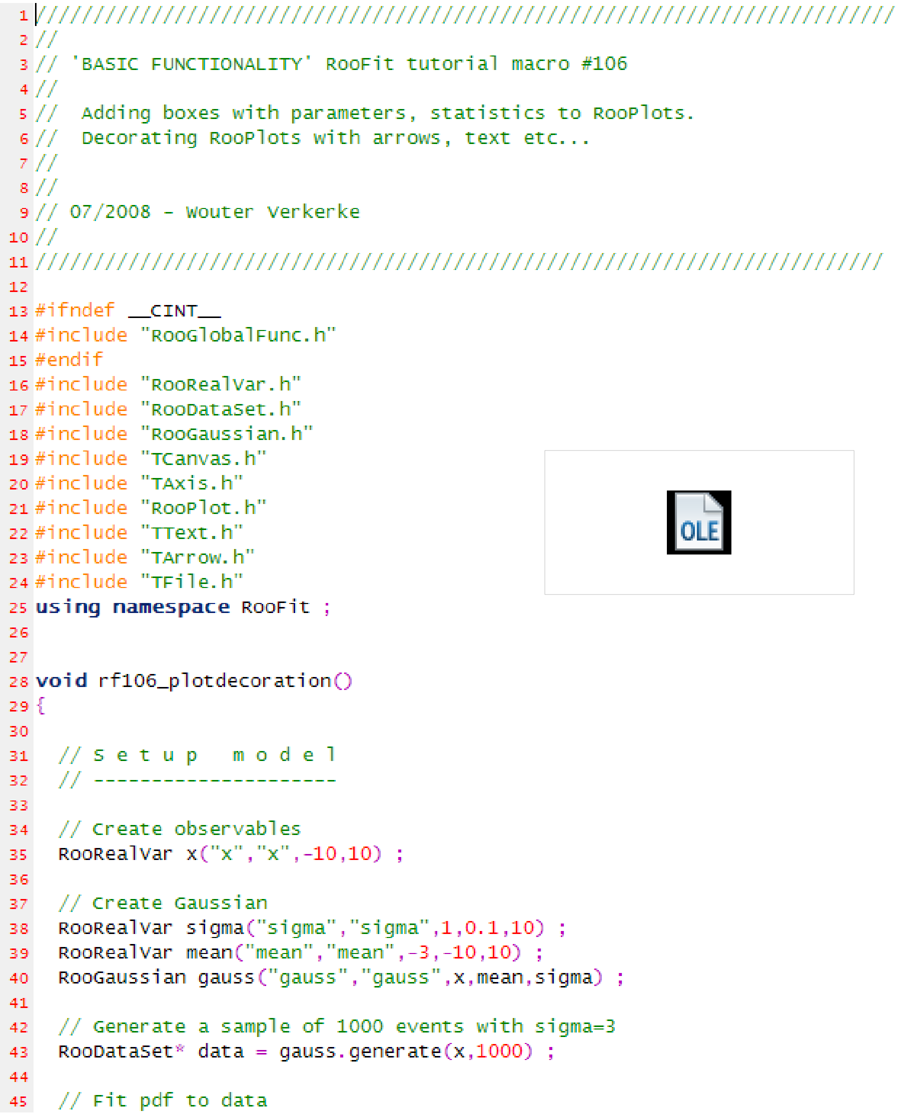
- first: create empty `RooPlot` frame for an observable (i.e. “x”)
- an unbinned dataset is automatically shown as binned histogram when drawn on the frame with `data->plotOn()`
  - customizeable with `Binning(int nbins, double xlo, double xhi)`
  - Markerstyle/color/width etc can of course be changed too
- PDF drawn with `pdf.plotOn()`
  - gets automatically normalized to data set
  - gets automatically projected over all other observables if necessary
- `RooPlot`-frames can hold any other ROOT drawable objects (arrows, text boxes, …): i.e. `xframe.addObject(TArrow)`
- useful information about PDF an data:

  ```
  pdf.paramOn(xframe,data) ;
  data.statOn(xframe) ;
  ```
- further possibilities: plot small slice or larger range of a data set and a PDF
- for >1D PDFs & data: `createHistogram()` method gives a ROOT TH2/TH3

Tutorial macros rf106_plotdecoration.C, rf107_plotstyles.C



```
//////////////////////////////////////////////////////////////////////////
//
// 'LIKELIHOOD AND MINIMIZATION' RooFit tutorial macro #607
//
// Demonstration of options of the RooFitResult class
//
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooConstVar.h"
#include "RooAddPdf.h"
#include "RooChebychev.h"
#include "RooFitResult.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "RooPlot.h"
#include "TFile.h"
#include "TStyle.h"
#include "TH2.h"
#include "TMatrixDSym.h"

using namespace RooFit ;


void rf607_fitresult()
{
  // C r e a t e   p d f ,   d a t a
  // --------------------------------

  // Declare observable x
  RooRealVar x("x","x",0,10) ;

  // Create two Gaussian PDFs g1(x,mean1,sigma) anf g2(x,mean2,sigma) and t
  RooRealVar mean("mean","mean of gaussians",5,-10,10) ;
  RooRealVar sigma1("sigma1","width of gaussians",0.5,0.1,10) ;
  RooRealVar sigma2("sigma2","width of gaussians",1,0.1,10) ;
```



```
//////////////////////////////////////////////////////////////////////////
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #106
//
//  Adding boxes with parameters, statistics to RooPlots.
//  Decorating RooPlots with arrows, text etc...
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "RooPlot.h"
#include "TText.h"
#include "TArrow.h"
#include "TFile.h"
using namespace RooFit ;


void rf106_plotdecoration()
{

  // S e t u p   m o d e l
  // ---------------------

  // Create observables
  RooRealVar x("x","x",-10,10) ;

  // Create Gaussian
  RooRealVar sigma("sigma","sigma",1,0.1,10) ;
  RooRealVar mean("mean","mean",-3,-10,10) ;
  RooGaussian gauss("gauss","gauss",x,mean,sigma) ;

  // Generate a sample of 1000 events with sigma=3
  RooDataSet* data = gauss.generate(x,1000) ;

  // Fit pdf to data
```

Simple example of complete maximum likelihood fit

```
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",
                1,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",
                 1,0.1,10) ;
```

1. define 3 variables:
- *observable* x
- free *parameters* mean, sigma

```
RooGaussian gauss("gauss","gaussian PDF",
                  x,mean,sigma) ;
```

2. create *PDF* model with these variables

```
RooDataSet* data = gauss.generate(x,10000) ;
```

3. generate 10^4 toy events

```
gauss.fitTo(*data) ;
```

4. fit PDF and all floating parameters to data

goodness-of-fit test

```
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

5. plot data and PDF

Testing the *Goodness-of-fit* (1)

How do you know if your fit was good?

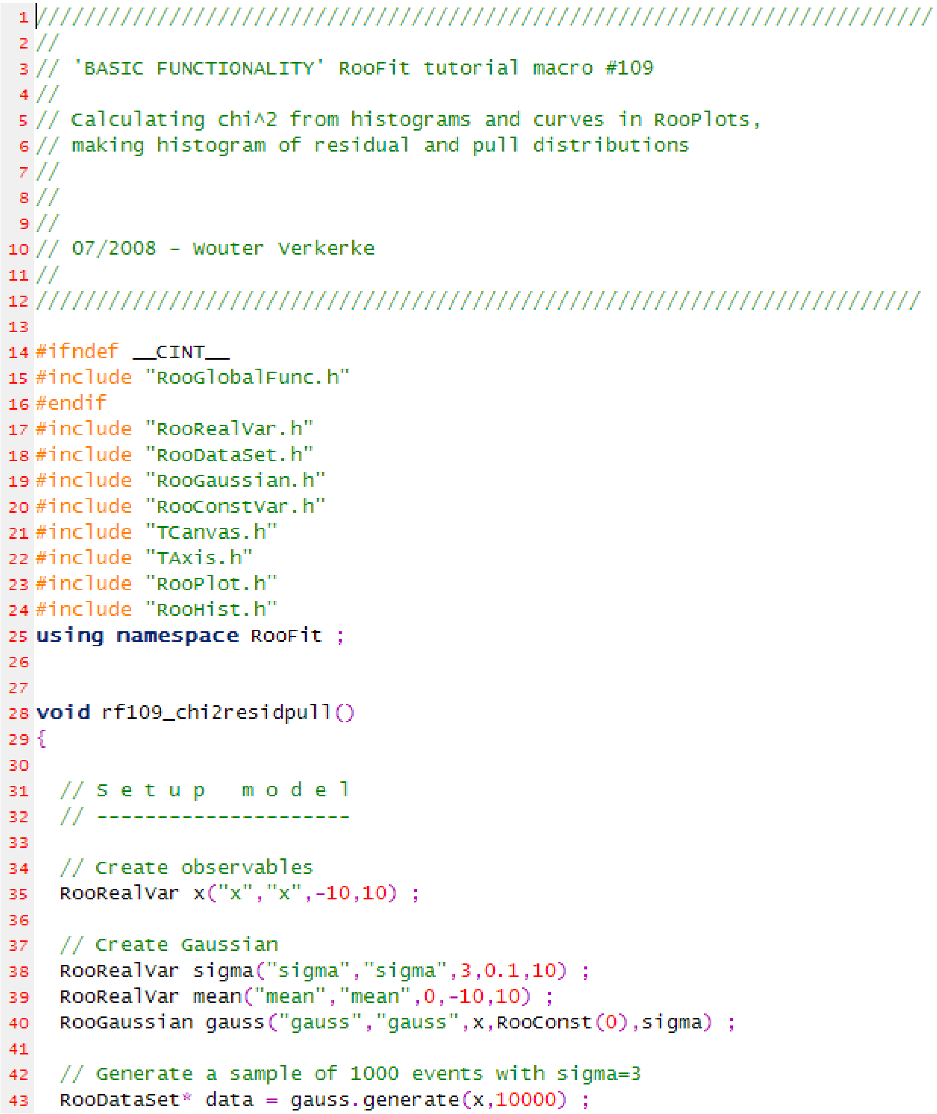
- for 1-D fit:
  - calculate χ^2/d.o.f. of a curve w.r.t. data:

    ```
    frame->chiSquare()
    ```
  - make pull and residual histogram:

    ```
    frame->makePullHist() ;
    frame->makeResidHist() ;
    ```

$$\mathrm{pull}(\mathrm{N}_{\mathrm{sig}}) = \frac{N_{sig}^{fit} - N_{sig}^{true}}{\sigma_{N}^{fit}}$$

Tutorial macro rf109_chi2residpull



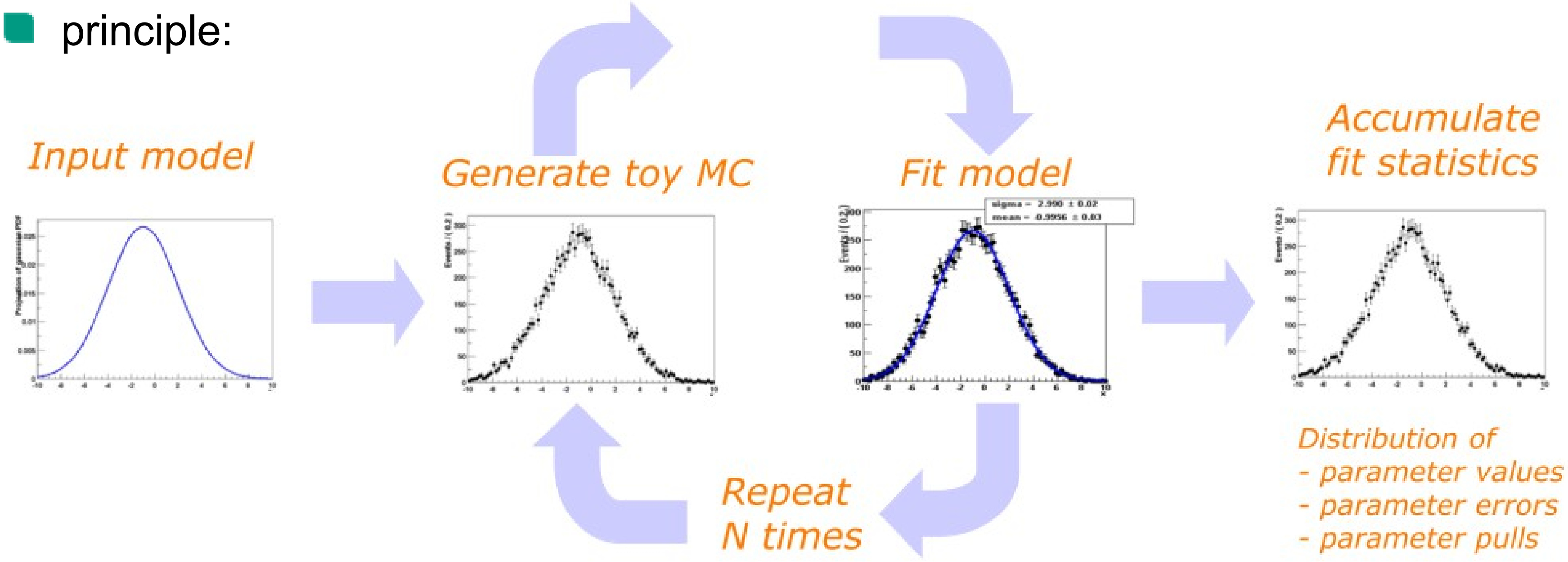
```
//////////////////////////////////////////////////////////////////////////
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #109
//
// Calculating chi^2 from histograms and curves in RooPlots,
// making histogram of residual and pull distributions
//
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooConstVar.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "RooPlot.h"
#include "RooHist.h"
using namespace RooFit ;


void rf109_chi2residpull()
{

  // S e t u p   m o d e l
  // ---------------------

  // Create observables
  RooRealVar x("x","x",-10,10) ;

  // Create Gaussian
  RooRealVar sigma("sigma","sigma",3,0.1,10) ;
  RooRealVar mean("mean","mean",0,-10,10) ;
  RooGaussian gauss("gauss","gauss",x,RooConst(0),sigma) ;

  // Generate a sample of 1000 events with sigma=3
  RooDataSet* data = gauss.generate(x,10000) ;
```

Testing the *Goodness-of-fit* (2)

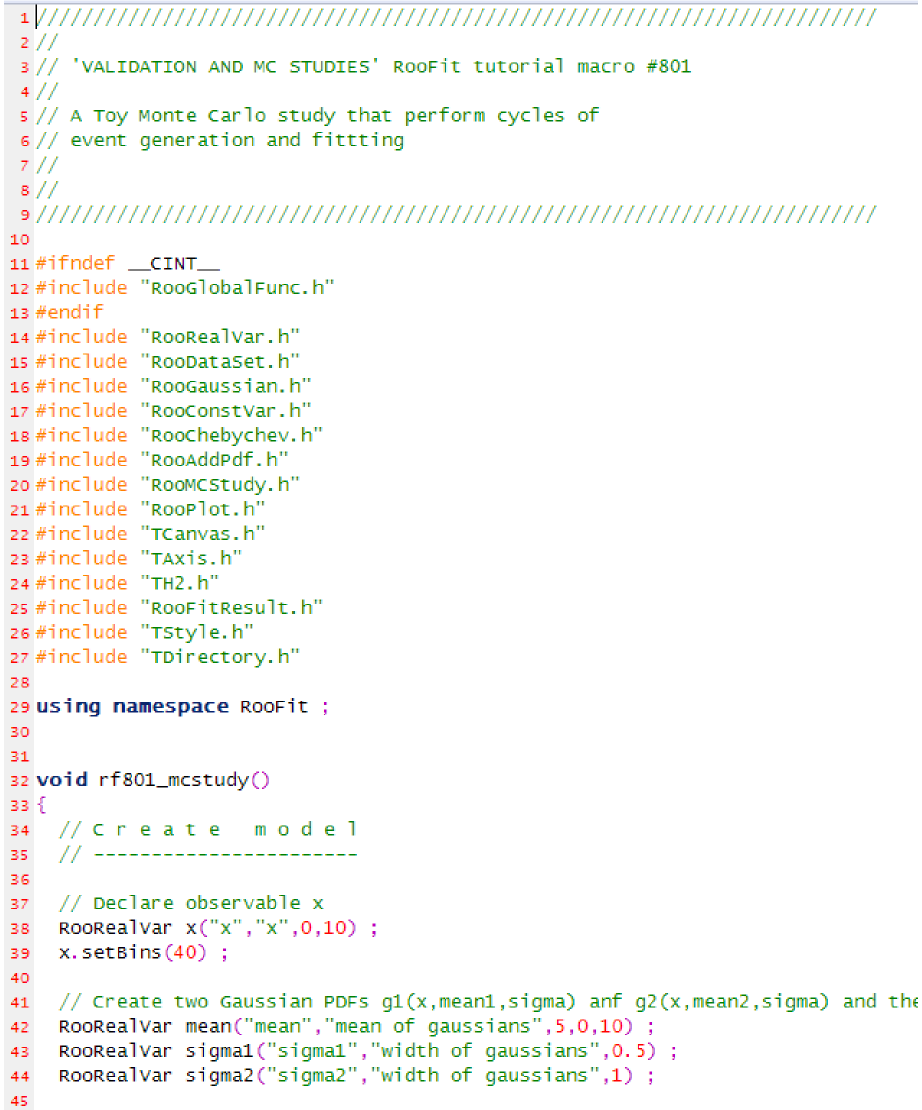
- for > 1-D: *toy Monte Carlo study* using class **RooMCstudy**

```
// 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()
```

- principle:



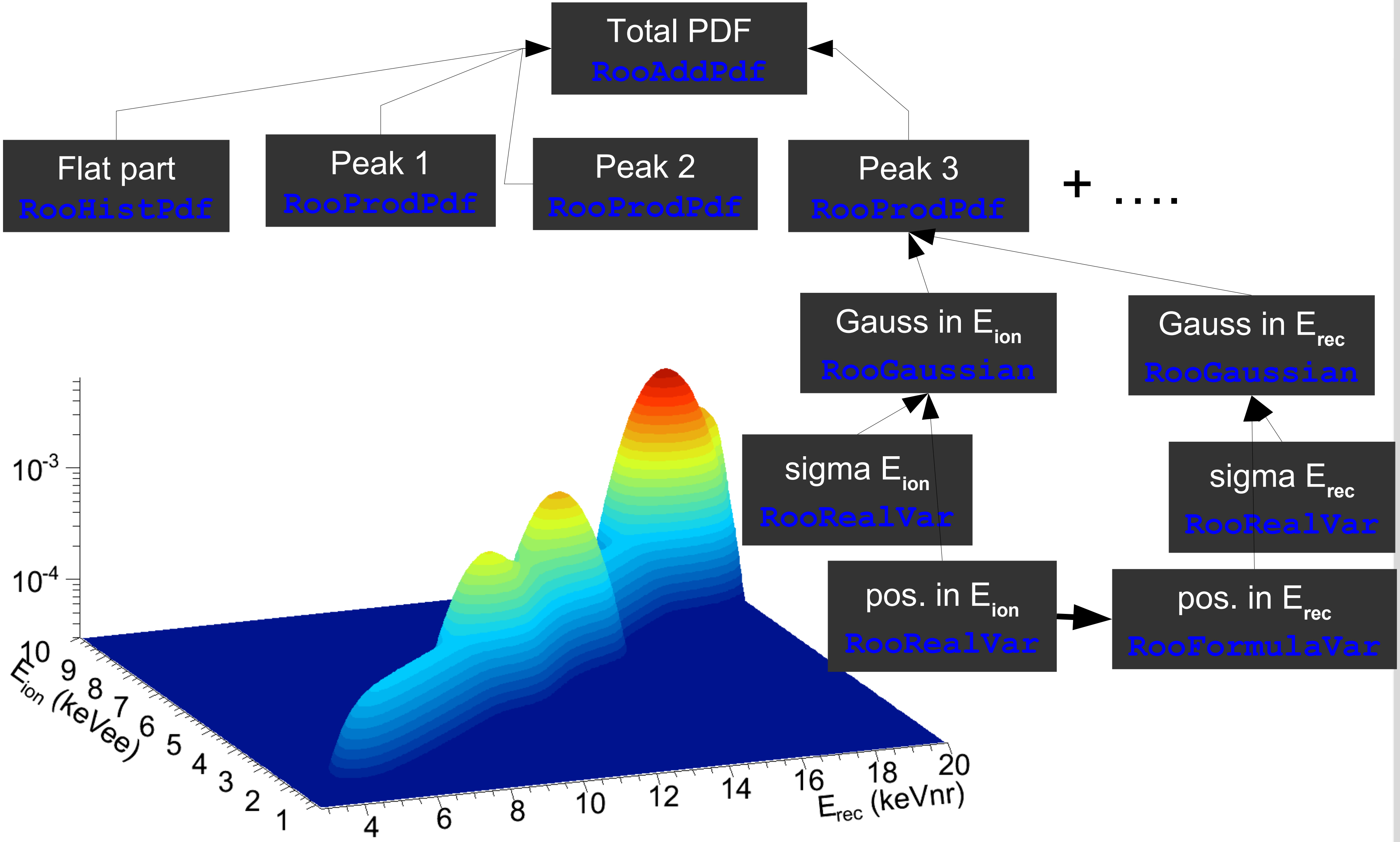
Tutorial macro rf801_mcstudy.C



```
/////////////////////////////////////////////////////////////////////////
//
// 'VALIDATION AND MC STUDIES' RooFit tutorial macro #801
//
// A Toy Monte Carlo study that perform cycles of
// event generation and fittting
//
//
/////////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooConstVar.h"
#include "RooChebychev.h"
#include "RooAddPdf.h"
#include "RooMCStudy.h"
#include "RooPlot.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "TH2.h"
#include "RooFitResult.h"
#include "TStyle.h"
#include "TDirectory.h"

using namespace RooFit ;


void rf801_mcstudy()
{
  // C r e a t e   m o d e l
  // -----------------------

  // Declare observable x
  RooRealVar x("x","x",0,10) ;
  x.setBins(40) ;

  // Create two Gaussian PDFs g1(x,mean1,sigma) anf g2(x,mean2,sigma) and the
  RooRealVar mean("mean","mean of gaussians",5,0,10) ;
  RooRealVar sigma1("sigma1","width of gaussians",0.5) ;
  RooRealVar sigma2("sigma2","width of gaussians",1) ;

```

Some examples of how I use RooFit

PDF for γ-background in EDELWEISS detector



Total PDF
RooAddPdf

Flat part
RooHistPdf

Peak 1
RooProdPdf

Peak 2
RooProdPdf

Peak 3
RooProdPdf

+ ....

Gauss in E_{ion}
RooGaussian

Gauss in E_{rec}
RooGaussian

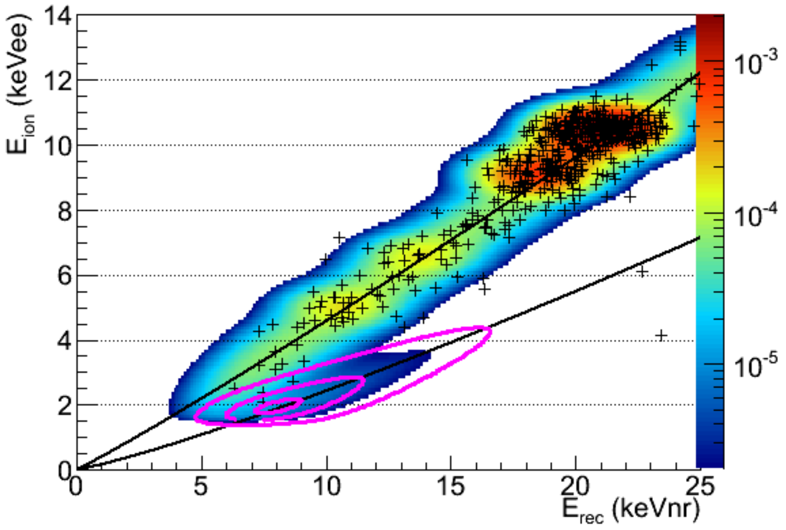
sigma E_{ion}
RooRealVar

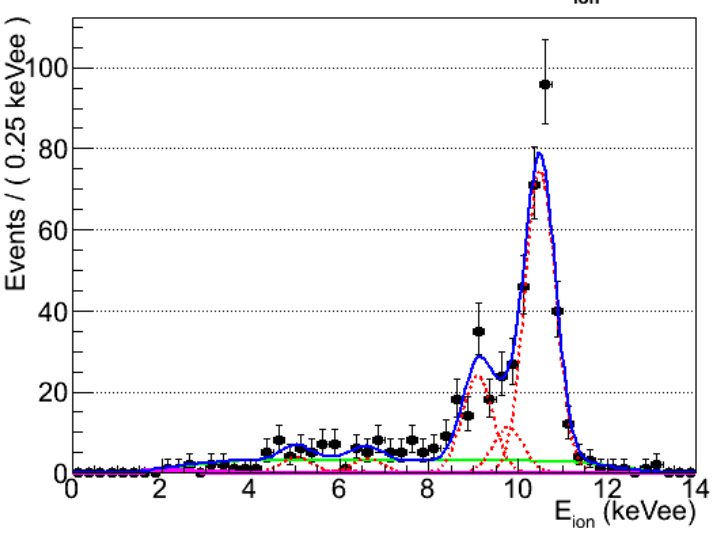
sigma E_{rec}
RooRealVar

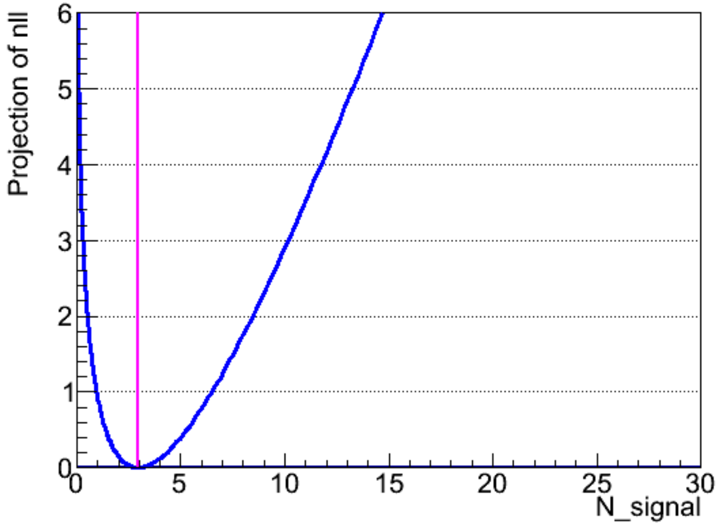
pos. in E_{ion}
RooRealVar

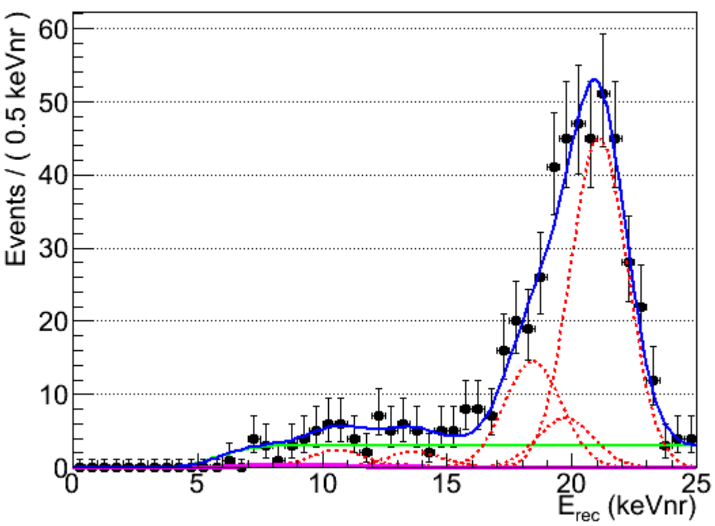
pos. in E_{rec}
RooFormulaVar

Fit of real data to background and signal


ID401 best fit PDF Bckgd + 15GeV Signal


PDF component projection in E_{ion}


Negative Log-Likelihood for N_signal


PDF component projection in E_{rec}

RooFit Summary

- RooFit is a powerful tool for maximum likelihood fits

  … but the documentation could be better :(
- it can be used easily from within ROOT
- there are lots of *different* possibilities to create the PDF describing your data

  … in the worst case by importing a root histogram
- some difficult tmethods are already implemented and very easy to use at first (i.e. toy MC statistics)
- not shown in this *introduction*: short comings and pitfalls of RooFit (how to interpret goodness-of-fit for small signal/noise ratio, convolution in >1D, ...)