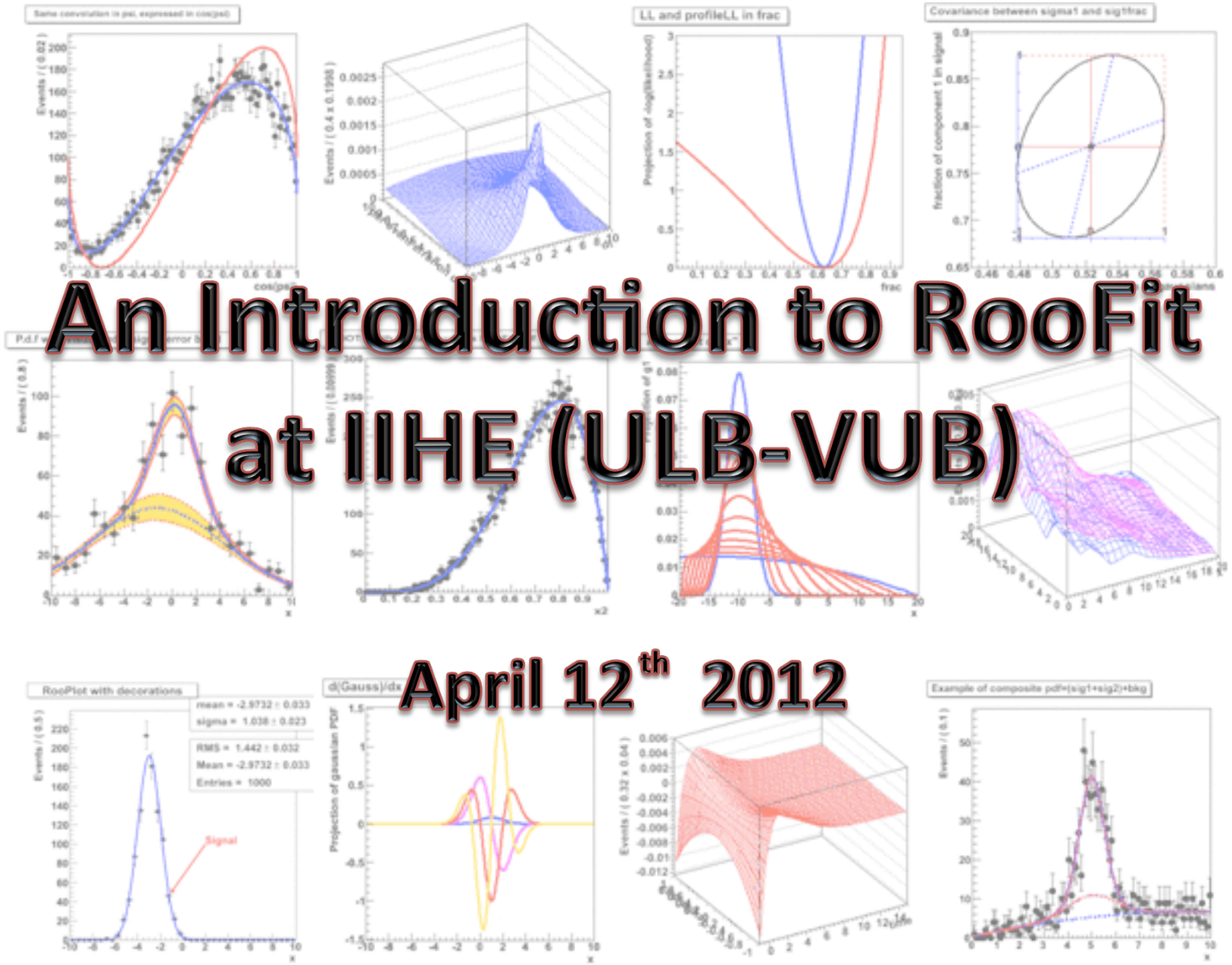


An Introduction to RooFit at IIHE (ULB-VUB)

April 12th 2012

Tutorial Organisation

- The goal of this Tutorial is to introduce the RooFit framework and to perform simple operations with RooFit objects .
- Advanced topics will be taught by Wouter.
- We are not RooFit gurus ☺
- At the end of the session, you should be able to :
  - Define an arbitrary pdf and generate pseudo-experiments according to it.
  - Perform a binned/unbinned likelihood fit of a given dataset.
  - Quantify the goodnees of a fit.

Tutorial Organisation

- The material (**slides**, **root file**) for this Tutorial can be found here: https://lathomas.web.cern.ch/lathomas/Public/

- The **slides** introduce several concepts and propose right after an exercise to apply them.

- The aim of today is to get familiarized with the framework. **Feel free to adapt the examples and to play with them.**

- The last exercise uses a **root file** containing the dielectron invariant mass spectrum of the 2011 CMS data (+ a small extra).

Before Getting Started

- We hope everybody feels comfortable with Root and C++

- We are going to practice just a bit. For that purpose, we will write small codes which we will execute as follows:

  1) Launch your VirtualBox and start your Fedora 16 OS
  2) Find out your IP address by executing the command `ifconfig`
  3) Connect via `ssh` to your virtual machine from your computer
  4) Write the code (small macro directly executable by Root)
  5) Execute the code:

     `course@192.168.56.X: $ root -l`

     `root [0] .x myMacro.C`

     Or even simpler if you give the same name to your file and your function:

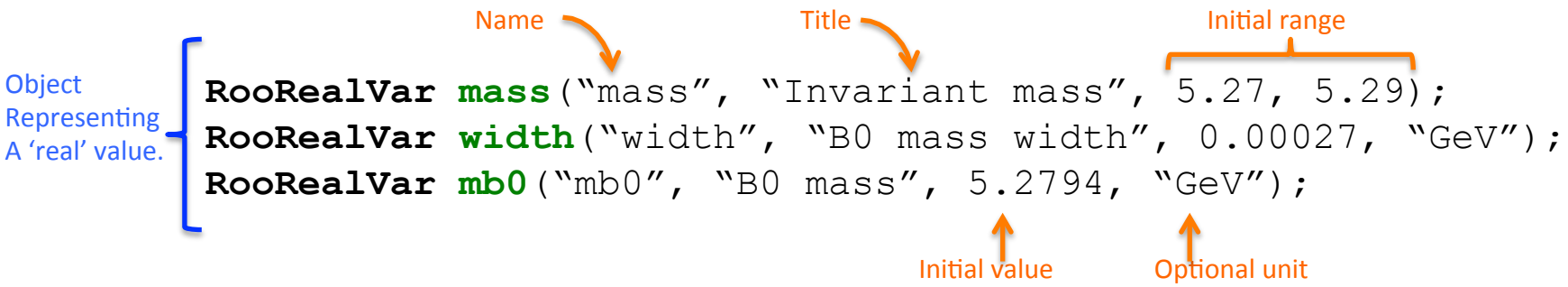
     `course@192.168.56.X: $ root -l myMacro.C`

     Quit Root if you've finished

     `root[i] .q` (**`.qqqqqqqqqq`** is sometimes needed)

RooFit Objects and PDFs

- In RooFit, variables, data points, functions, PDF, are represented in a C++ object.
- To represent a real value object one uses the **`RooRealVar`** class:



- Many objects are already implemented in RooFit:

PDF object

```
RooGaussian b0sig("b0sig", "B0 sig PDF", mass, mb0, width);
```

- For each object, many methods are accessible. Class documentation available at:
  http://root.cern.ch/root/html/ROOFIT_Index.html

RooFit Objects and PDFs

- Print() method, works for all RooFit objects:

Print value and attributes

```
mass.Print();
RooRealVar::mass:  5.28  L(5.27 - 5.29)
```

Assign new value

```
mass = 5.285;
mass.setVal(5.285);
```

Retrieve contents

```
Double_t massVal = mass.getVal();
```

```
b0sig.Print();
RooGaussian::b0sig(mass,mb0,width) = 0
```

- How to plot in RooFit ? -> Use the **RooPlot** class:

Create the frame for the mass variable

```
RooPlot *massFrame = mass.frame();
```

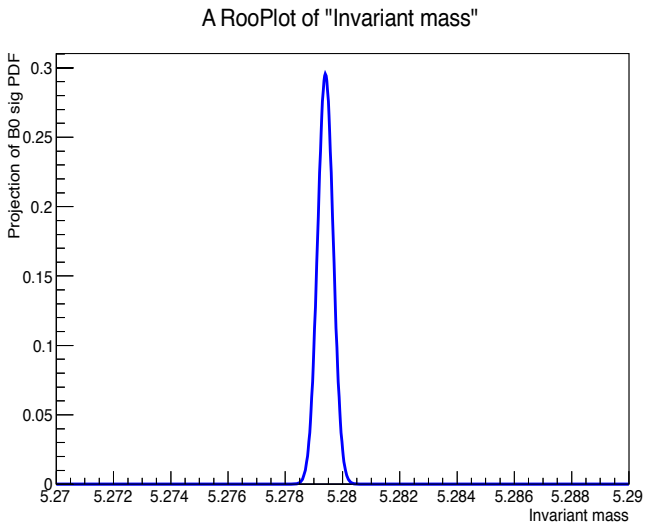
Plot the PDF on that frame

```
b0sig.plotOn(massFrame);
```

Draw your frame

```
massFrame->Draw();
```

A **RooPlot** is an empty frame capable of holding anything plotted versus its variable



RooFit Objects and PDFs

- Some other classes:

Set for which:
- Each element may appear only once
- No ordering of elements

```
RooArgSet s1(x,y,z);
RooArgSet s2(x,x,z); //ERROR !
```

List for which:
- Elements may be inserted multiple times
- Insertion order is preserved

```
RooArgList l1(x,y,z);
RooArgList l2(x,x,y);
l2.Print();
RooArgList:::
    1) RooRealVar::x: "x"
    2) RooRealVar::x: "x"
    3) RooRealVar::y: "y"
```

- Frequently used to pass or return arguments

```
RooArgSet mySet(mass, mb0, width);
RooGaussian myGauss("myGauss", "b0 signal",mySet);
```

RooFit Objects and PDFs

- More PDFs:

RooArgusBG	- Argus background shape
RooBCPEffDecay	- B0 decay with CP violation
RooBMixDecay	- B0 decay with mixing
RooBifurGauss	- Bifurcated Gaussian
RooBreitWigner	- Breit-Wigner shape
RooCBShape	- Crystal Ball function
RooChebychev	- Chebychev polynomial
RooDecay	- Simple decay function
RooDircPdf	- DIRC resolution description
RooDstD0BG	- D* background description
RooExponential	- Exponential function
RooGaussian	- Gaussian function
RooKeysPdf	- Non-parametric data description
Roo2DKeysPdf	- Non-parametric data description
RooPolynomial	- Generic polynomial PDF
RooVoigtian	- Breit-Wigner (X) Gaussian

- More information at:
  http://root.cern.ch/root/html/ROOFIT_ROOFIT_Index.html

RooFit Objects and PDFs

- Even more PDFs thanks to **RooGenericPDF**:

```
// PDF variables
RooRealVar x("x","x",-10,10) ;
RooRealVar y("y","y",0,5) ;
RooRealVar a("a","a",3.0) ;
RooRealVar b("b","b",-2.0) ;

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

- Automatic normalization:
  - Expression divided by numerical integral of expression

RooFit Objects and PDFs

- **Exercise 1:**

  Write a code which draws a Breit-Wigner PDF with mean = 91.2 GeV and width = 2.5 GeV.
  On the same plot, in red color draw a PDF which simulates the effect of the detector's resolution of about 1 GeV (check the the **`RooVoigtian`** doc.)

Combining PDFs and Fitting

- Complex PDFs can be easily composed using operator classes. It is then possible to add, multiply, compose, convolve different PDFs.

1) Adding PDFs is done by using the **RooAddPdf** class which constructs the sum of N PDFs with N-1 coefficients:

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

Build 2 Gaussian PDFs

```
// Build two Gaussian PDFs
RooRealVar x("x","x",0,10) ;
RooRealVar mean1("mean1","mean of gaussian 1",2) ;
RooRealVar mean2("mean2","mean of gaussian 2",3) ;
RooRealVar sigma("sigma","width of gaussians",1) ;
RooGaussian gauss1("gauss1","gaussian PDF",x,mean1,sigma) ;
RooGaussian gauss2("gauss2","gaussian PDF",x,mean2,sigma) ;
```

Build ArgusBG PDF

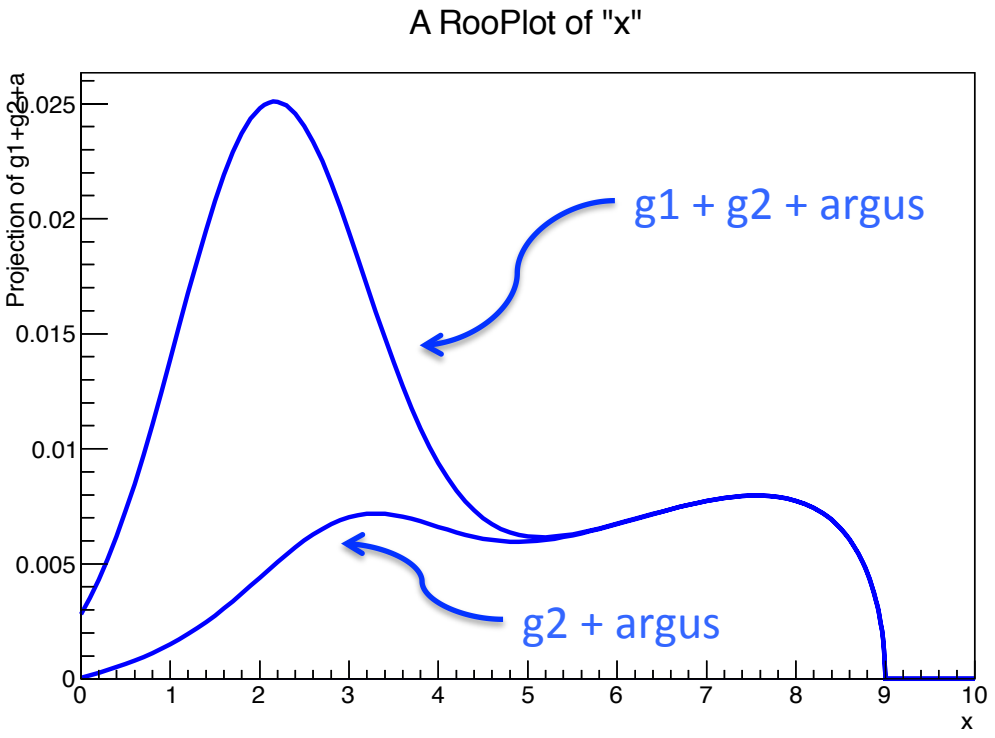
```
// Build Argus background PDF
RooRealVar argpar("argpar","argus shape parameter",-1.0) ;
RooRealVar cutoff("cutoff","argus cutoff",9.0) ;
RooArgusBG argus("argus","Argus PDF",x,cutoff,argpar) ;
```

Build the sum of the 3 PDFs

```
// Add the components
RooRealVar g1frac("g1frac","fraction of gauss1",0.5) ;
RooRealVar g2frac("g2frac","fraction of gauss2",0.1) ;
RooAddPdf sum("sum","g1+g2+a",RooArgList(gauss1,gauss2,argus),
                              RooArgList(g1frac,g2frac)) ;
```

Combining PDFs and Fitting

```
// Plot data and PDF overlaid
RooPlot* xframe = x.frame();
sum->plotOn(xframe);
// Plot only argus and gauss2
sum->plotOn(xframe,Components(RooArgSet(argus,gauss2)));
xframe->Draw();
```



Combining PDFs and Fitting

- Complex PDFs can be easily composed using operator classes. It is then possible to add, multiply, compose, convolve different PDFs.

  2) Multiplying PDFs is done by using the **`RooProdPdf`** class which constructs the joint PDF of N PDFs:

$$P = P_0(x_1, x_2, \dots) \cdot P_1(y_1, y_2, \dots) \cdots P_n(w_1, w_2, \dots)$$

```
// Build two Gaussian PDFs
RooRealVar x("x","x",-5,5) ;
RooRealVar y("y","y",-5,5) ;
RooGaussian gaussx("gaussx","gaussx",x,meanx,sigmax);
RooGaussian gaussy("gaussy","gaussx",y,meany,sigmay);

// Multiply the components
RooProdPdf prod("gaussxy","gaussx*gaussy",RooArgList(gaussx,gaussy));
```

- For the others (compose, convolution, …) see the bibliography

Combining PDFs and Fitting

- With RooFit, generating MC events according to some PDF is a matter of one line !
- With RooFit, fitting is a matter of two lines !
  No, just kidding, only one is enough ☺

Let's see how that works !

```
// Build two Gaussian PDFs
RooRealVar x("x","x",0,10) ;
RooRealVar mean("mean","mean of gaussian ",2) ;
RooRealVar sigma("sigma","width of gaussian",1) ;
RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma) ;
```

Simulation of a gaussian

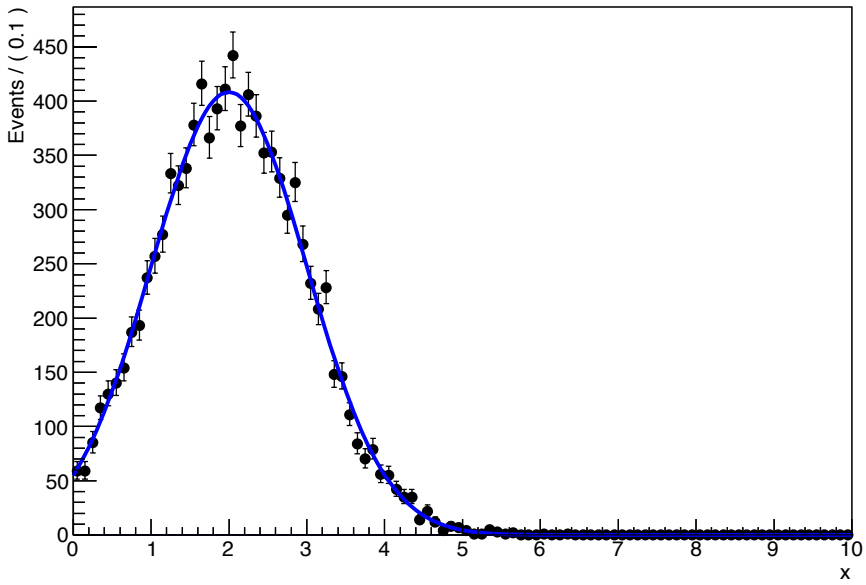
```
RooDataSet *data = gauss.generate(x, 10000);
```

Unbinned likelihood fit

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

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

A RooPlot of "x"



Combining PDFs and Fitting

- Getting the fit results is achieved by what follows:

We use a **RooFitResult** object which holds complete snapshot of fit results.

```
RooFitResult *res = gauss.fitTo(*data, Save());
res->Print();
Double_t myMean =
    ((RooRealVar*)res->floatParsFinal().find("mean"))->getVal();

Double_t meanErrHi =
    ((RooRealVar*)res->floatParsFinal().find("mean"))->getErrorHi();

Double_t corr = res->correlation(mean, sigma);

Double_t minNll = res->minNll();
```

Many other methods exist for **RooFitResult**… You know how to find it !!!

Combining PDFs and Fitting

- Getting the goodness-of-fit for an unbinned likelihood fit:

1) Binning the data

One way to get an estimation of the fit quality is obtained by computing the chi2 on the binned data.

Binning

```
RooPlot *xframe = x.frame(50);
data->plotOn(xframe);
gauss.plotOn(xframe);
xframe->Draw();
Double_t chi2 = xframe->chiSquare();
Double_t ndof = xframe->GetNbinsX();
ndof -= nParams;
```

χ^2/ndof

2) Toy MC study

Compare the observed Likelihood to the Likelihood distribution of pseudo-experiments generated according to the fit function.

```
RooMCStudy mgr(gauss, gauss, x);
mgr.generateAndFit(1000,100);
RooPlot* frame = mgr.plotNLL(Bins(50));
```

Combining PDFs and Fitting

- **Exercise 2:**

  Build a PDF, between 0. and 300. which is the sum of 95% of a decreasing exponential plus 5% of a Breit-Wigner. For the exponential take a slope of -0.01/GeV and for the Breit- Wigner, take a mean of 150 Gev and a width of 12 GeV.

  Generate a dataset of 1000 events and fit it with the PDF you built.
  Check the χ^2 of the fit, and save the value of min(–log(L))

  Generate 1000 pseudo-experiments of 1000 events each and plot the distribution of the min(–log(L)).

  Compare your min(–log(L)) to the distribution you've obtained.

Dealing With Real Data

- Import data from a Tree:

```
RooRealVar myVar("myVar", "Tree Variable", 0, 1000);
TTree* tree = <someTFile>.Get("<someTTree>");
RooDataSet data("data", "data", tree, myVar);
```

Where the tree is assumed to have a branch named "**myVar**"

- Import binned data:

```
RooRealVar x("x","x", 0, 100);
RooRealVar y("y","y", 0, 100);

TH2* histo = <someTH2>;
RooDataHist bdata("bdata","bdata",RooArgList(x,y), histo);

RooDataSet* data = <someUnbinnedData>;
RooDataHist bdata("bdata","bdata",RooArgList(x,y), data);
```

Dealing With Real Data

- **Exercise 3:**

  Write a code which fits (with an unbinned maximum likelihood fit) the invariant mass distribution contained in the tree from the Zmass.root file. Use the following formula for the background:

$$x^n \cdot \exp(b \cdot x)$$

  Ranges for n: (-10, 0); b: (-0.5, 0).

  Check the χ^2 of the fit.
  Find the Z' by fitting a PDF which is the sum of the background and a Voigtian.
  Check the χ^2 of the fit.

  Repeat the exercise with a binned likelihood fit on the histogram contained in the file Zmass.root.