RooFit

Siguang WANG

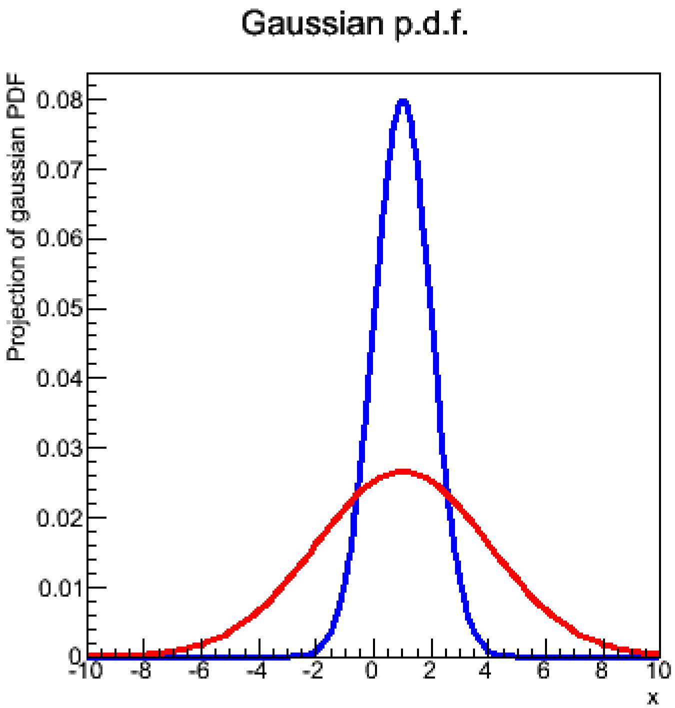
siguang@pku.edu.cn

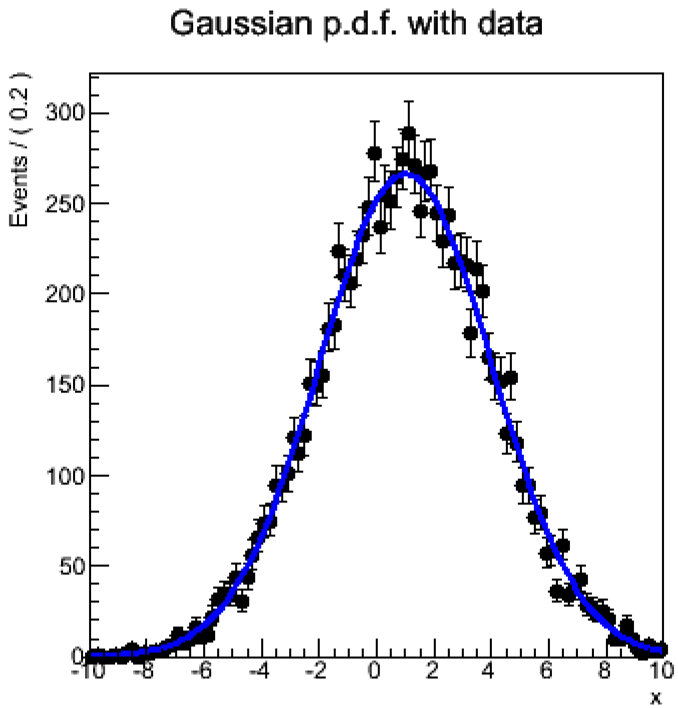
$ROOTSYS/tutorials/roofit

1. rf101_basics.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #101
2. rf102_dataimport.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #102
3. rf103_interprfuncs.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #103
4. rf104_classfactory.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #104
5. rf105_funcbinding.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #105
6. rf106_plotdecoration.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #106
7. rf107_plotstyles.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #107
8. rf108_plotbinning.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #108
9. rf109_chi2residpull.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #109
10. rf110_normintegration.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #110
11. rf111_derivatives.C	'BASIC FUNCTIONALITY' RooFit tutorial macro #111
12. rf201_composite.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #201
13. rf202_extendedmlfit.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #202
14. rf203_ranges.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #203
15. rf204_extrangefit.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #204
16. rf205_compplot.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #205
17. rf206_treevistools.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #206
18. rf207_comptools.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #207
19. rf208_convolution.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #208
20. rf209_anaconv.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #209
21. rf210_angularconv.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #210
22. rf211_paramconv.C	'ADDITION AND CONVOLUTION' RooFit tutorial macro #211

……

rf101_basics.C



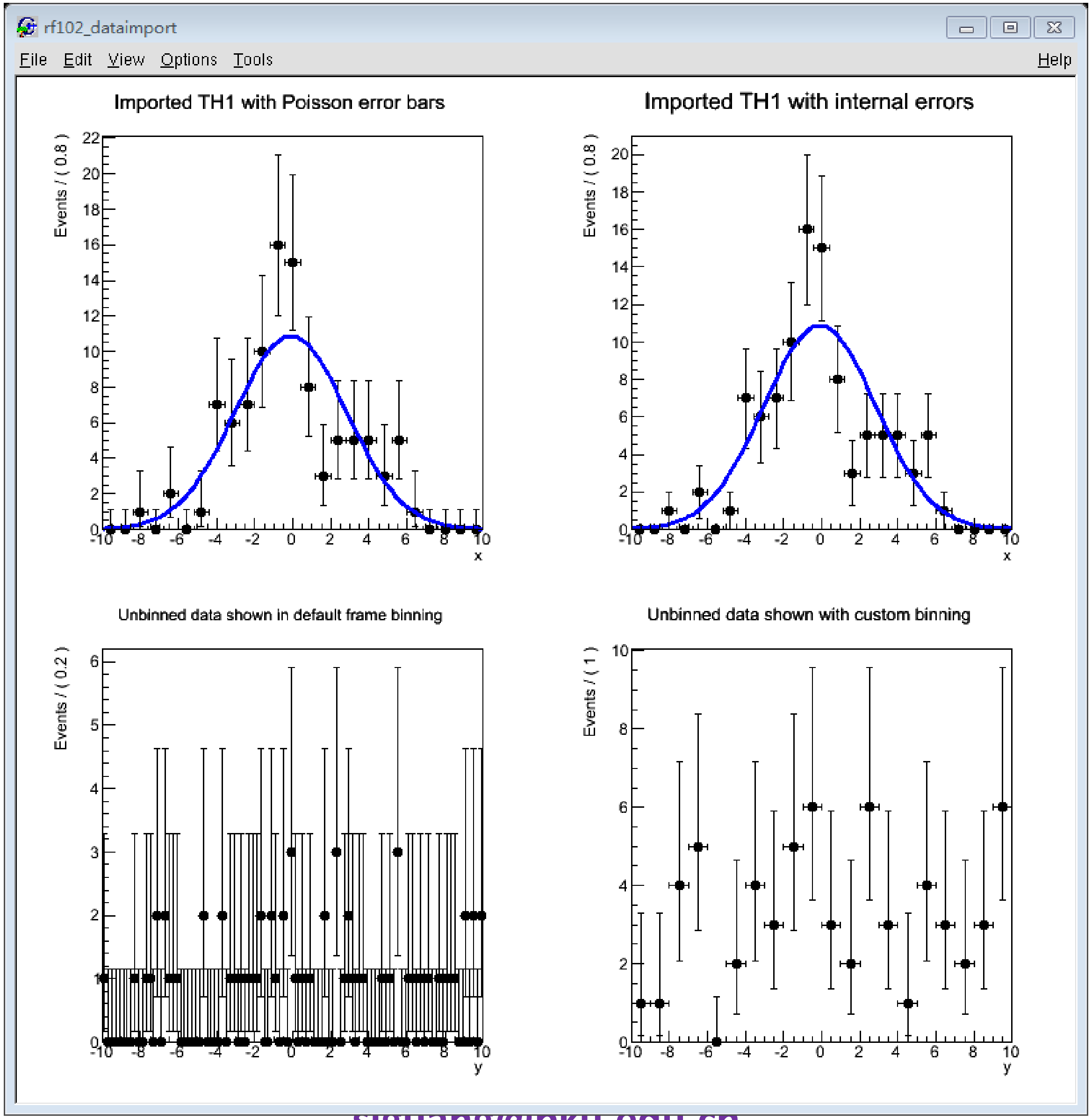


```
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #101
//
// Fitting, plotting, toy data generation on one-dimensional p.d.f
//
// pdf = gauss(x,m,s)
//
//
// 07/2008 - Wouter Verkerke
//
/////////////////////////////////////////////////////////////////////////

#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.")) ;


// P l o t   m o d e l   a n d   c h a n g e   p a r a m e t e r   v a l u e s
// ---------------------------------------------------------------------------

// Plot gauss in frame (i.e. in x)
gauss.plotOn(xframe) ;

// Change the value of sigma to 3
sigma.setVal(3) ;

// Plot gauss in frame (i.e. in x) and draw frame on canvas
gauss.plotOn(xframe,LineColor(kRed)) ;


// G e n e r a t e   e v e n t s
// -----------------------------

// Generate a dataset of 1000 events in x from gauss
RooDataSet* data = gauss.generate(x,10000) ;

// Make a second plot frame in x and draw both the
// data and the p.d.f in the frame
RooPlot* xframe2 = x.frame(Title("Gaussian p.d.f. with data")) ;
data->plotOn(xframe2) ;
```

```
// F i t   m o d e l   t o   d a t a
// -----------------------------

// Fit pdf to data
gauss.fitTo(*data) ;

// Print values of mean and sigma (that now reflect fitted values and errors)
mean.Print() ;
sigma.Print() ;

// Draw all frames on a canvas
TCanvas* c = new TCanvas("rf101_basics","rf101_basics",800,400) ;
c->Divide(2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; xframe->GetYaxis()->SetTitleOffset(1.6) ; xframe->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; xframe2->GetYaxis()->SetTitleOffset(1.6) ; xframe2->Draw() ;


} ;
```

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() ;
```

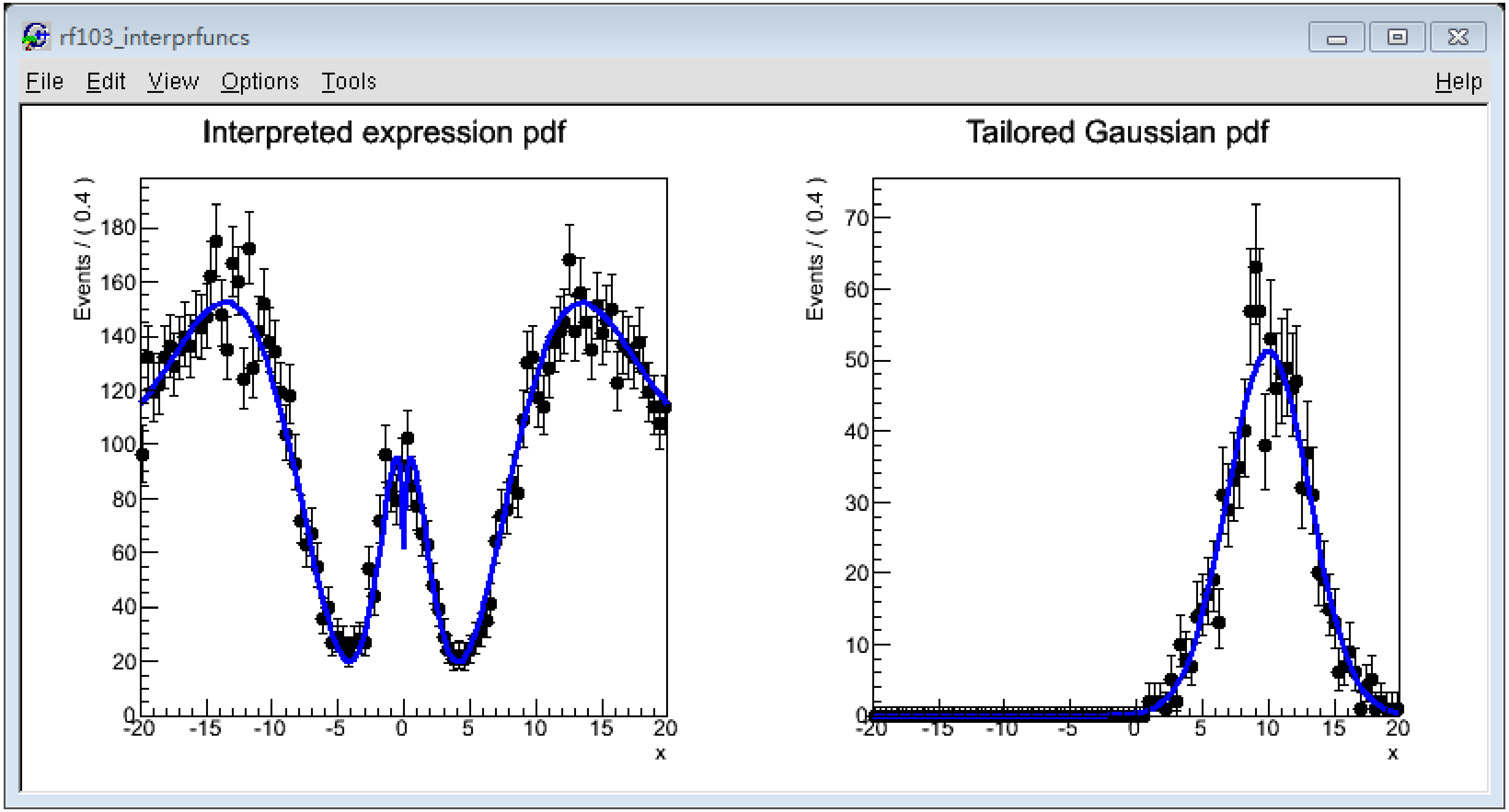
```
void rf102_dataimport()
{
  ////////////////////////////////////////////////////////
  // I m p o r t i n g   R O O T   h i s t o g r a m s //
  ////////////////////////////////////////////////////////

  // I m p o r t   T H 1   i n t o   a   R o o D a t a H i s t
  // ---------------------------------------------------------

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

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

  // Create a binned dataset that imports contents of TH1 and associates its contents to observable 'x'
  RooDataHist dh("dh","dh",x,Import(*hh)) ;

  // P l o t   a n d   f i t   a   R o o D a t a H i s t
  // ---------------------------------------------------

  // Make plot of binned dataset showing Poisson error bars (RooFit default)
  RooPlot* frame = x.frame(Title("Imported TH1 with Poisson error bars")) ;
  dh.plotOn(frame) ;

  // Fit a Gaussian p.d.f to the data
  RooRealVar mean("mean","mean",0,-10,10) ;
  RooRealVar sigma("sigma","sigma",3,0.1,10) ;
  RooGaussian gauss("gauss","gauss",x,mean,sigma) ;
  gauss.fitTo(dh) ;
  gauss.plotOn(frame) ;
```

```
// P l o t   a n d   f i t   a   R o o D a t a H i s t   w i t h   i n t e r n a l   e r r o r s
// ---------------------------------------------------------------------------------------------

// If histogram has custom error (i.e. its contents is does not originate from a Poisson process
// but e.g. is a sum of weighted events) you can data with symmetric 'sum-of-weights' error instead
// (same error bars as shown by ROOT)
RooPlot* frame2 = x.frame(Title("Imported TH1 with internal errors")) ;
dh.plotOn(frame2,DataError(RooAbsData::SumW2)) ;
gauss.plotOn(frame2) ;

// Please note that error bars shown (Poisson or SumW2) are for visualization only, the are NOT used
// in a maximum likelihood fit
//
// A (binned) ML fit will ALWAYS assume the Poisson error interpretation of data (the mathematical definition

// of likelihood does not take any external definition of errors). Data with non-unit weights can only be corr
ectly
// fitted with a chi^2 fit (see rf602_chi2fit.C)
```

```
// I m p o r t   T T r e e   i n t o   a   R o o D a t a S e t
// -----------------------------------------------------------

TTree* tree = makeTTree() ;

// Define 2nd observable y
RooRealVar y("y","y",-10,10) ;

// Construct unbinned dataset importing tree branches x and y matching between branches and RooRealVars
// is done by name of the branch/RRV
//
// Note that ONLY entries for which x,y have values within their allowed ranges as defined in
// RooRealVar x and y are imported. Since the y values in the import tree are in the range [-15,15]
// and RRV y defines a range [-10,10] this means that the RooDataSet below will have less entries than the TTr
ee 'tree'

RooDataSet ds("ds","ds",RooArgSet(x,y),Import(*tree)) ;


// P l o t   d a t a s e t   w i t h   m u l t i p l e   b i n n i n g   c h o i c e s
// ------------------------------------------------------------------------------------

// Print number of events in dataset
ds.Print() ;

// Print unbinned dataset with default frame binning (100 bins)
RooPlot* frame3 = y.frame(Title("Unbinned data shown in default frame binning")) ;
ds.plotOn(frame3) ;

// Print unbinned dataset with custom binning choice (20 bins)
RooPlot* frame4 = y.frame(Title("Unbinned data shown with custom binning")) ;
ds.plotOn(frame4,Binning(20)) ;

// Draw all frames on a canvas
TCanvas* c = new TCanvas("rf102_dataimport","rf102_dataimport",800,800) ;
c->Divide(2,2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4) ; frame->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.4) ; frame2->Draw() ;
c->cd(3) ; gPad->SetLeftMargin(0.15) ; frame3->GetYaxis()->SetTitleOffset(1.4) ; frame3->Draw() ;
c->cd(4) ; gPad->SetLeftMargin(0.15) ; frame4->GetYaxis()->SetTitleOffset(1.4) ; frame4->Draw() ;

}
```

```
TH1* makeTH1()
{
  // Create ROOT TH1 filled with a Gaussian distribution

  TH1D* hh = new TH1D("hh","hh",25,-10,10) ;
  for (int i=0 ; i<100 ; i++) {
    hh->Fill(gRandom->Gaus(0,3)) ;
  }
  return hh ;
}


TTree* makeTTree()
{
  // Create ROOT TTree filled with a Gaussian distribution in x and a uniform distribution in y

  TTree* tree = new TTree("tree","tree") ;
  Double_t* px = new Double_t ;
  Double_t* py = new Double_t ;
  tree->Branch("x",px,"x/D") ;
  tree->Branch("y",py,"y/D") ;
  for (int i=0 ; i<100 ; i++) {
    *px = gRandom->Gaus(0,3) ;
    *py = gRandom->Uniform()*30 - 15 ;
    tree->Fill() ;
  }
  return tree ;
}
```

rf103_interprfuncs.C

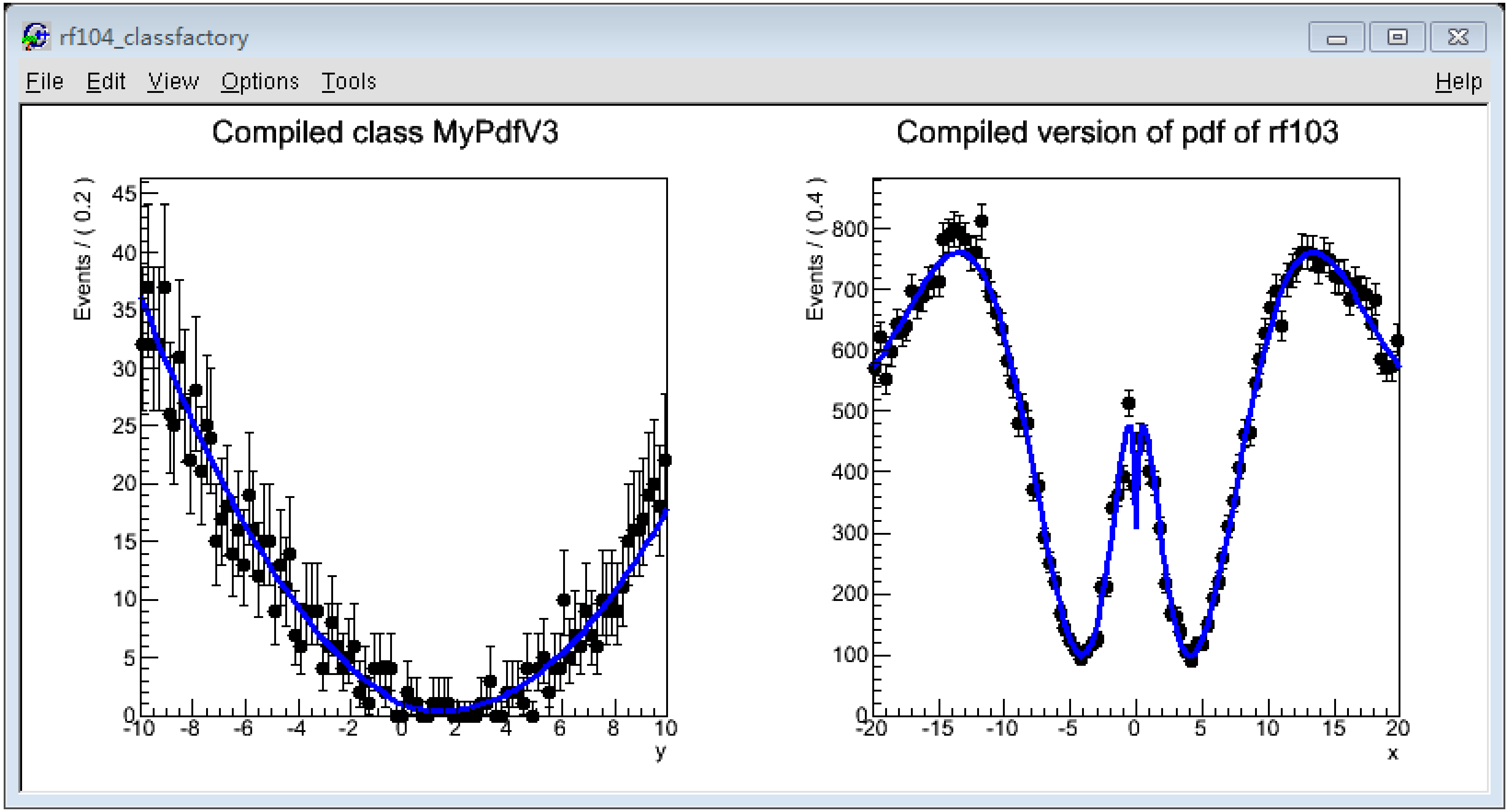


```
////////////////////////////////////////////////////////////////////////////
//
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #103
//
// Interpreted functions and p.d.f.s
//
//
//
// 07/2008 - Wouter Verkerke
//
///////////////////////////////////////////////////////////////////////////

#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 "RooFitResult.h"
#include "RooGenericPdf.h"
#include "RooConstVar.h"

using namespace RooFit ;
```

```
void rf103_interprfuncs()
{
  /////////////////////////////////////////////////////////
  // G e n e r i c   i n t e r p r e t e d   p . d . f . //
  /////////////////////////////////////////////////////////

  // Declare observable x
  RooRealVar x("x","x",-20,20) ;

  // C o n s t r u c t   g e n e r i c   p d f   f r o m   i n t e r p r e t e d   e x p r e s s i o n
  // -------------------------------------------------------------------------------------------------

  // To construct a proper p.d.f, the formula expression is explicitly normalized internally by dividing
  // it by a numeric integral of the expresssion over x in the range [-20,20]
  //
  RooRealVar alpha("alpha","alpha",5,0.1,10) ;
  RooGenericPdf genpdf("genpdf","genpdf","(1+0.1*abs(x)+sin(sqrt(abs(x*alpha+0.1))))",RooArgSet(x,alpha)) ;


  // S a m p l e ,   f i t   a n d   p l o t   g e n e r i c   p d f
  // ---------------------------------------------------------------

  // Generate a toy dataset from the interpreted p.d.f
  RooDataSet* data = genpdf.generate(x,10000) ;

  // Fit the interpreted p.d.f to the generated data
  genpdf.fitTo(*data) ;

  // Make a plot of the data and the p.d.f overlaid
  RooPlot* xframe = x.frame(Title("Interpreted expression pdf")) ;
  data->plotOn(xframe) ;
  genpdf.plotOn(xframe) ;
```

```
// Construct parameter mean2 and sigma
RooRealVar mean2("mean2","mean^2",10,0,200) ;
RooRealVar sigma("sigma","sigma",3,0.1,10) ;

// Construct interpreted function mean = sqrt(mean^2)
RooFormulaVar mean("mean","mean","sqrt(mean2)",mean2) ;

// Construct a gaussian g2(x,sqrt(mean2),sigma) ;
RooGaussian g2("g2","h2",x,mean,sigma) ;


// G e n e r a t e   t o y   d a t a
// ---------------------------------

// Construct a separate gaussian g1(x,10,3) to generate a toy Gaussian dataset with mean 10 and width 3
RooGaussian g1("g1","g1",x,RooConst(10),RooConst(3)) ;
RooDataSet* data2 = g1.generate(x,1000) ;


// F i t   a n d   p l o t   t a i l o r e d   s t a n d a r d   p d f
// -------------------------------------------------------------------

// Fit g2 to data from g1
RooFitResult* r = g2.fitTo(*data2,Save()) ;
r->Print() ;

// Plot data on frame and overlay projection of g2
RooPlot* xframe2 = x.frame(Title("Tailored Gaussian pdf")) ;
data2->plotOn(xframe2) ;
g2.plotOn(xframe2) ;


// Draw all frames on a canvas
TCanvas* c = new TCanvas("rf103_interprfuncs","rf103_interprfuncs",800,400) ;
c->Divide(2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; xframe->GetYaxis()->SetTitleOffset(1.4) ; xframe->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; xframe2->GetYaxis()->SetTitleOffset(1.4) ; xframe2->Draw() ;


}
```

rf104_classfactory.C



```
// NOTE: This demo uses code that is generated by the macro,
//       therefore it cannot be compiled in one step by ACliC.
//       To run this macro compiled with ACliC do
//
//         root>.x rf104_classfactory.C // run interpreted to generate code
//         root>.L MyPdfV3.cxx+         // Compile and load created classs
//         root>.x rf104_classfactory.C+ // run compiled code
//
//
// 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 "RooClassFactory.h"
#include "TROOT.h"

#ifndef __CINT__
#include "MyPdfV3.h"
#endif


using namespace RooFit ;
```

```
void rf104_classfactory()
{
  // W r i t e   c l a s s   s k e l e t o n   c o d e
  // --------------------------------------------------

  // Write skeleton p.d.f class with variable x,a,b
  // To use this class,
  //    - Edit the file MyPdfV1.cxx and implement the evaluate() method in terms of x,a and b
  //    - Compile and link class with '.x MyPdfV1.cxx+'
  //
  RooClassFactory::makePdf("MyPdfV1","x,A,B") ;


  // W i t h   a d d e d   i n i t i a l   v a l u e   e x p r e s s i o n
  // ---------------------------------------------------------------------

  // Write skeleton p.d.f class with variable x,a,b and given formula expression
  // To use this class,
  //    - Compile and link class with '.x MyPdfV2.cxx+'
  //
  RooClassFactory::makePdf("MyPdfV2","x,A,B","","A*fabs(x)+pow(x-B,2)") ;


  // W i t h   a d d e d   a n a l y t i c a l   i n t e g r a l   e x p r e s s i o n
  // ---------------------------------------------------------------------------------

  // Write skeleton p.d.f class with variable x,a,b, given formula expression _and_
  // given expression for analytical integral over x
  // To use this class,
  //    - Compile and link class with '.x MyPdfV3.cxx+'
  //
  RooClassFactory::makePdf("MyPdfV3","x,A,B","","A*fabs(x)+pow(x-B,2)",kTRUE,kFALSE,
                           "x:(A/2)*(pow(x.max(rangeName),2)+pow(x.min(rangeName),2))+(1./3)*(pow(x.max(rangeNam
e)-B,3)-pow(x.min(rangeName)-B,3))") ;
```

Bool_t makePdf(const char* name, const char* realArgNames = 0, const char* catArgNames = 0, const char* expression = "1.0", Bool_t hasAnaInt = kFALSE, Bool_t hasIntGen = kFALSE, const char* intExpression = 0)

```
  // Use instance of created class
  // ---------------------------------------------------

  // Compile MyPdfV3 class (only when running in CINT)
#ifdef __CINT__
  gROOT->ProcessLineSync(".x MyPdfV3.cxx+") ;
#endif

  // Creat instance of MyPdfV3 class
  RooRealVar a("a","a",1) ;
  RooRealVar b("b","b",2,-10,10) ;
  RooRealVar y("y","y",-10,10);
  MyPdfV3 pdf("pdf","pdf",y,a,b) ;

  // Generate toy data from pdf and plot data and p.d.f on frame
  RooPlot* frame1 = y.frame(Title("Compiled class MyPdfV3")) ;
  RooDataSet* data = pdf.generate(y,1000) ;
  pdf.fitTo(*data) ;
  data->plotOn(frame1) ;
  pdf.plotOn(frame1) ;
```

一定让同学们实现自己做拟合的Class

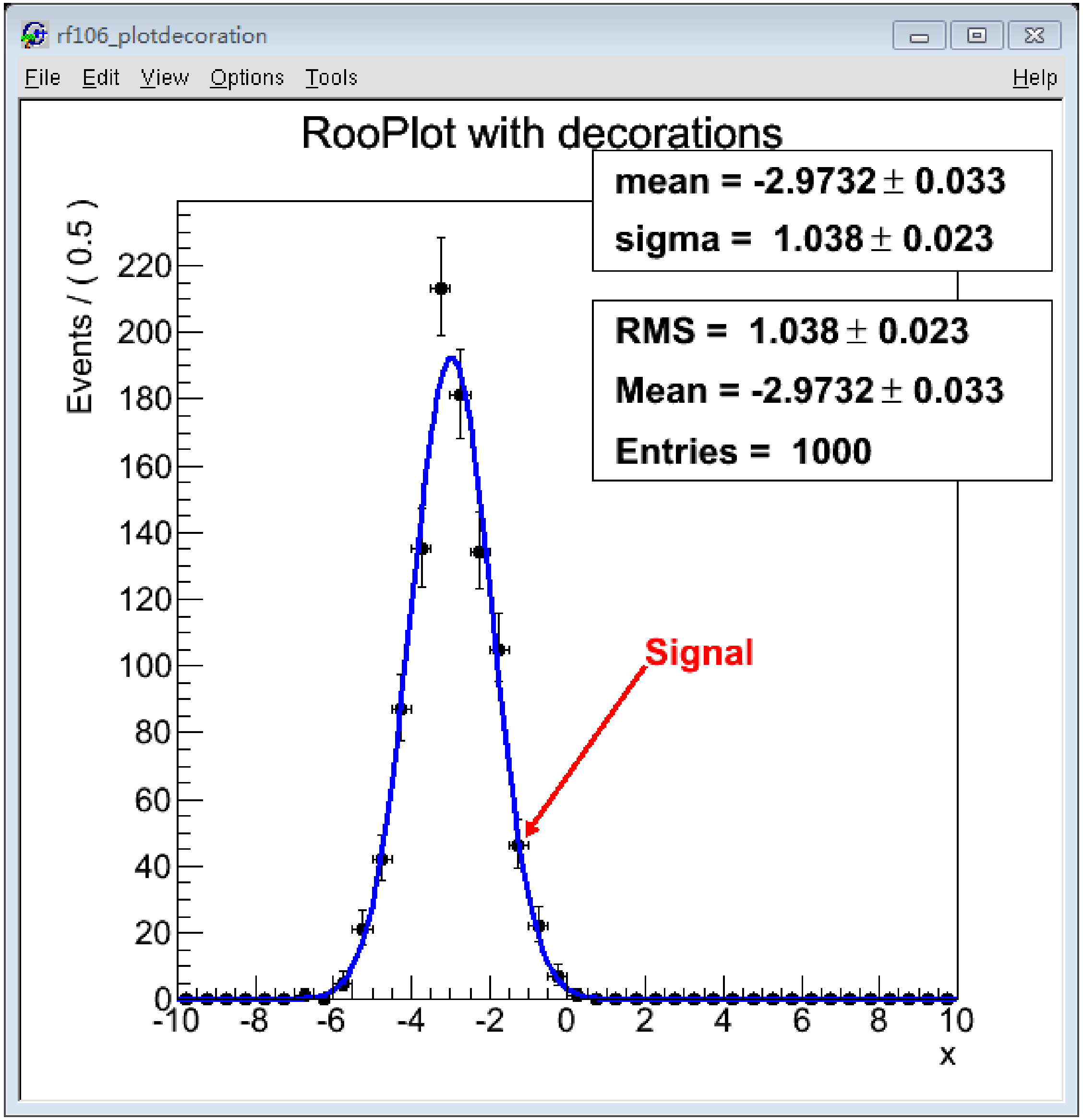
用rootlogon.C函数内写编译后的.so文件
注：用gSystem->Load("…….so");

修改rf104_classfactory.C,把自己编译的文件当成RooFit自带的文件进行参加拟合

siguang@pku.edu.cn

```
// C o m p i l e d   v e r s i o n   o f   e x a m p l e   r f 1 0 3 //
////////////////////////////////////////////////////////////////////////

// Declare observable x
RooRealVar x("x","x",-20,20) ;

// The RooClassFactory::makePdfInstance() function performs code writing, compiling, linking
// and object instantiation in one go and can serve as a straight replacement of RooGenericPdf

RooRealVar alpha("alpha","alpha",5,0.1,10) ;
RooAbsPdf* genpdf = RooClassFactory::makePdfInstance("GenPdf","(1+0.1*fabs(x)+sin(sqrt(fabs(x*alpha+0.1))))",R
ooArgSet(x,alpha)) ;

// Generate a toy dataset from the interpreted p.d.f
RooDataSet* data2 = genpdf->generate(x,50000) ;

// Fit the interpreted p.d.f to the generated data
genpdf->fitTo(*data2) ;

// Make a plot of the data and the p.d.f overlaid
RooPlot* frame2 = x.frame(Title("Compiled version of pdf of rf103")) ;
data2->plotOn(frame2) ;
genpdf->plotOn(frame2) ;

// Draw all frames on a canvas
TCanvas* c = new TCanvas("rf104_classfactory","rf104_classfactory",800,400) ;
c->Divide(2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame1->GetYaxis()->SetTitleOffset(1.4) ; frame1->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.4) ; frame2->Draw() ;

}
```

rf106_plotdecoration.C



```
//
// '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 ;
```

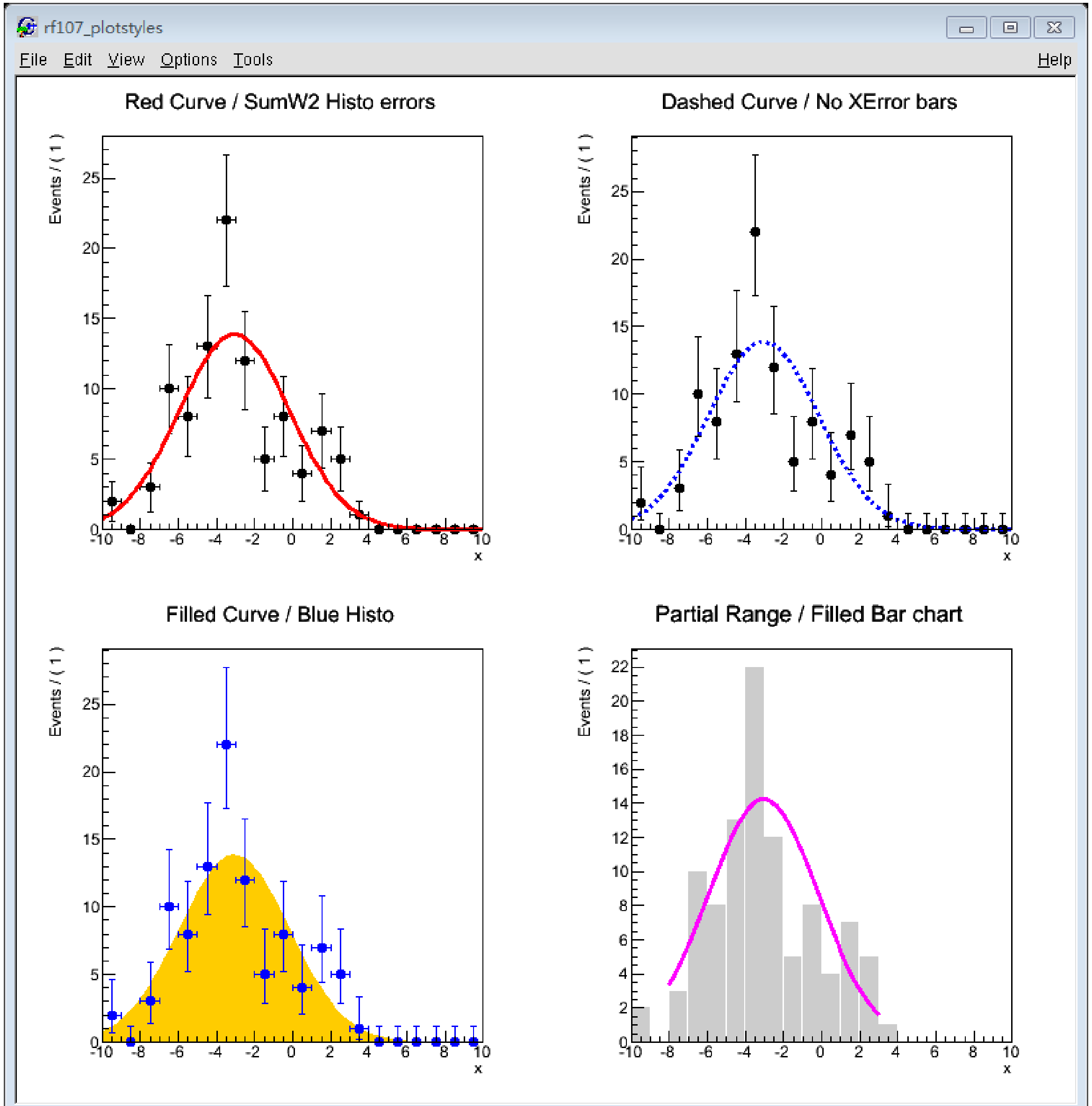
```
void rf106_plotdecoration()
{

  // 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
  gauss.fitTo(*data) ;


  // P l o t   p . d . f   a n d   d a t a
  // -------------------------------------

  // Overlay projection of gauss on data
  RooPlot* frame = x.frame(Name("xframe"),Title("RooPlot with decorations"),Bins(40)) ;
  data->plotOn(frame) ;
  gauss.plotOn(frame) ;
```

data->plotOn(frame,Binning(40));

```
// A d d   b o x   w i t h   p d f   p a r a m e t e r s
// -----------------------------------------------------

// Left edge of box starts at 55% of Xaxis)
gauss.paramOn(frame,Layout(0.55)) ;


// A d d   b o x   w i t h   d a t a   s t a t i s t i c s
// -------------------------------------------------------

// X size of box is from 55% to 99% of Xaxis range, top of box is at 80% of Yaxis range)
data->statOn(frame,Layout(0.55,0.99,0.8)) ;


// A d d   t e x t   a n d   a r r o w
// -----------------------------------

// Add text to frame
TText* txt = new TText(2,100,"Signal") ;
txt->SetTextSize(0.04) ;
txt->SetTextColor(kRed) ;
frame->addObject(txt) ;

// Add arrow to frame
TArrow* arrow = new TArrow(2,100,-1,50,0.01,"|>") ;
arrow->SetLineColor(kRed) ;
arrow->SetFillColor(kRed) ;
arrow->SetLineWidth(3) ;
frame->addObject(arrow) ;
```

```
// Persist frame with all decorations in ROOT file
// ---------------------------------------------------------------

TFile f("rf106_plotdecoration.root","RECREATE") ;
frame->Write() ;
f.Close() ;

// To read back and plot frame with all decorations in clean root session do
// root> TFile f("rf106_plotdecoration.root") ;
// root>  xframe->Draw() ;

new TCanvas("rf106_plotdecoration","rf106_plotdecoration",600,600) ;
gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.6) ; frame->Draw() ;

}
```

rf107_plotstyles.C



```
// 'BASIC FUNCTIONALITY' RooFit tutorial macro #107
//
//   Demonstration of various plotting styles of data, functions
//   in a RooPlot
//
// 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"
using namespace RooFit ;
```

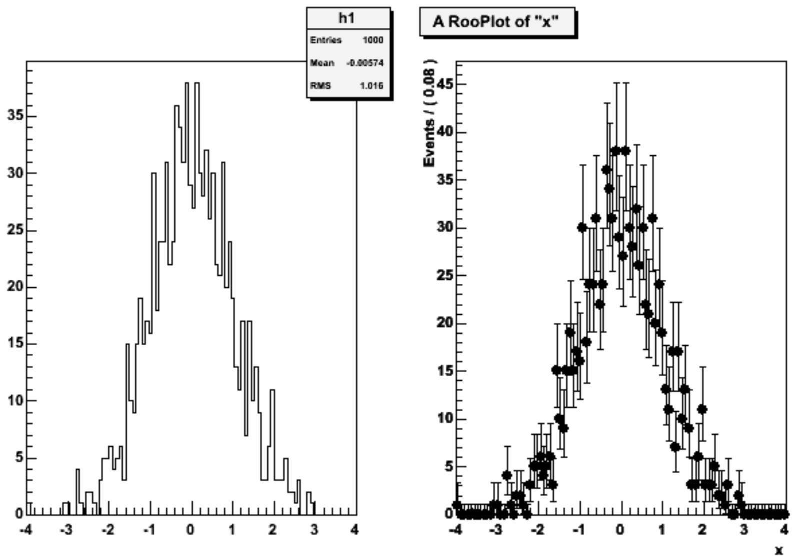
```
void rf107_plotstyles()
{

  // 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",-3,-10,10) ;
  RooGaussian gauss("gauss","gauss",x,mean,sigma) ;

  // Generate a sample of 100 events with sigma=3
  RooDataSet* data = gauss.generate(x,100) ;

  // Fit pdf to data
  gauss.fitTo(*data) ;



  // M a k e   p l o t   f r a m e s
  // -------------------------------

  // Make four plot frames to demonstrate various plotting features
  RooPlot* frame1 = x.frame(Name("xframe"),Title("Red Curve / SumW2 Histo errors"),Bins(20)) ;
  RooPlot* frame2 = x.frame(Name("xframe"),Title("Dashed Curve / No XError bars"),Bins(20)) ;
  RooPlot* frame3 = x.frame(Name("xframe"),Title("Filled Curve / Blue Histo"),Bins(20)) ;
  RooPlot* frame4 = x.frame(Name("xframe"),Title("Partial Range / Filled Bar chart"),Bins(20)) ;
```

```
// D a t a   p l o t t i n g   s t y l e s
// ---------------------------------------

// Use sqrt(sum(weights^2)) error instead of Poisson errors
data->plotOn(frame1,DataError(RooAbsData::SumW2)) ;

// Remove horizontal error bars
data->plotOn(frame2,XErrorSize(0)) ;

// Blue markers and error bors
data->plotOn(frame3,MarkerColor(kBlue),LineColor(kBlue)) ;

// Filled bar chart
data->plotOn(frame4,DrawOption("B"),DataError(RooAbsData::None),XErrorSize(0),FillColor(kGray)) ;


// F u n c t i o n   p l o t t i n g   s t y l e s
// -----------------------------------------------

// Change line color to red
gauss.plotOn(frame1,LineColor(kRed)) ;

// Change line style to dashed
gauss.plotOn(frame2,LineStyle(kDashed)) ;

// Filled shapes in green color
gauss.plotOn(frame3,DrawOption("F"),FillColor(kOrange),MoveToBack()) ;

//
gauss.plotOn(frame4,Range(-8,3),LineColor(kMagenta)) ;
```

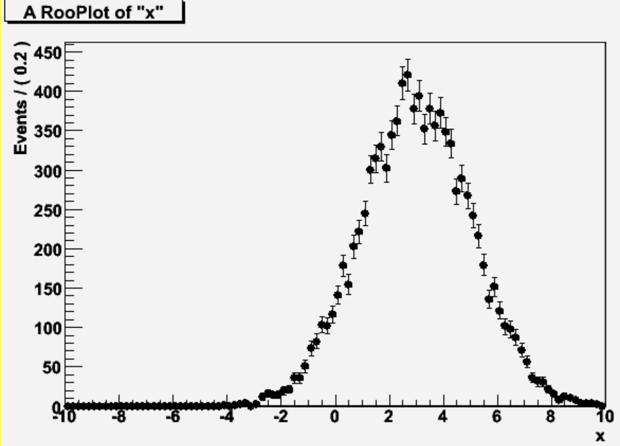
```
  TCanvas* c = new TCanvas("rf107_plotstyles","rf107_plotstyles",800,800) ;
  c->Divide(2,2) ;
  c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame1->GetYaxis()->SetTitleOffset(1.6) ; frame1->Draw() ;
  c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.6) ; frame2->Draw() ;
  c->cd(3) ; gPad->SetLeftMargin(0.15) ; frame3->GetYaxis()->SetTitleOffset(1.6) ; frame3->Draw() ;
  c->cd(4) ; gPad->SetLeftMargin(0.15) ; frame4->GetYaxis()->SetTitleOffset(1.6) ; frame4->Draw() ;


}
```

Binned data (histograms)

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

 TH1F *h1 = new TH1F("h1","",100,-4,4);
 h1 -> FillRandom("gaus",1000);
 RooRealVar x("x","x",-4,4);
 RooDataHist data("data", "Dataset With X",x,h1);
 RooPlot* frame = x.frame();
 data.plotOn(frame);

 TCanvas *c1 = new TCanvas("c1","");
 c1 -> SetFillColor(10);
 c1->Divide(2,1);
 c1->cd(1);
 h1->Draw();
 c1->cd(2);
 frame->Draw();
c1->cd();
}
```



Unbinned data (add)

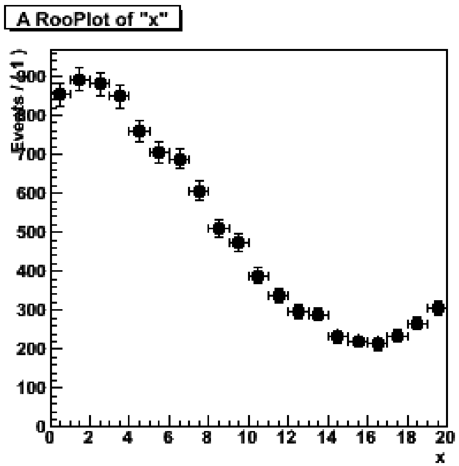



```
void UnbinnedData_add(){
 gSystem->Load("libRooFit");
 using namespace RooFit;

 RooRealVar x("x","x",-10,10);
 RooDataSet data("data","Dataset With X",x);

 for(Int_t j=0; j<10000; j++){
  x = gRandom->Gaus(3.0,2);//mean=-1, sigma =2
  data.add(x);
 }

 RooPlot* frame = x.frame();
 data.plotOn(frame,Binning(100));
 frame->Draw();
}
```

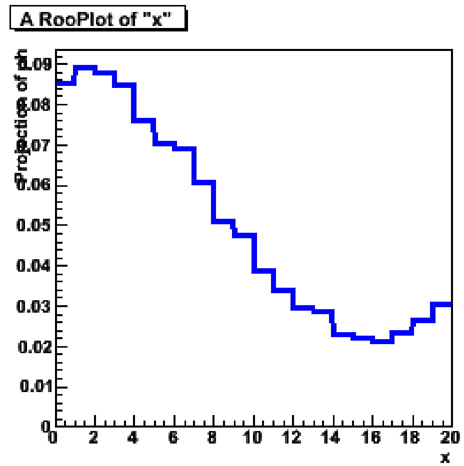
Highlight of non-parametric shapes - histograms

- Will highlight two types of non-parametric p.d.f.s
- Class **`RooHistPdf`** – a p.d.f. described by a histogram

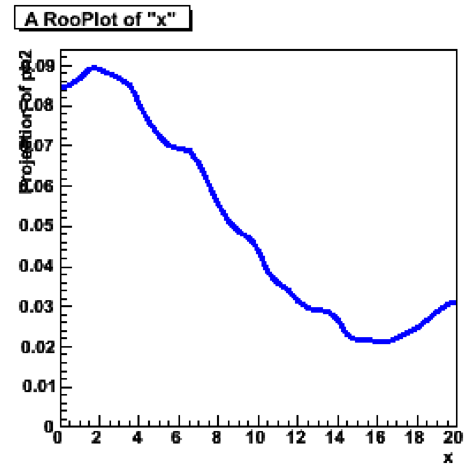
dataHist



RooHistPdf(N=0)



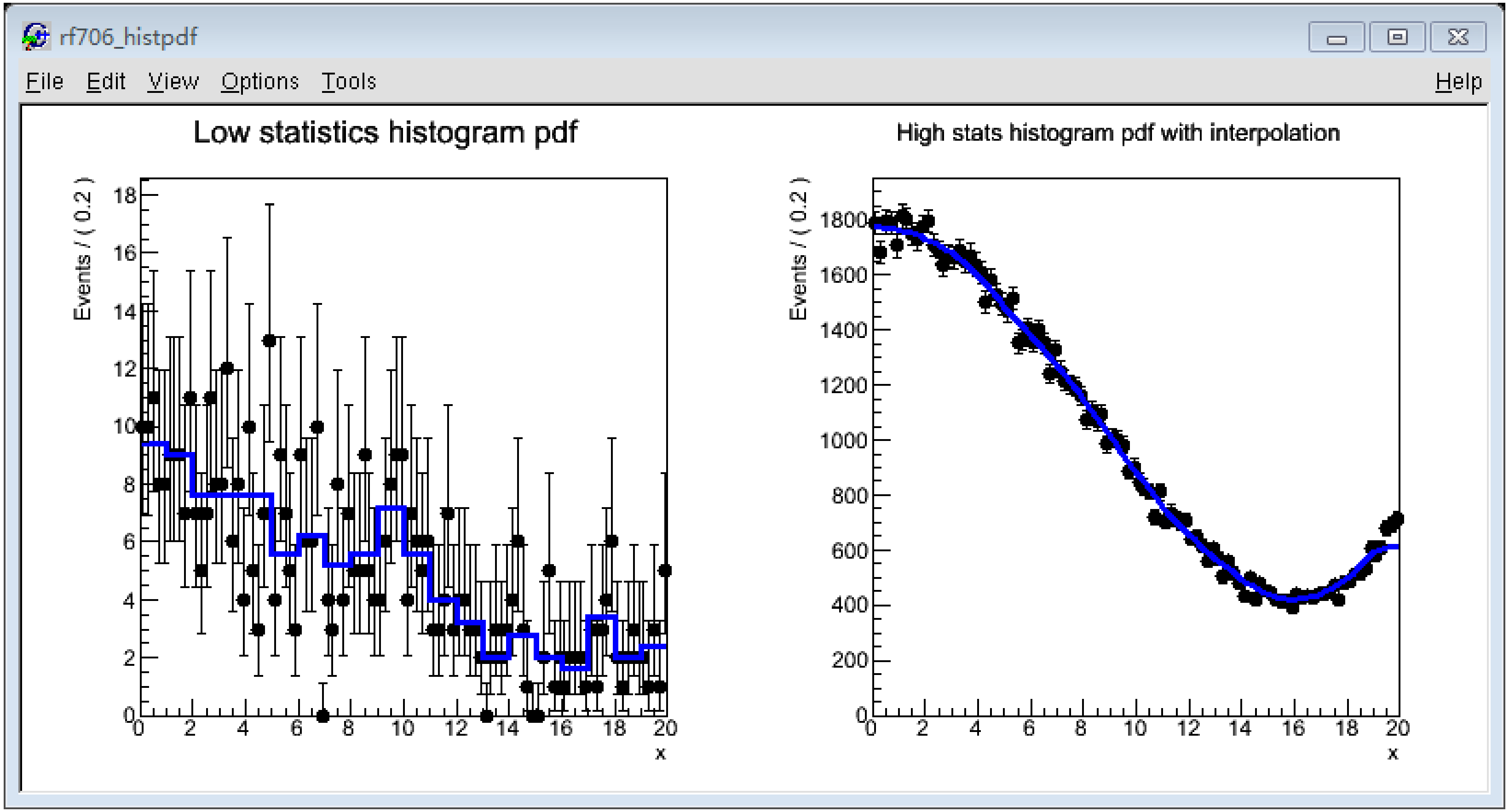
RooHistPdf(N=4)




```
// Histogram based p.d.f with N-th order interpolation
RooHistPdf ph("ph","ph",x,*dataHist,N) ;
```

- Not so great at low statistics (especially problematic in >1 dim)

rf706_histpdf.C

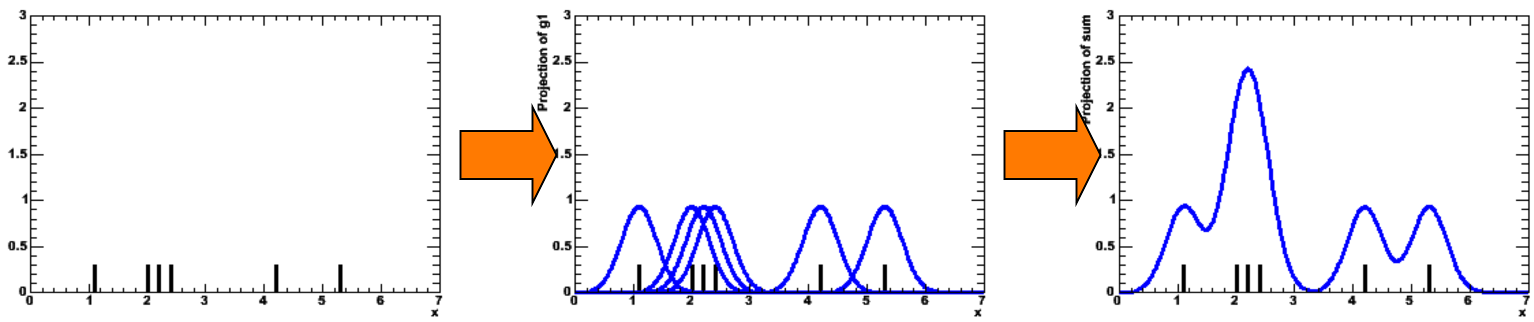


- Class **RooKeysPdf** – A kernel estimation p.d.f.
  - Uses *unbinned* data
  - Idea represent each event of your MC sample as a Gaussian probability distribution
  - Add probability distributions from all events in sample

Sample of events

Gaussian probability distributions for each event

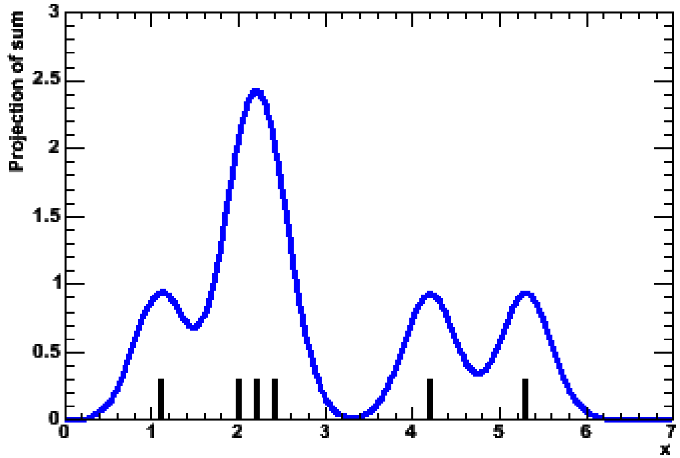
Summed probability distribution for all events in sample



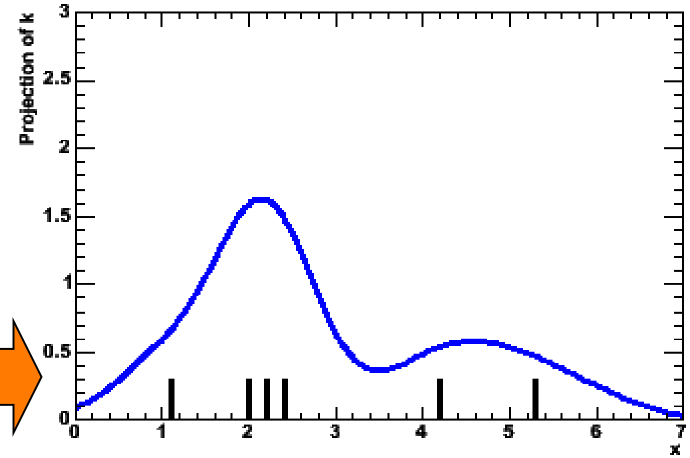
Highlight of non-parametric shapes – kernel estimation

- Width of Gaussian kernels need not be the same for all events
  - As long as each event contributes 1/N to the integral
- Idea: 'Adaptive kernel' technique
  - Choose wide Gaussian if local density of events is low
  - Choose narrow Gaussian if local density of events is high
  - Preserves small features in high statistics areas, minimize jitter in low statistics areas
  - Automatically calculated

Static Kernel
(with of all Gaussian identical)



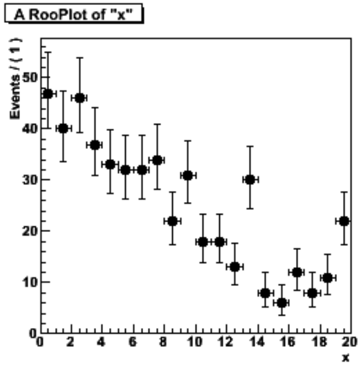
Adaptive Kernel
(width of all Gaussian depends
on local density of events)



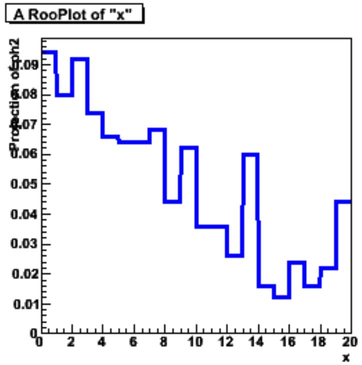
```
// Adaptive kernel estimation p.d.f
 RooKeysPdf k("k","k",x,*d,RooKeysPdf::MirrorBoth) ;
```

- Example with comparison to histogram based p.d.f
  - Superior performance at low statistics
  - Can mirror input data over boundaries to reduce 'edge leakage'
  - Works also in >1 dimensions (class `RooNDKeysPdf`)

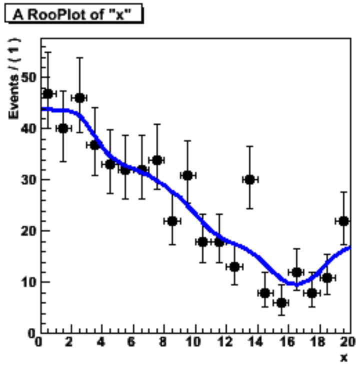
Data (N=500)



RooHistPdf(data)

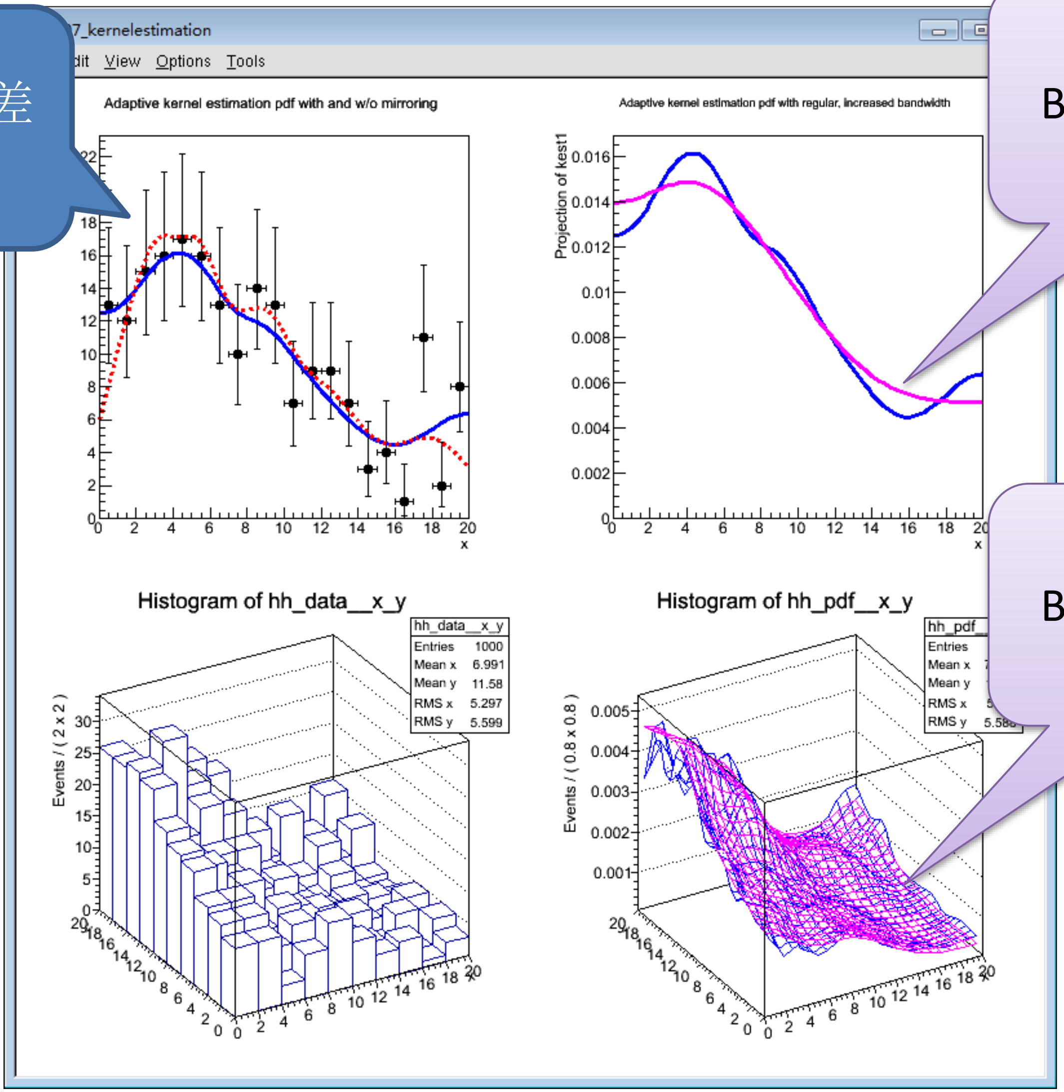


RooKeysPdf(data)



rf707_kernelestimation.C

是否镜像的差别



Bandwidth=2

Bandwidth=2

rf707_kernelestimation.C

```
void rf707_kernelestimation()
{
  // C r e a t e   l o w   s t a t s   1 - D   d a t a s e t
  // -------------------------------------------------------

  // Create a toy pdf for sampling
  RooRealVar x("x","x",0,20) ;
  RooPolynomial p("p","p",x,RooArgList(RooConst(0.01),RooConst(-0.01),RooConst(0.0004))) ;

  // Sample 500 events from p
  RooDataSet* data1 = p.generate(x,200) ;



  // C r e a t e   1 - D   k e r n e l   e s t i m a t i o n   p d f
  // ---------------------------------------------------------------

  // Create adaptive kernel estimation pdf. In this configuration the input data
  // is mirrored over the boundaries to minimize edge effects in distribution
  // that do not fall to zero towards the edges
  RooKeysPdf kest1("kest1","kest1",x,*data1,RooKeysPdf::MirrorBoth) ;

  // An adaptive kernel estimation pdf on the same data without mirroring option
  // for comparison
  RooKeysPdf kest2("kest2","kest2",x,*data1,RooKeysPdf::NoMirror) ;
```

rf707_kernelestimation.C

2！ 比缺省光滑！

```
// Adaptive kernel estimation pdf with increased bandwidth scale factor
// (promotes smoothness over detail preservation)
RooKeysPdf kest3("kest1","kest1",x,*data1,RooKeysPdf::MirrorBoth,2) ;


// Plot kernel estimation pdfs with and without mirroring over data
RooPlot* frame = x.frame(Title("Adaptive kernel estimation pdf with and w/o mirroring"),Bins(20)) ;
data1->plotOn(frame) ;
kest1.plotOn(frame) ;
kest2.plotOn(frame,LineStyle(kDashed),LineColor(kRed)) ;


// Plot kernel estimation pdfs with regular and increased bandwidth
RooPlot* frame2 = x.frame(Title("Adaptive kernel estimation pdf with regular, increased bandwidth"))
kest1.plotOn(frame2) ;
kest3.plotOn(frame2,LineColor(kMagenta)) ;
```

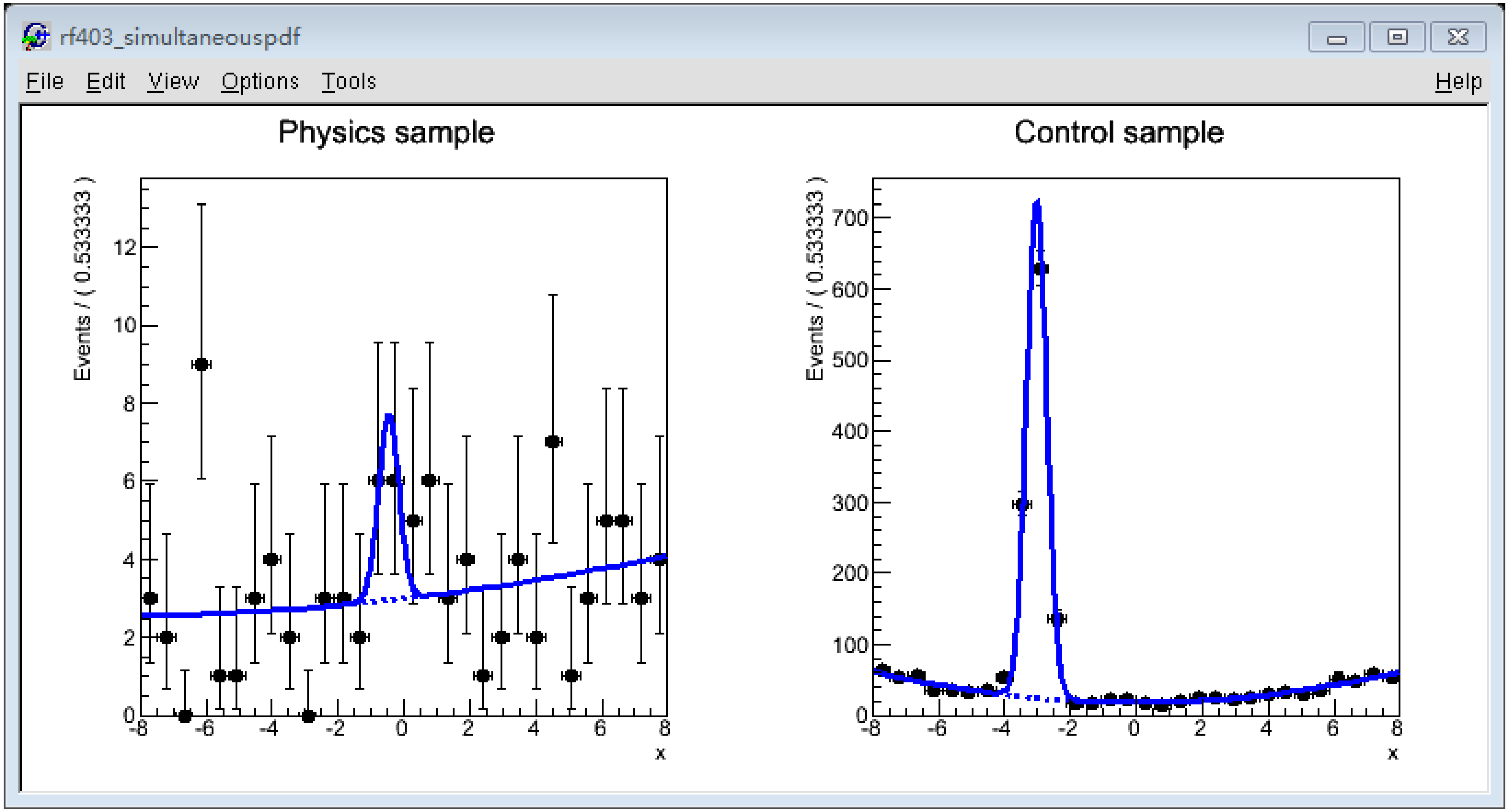
rf707_kernelestimation.C

```
// Construct a 2D toy pdf for sampleing
RooRealVar y("y","y",0,20) ;
RooPolynomial py("py","py",y,RooArgList(RooConst(0.01),RooConst(0.01),RooConst(-0.0004))) ;
RooProdPdf pxy("pxy","pxy",RooArgSet(p,py)) ;
RooDataSet* data2 = pxy.generate(RooArgSet(x,y),1000) ;



// C r e a t e   2 - D   k e r n e l   e s t i m a t i o n   p d f
// ---------------------------------------------------------------

// Create 2D adaptive kernel estimation pdf with mirroring
RooNDKeysPdf kest4("kest4","kest4",RooArgSet(x,y),*data2,"am") ;


// Create 2D adaptive kernel estimation pdf with mirroring and double bandwidth
RooNDKeysPdf kest5("kest5","kest5",RooArgSet(x,y),*data2,"am",2) ;
```

rf707_kernelestimation.C

```
// Create a histogram of the data
TH1* hh_data = data2->createHistogram("hh_data",x,Binning(10),YVar(y,Binning(10))) ;

// Create histogram of the 2d kernel estimation pdfs
TH1* hh_pdf = kest4.createHistogram("hh_pdf",x,Binning(25),YVar(y,Binning(25))) ;
TH1* hh_pdf2 = kest5.createHistogram("hh_pdf2",x,Binning(25),YVar(y,Binning(25))) ;
hh_pdf->SetLineColor(kBlue) ;
hh_pdf2->SetLineColor(kMagenta) ;



TCanvas* c = new TCanvas("rf707_kernelestimation","rf707_kernelestimation",800,800) ;
c->Divide(2,2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4) ; frame->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.8) ; frame2->Draw() ;
c->cd(3) ; gPad->SetLeftMargin(0.15) ; hh_data->GetZaxis()->SetTitleOffset(1.4) ; hh_data->Draw("lego") ;
c->cd(4) ; gPad->SetLeftMargin(0.20) ; hh_pdf->GetZaxis()->SetTitleOffset(2.4) ; hh_pdf->Draw("surf") ; hh_pd
f2->Draw("surfsame") ;
```

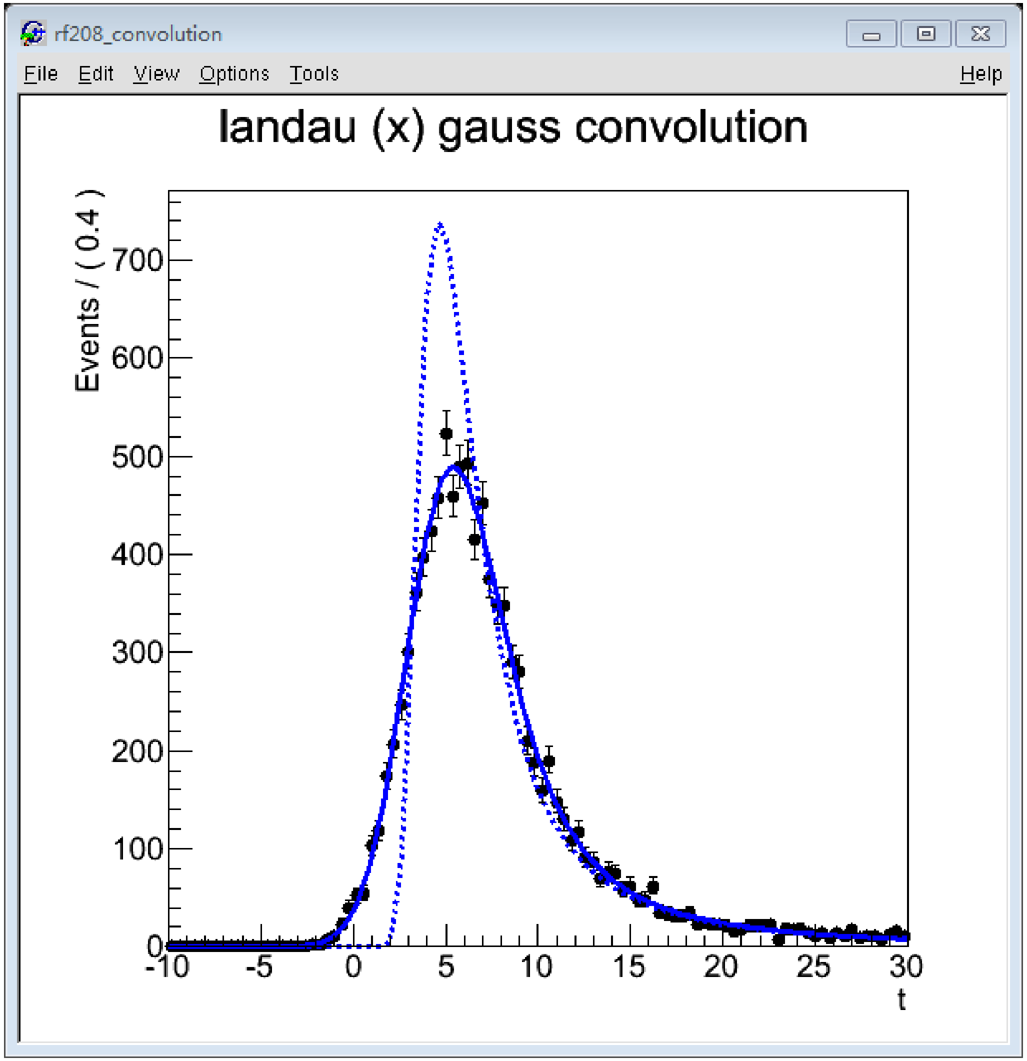
rf501_simultaneouspdf.C



rf501_simultaneouspdf.C

```
#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 "RooSimultaneous.h"
#include "RooCategory.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "RooPlot.h"
using namespace RooFit ;


void rf501_simultaneouspdf()
{
  // C r e a t e   m o d e l   f o r   p h y s i c s   s a m p l e
  // -------------------------------------------------------------

  // Create observables
  RooRealVar x("x","x",-8,8) ;
```

rf208_convolution.C



rf208_convolution.C

```
void rf208_convolution()
{
  // S e t u p   c o m p o n e n t   p d f s
  // ---------------------------------------

  // Construct observable
  RooRealVar t("t","t",-10,30) ;

  // Construct landau(t,ml,sl) ;
  RooRealVar ml("ml","mean landau",5.,-20,20) ;
  RooRealVar sl("sl","sigma landau",1,0.1,10) ;
  RooLandau landau("lx","lx",t,ml,sl) ;

  // Construct gauss(t,mg,sg)
  RooRealVar mg("mg","mg",0) ;
  RooRealVar sg("sg","sg",2,0.1,10) ;
  RooGaussian gauss("gauss","gauss",t,mg,sg) ;
```

```
// Set #bins to be used for FFT sampling to 10000
t.setBins(10000,"cache") ;

// Construct landau (x) gauss
RooFFTConvPdf lxg("lxg","landau (X) gauss",t,landau,gauss) ;
```

rf208_convolution.C

```
// Sample 1000 events in x from gxlx
RooDataSet* data = lxg.generate(t,10000) ;

// Fit gxlx to data
lxg.fitTo(*data) ;

// Plot data, landau pdf, landau (X) gauss pdf
RooPlot* frame = t.frame(Title("landau (x) gauss convolution")) ;
data->plotOn(frame) ;
lxg.plotOn(frame) ;
landau.plotOn(frame,LineStyle(kDashed)) ;
```

```
  // Draw frame on canvas
  new TCanvas("rf208_convolution","rf208_convolution",600,600) ;
  gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4) ; frame->Draw() ;

}
```

Install Root with FFTW3

- tar -zxvf fftw-3.2.2.tar.gz
- cd fftw-3.2.2
- fftw-3.2.2 > ./configure --prefix=/ClusterDisks/HDN13/WorkSpace/testroot/root/fftw
- make –j7
- make install

```
[hepfarm02] /ClusterDisks/HDN13/WorkSpace/testroot/root/fftw > ll
total 32
drwxr-xr-x  2 testroot testroot 4096 Nov 19  2013 bin
drwxr-xr-x  2 testroot testroot 4096 Nov 19  2013 include
drwxr-xr-x  3 testroot testroot 4096 Nov 19  2013 lib
drwxr-xr-x  4 testroot testroot 4096 Nov 19  2013 share
```

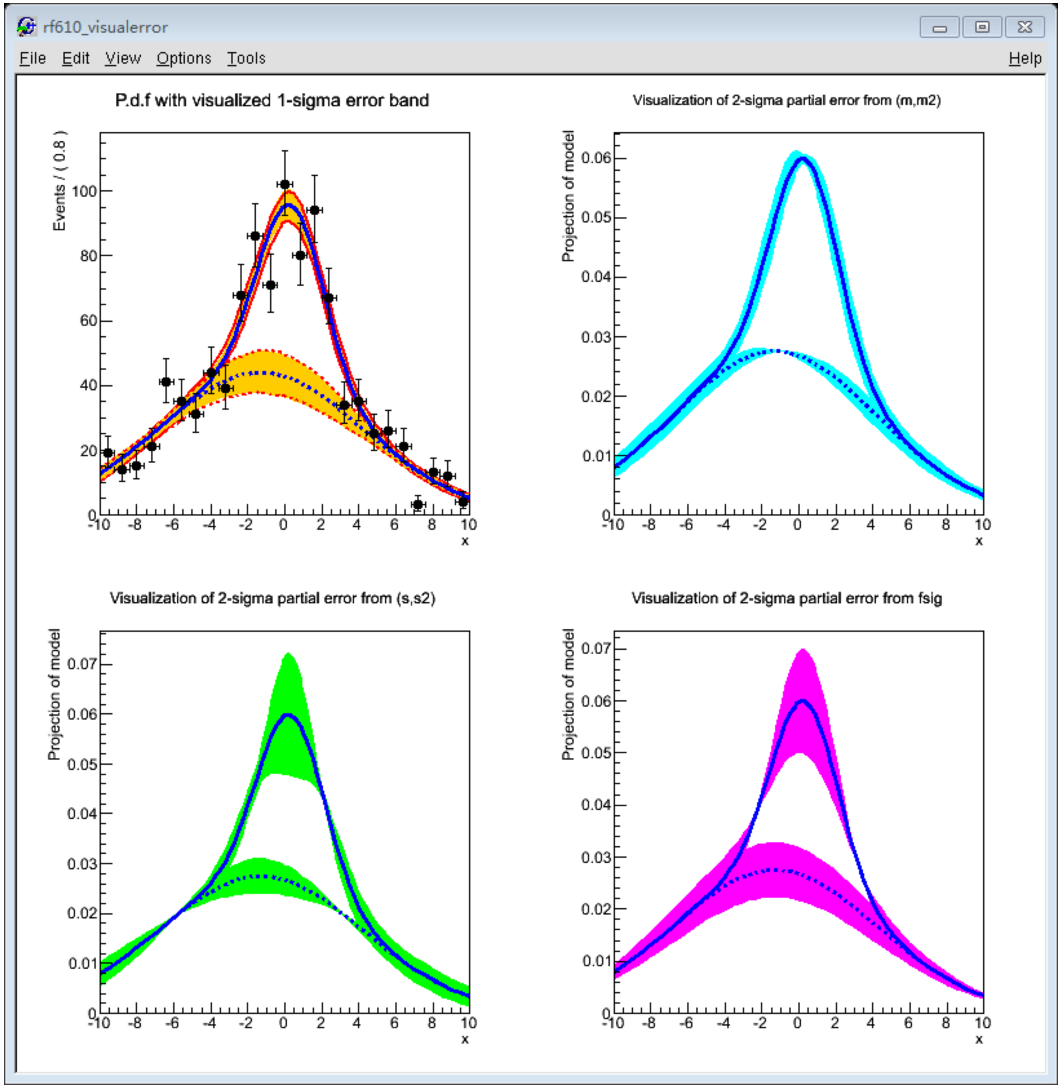
Install Root with FFTW3

```
#!/bin/bash

export ROOTSYS=/ClusterDisks/HDN13/WorkSpace/testroot/root/528

#if you have not un-zipped the file, remove the # at the head of next line
#tar -zxvf root_v5.28.00.source.tar.gz
cd root
./configure  --enable-roofit --enable-fftw3  --with-fftw3-incdir=/ClusterDisks/HDN13/WorkSpace/testroot/root/fftw/include --with-fftw3-libdir=/ClusterDisks/HDN13/WorkSpace/testroot/root/fftw/lib

make –j7
make install
```

RooVoigtian

RooVoigtian(const char* name, const char* title, RooAbsReal& _x, RooAbsReal& _mean, RooAbsReal& _width,RooAbsReal& _sigma, Bool_t doFast = kFALSE)

rf610_visualerror.C



rf610_visualerror.C

```
// Create sum of two Gaussians p.d.f. with factory
RooRealVar x("x","x",-10,10) ;

RooRealVar m("m","m",0,-10,10) ;
RooRealVar s("s","s",2,1,50) ;
RooGaussian sig("sig","sig",x,m,s) ;

RooRealVar m2("m2","m2",-1,-10,10) ;
RooRealVar s2("s2","s2",6,1,50) ;
RooGaussian bkg("bkg","bkg",x,m2,s2) ;

RooRealVar fsig("fsig","fsig",0.33,0,1) ;
RooAddPdf model("model","model",RooArgList(sig,bkg),fsig) ;


// Create binned dataset
x.setBins(25) ;
RooAbsData* d = model.generateBinned(x,1000) ;

// Perform fit and save fit result
RooFitResult* r = model.fitTo(*d,Save()) ;

// Make plot frame
RooPlot* frame = x.frame(Bins(40),Title("P.d.f with visualized 1-sigma error band"
d->plotOn(frame) ;
```

rf610_visualerror.C

```
// Visualize 1-sigma error encoded in fit result 'r' as orange band using linear error propagation
// This results in an error band that is by construction symmetric
//
// The linear error is calculated as
// error(x) = Z* F_a(x) * Corr(a,a') F_a'(x)
//
// where     F_a(x) = [ f(x,a+da) - f(x,a-da) ] / 2,
//
//         with f(x) = the plotted curve
//              'da' = error taken from the fit result
//        Corr(a,a') = the correlation matrix from the fit result
//                Z = requested significance 'Z sigma band'
//
// The linear method is fast (required 2*N evaluations of the curve, where N is the number of parameters),
// but may not be accurate in the presence of strong correlations (~>0.9) and at Z>2 due to linear and
// Gaussian approximations made
//
model.plotOn(frame,VisualizeError(*r,1),FillColor(kOrange)) ;
```

rf610_visualerror.C

```
// Calculate error using sampling method and visualize as dashed red line.
//
// In this method a number of curves is calculated with variations of the parameter values, as sampled
// from a multi-variate Gaussian p.d.f. that is constructed from the fit results covariance matrix.
// The error(x) is determined by calculating a central interval that capture N% of the variations
// for each valye of x, where N% is controlled by Z (i.e. Z=1 gives N=68%). The number of sampling curves
// is chosen to be such that at least 100 curves are expected to be outside the N% interval, and is minimally
// 100 (e.g. Z=1->Ncurve=356, Z=2->Ncurve=2156)) Intervals from the sampling method can be asymmetric,
// and may perform better in the presence of strong correlations, but may take (much) longer to calculate
model.plotOn(frame,VisualizeError(*r,1,kFALSE),DrawOption("L"),LineWidth(2),LineColor(kRed)) ;
```

rf610_visualerror.C

```
  // Perform the same type of error visualization on the background component only.
  // The VisualizeError() option can generally applied to _any_ kind of plot (components, asymmetries, efficiencies etc.
.)
  model.plotOn(frame,VisualizeError(*r,1),FillColor(kOrange),Components("bkg")) ;
  model.plotOn(frame,VisualizeError(*r,1,kFALSE),DrawOption("L"),LineWidth(2),LineColor(kRed),Components("bkg"),LineStyl
e(kDashed)) ;

  // Overlay central value
  model.plotOn(frame) ;
  model.plotOn(frame,Components("bkg"),LineStyle(kDashed)) ;
  d->plotOn(frame) ;
  frame->SetMinimum(0) ;
```


siguang@pku.edu.cn

rf610_visualerror.C

```
RooPlot* frame2 = x.frame(Bins(40),Title("Visualization of 2-sigma partial error from (m,m2)")) ;

// Visualize partial error. For partial error visualization the covariance matrix is first reduced as follows
//        ___                   -1
// Vred = V22  = V11 - V12 * V22   * V21
//
// Where V11,V12,V21,V22 represent a block decomposition of the covariance matrix into observables that
// are propagated (labeled by index '1') and that are not propagated (labeled by index '2'), and V22bar
// is the Shur complement of V22, calculated as shown above
//
// (Note that Vred is _not_ a simple sub-matrix of V)

// Propagate partial error due to shape parameters (m,m2) using linear and sampling method
model.plotOn(frame2,VisualizeError(*r,RooArgSet(m,m2),2),FillColor(kCyan)) ;
model.plotOn(frame2,Components("bkg"),VisualizeError(*r,RooArgSet(m,m2),2),FillColor(kCyan)) ;

model.plotOn(frame2) ;
model.plotOn(frame2,Components("bkg"),LineStyle(kDashed)) ;
frame2->SetMinimum(0) ;
```

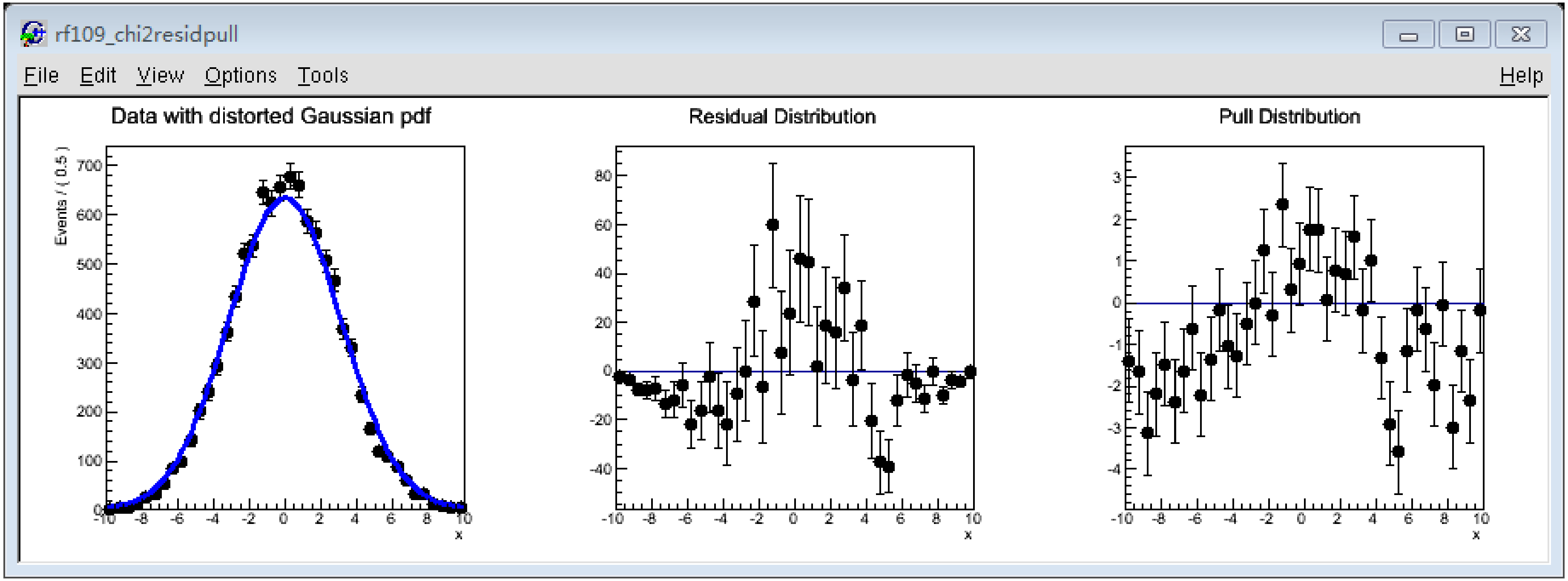
rf610_visualerror.C

```
// Make plot frame
RooPlot* frame3 = x.frame(Bins(40),Title("Visualization of 2-sigma partial error from (s,s2)")) ;

// Propagate partial error due to yield parameter using linear and sampling method
model.plotOn(frame3,VisualizeError(*r,RooArgSet(s,s2),2),FillColor(kGreen)) ;
model.plotOn(frame3,Components("bkg"),VisualizeError(*r,RooArgSet(s,s2),2),FillColor(kGreen)) ;

model.plotOn(frame3) ;
model.plotOn(frame3,Components("bkg"),LineStyle(kDashed)) ;
frame3->SetMinimum(0) ;
```

rf610_visualerror.C

```
// Make plot frame
RooPlot* frame4 = x.frame(Bins(40),Title("Visualization of 2-sigma partial error from fsig")) ;

// Propagate partial error due to yield parameter using linear and sampling method
model.plotOn(frame4,VisualizeError(*r,RooArgSet(fsig),2),FillColor(kMagenta)) ;
model.plotOn(frame4,Components("bkg"),VisualizeError(*r,RooArgSet(fsig),2),FillColor(kMagenta)) ;

model.plotOn(frame4) ;
model.plotOn(frame4,Components("bkg"),LineStyle(kDashed)) ;
frame4->SetMinimum(0) ;

 TCanvas* c = new TCanvas("rf610_visualerror","rf610_visualerror",800,800) ;
 c->Divide(2,2) ;
 c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4)  ; frame->Draw() ;
 c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.6) ; frame2->Draw() ;
 c->cd(3) ; gPad->SetLeftMargin(0.15) ; frame3->GetYaxis()->SetTitleOffset(1.6) ; frame3->Draw() ;
 c->cd(4) ; gPad->SetLeftMargin(0.15) ; frame4->GetYaxis()->SetTitleOffset(1.6) ; frame4->Draw() ;
}
```

rf109_chi2residpull.C

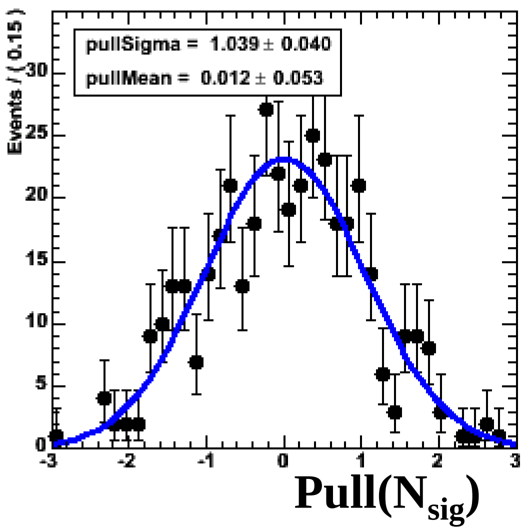


$$\mathrm{pull}(\mathrm{N_{sig}}) = \frac{N_{sig}^{fit} - N_{sig}^{true}}{\sigma_N^{fit}}$$

Pull分析

检查"Pull"分布,看误差是否对称

$$Pull(N_{sig}) = \frac{N_{sig}^{fit} - N_{sig}^{true}}{\sigma_{sig+bg}^{fit}}$$



a) 如果拟合结果无偏则均值为0;
b) 如果拟合误差正确则宽度为1。

```
// '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) ;

  // Change sigma to 3.15
  sigma=3.15 ;


  // P l o t   d a t a   a n d   s l i g h t l y   d i s t o r t e d   m o d e l
  // ---------------------------------------------------------------------------

  // Overlay projection of gauss with sigma=3.15 on data with sigma=3.0
  RooPlot* frame1 = x.frame(Title("Data with distorted Gaussian pdf"),Bins(40)) ;
  data->plotOn(frame1,DataError(RooAbsData::SumW2)) ;
  gauss.plotOn(frame1) ;
```

```
// C a l c u l a t e   c h i ^ 2
// ------------------------------

// Show the chi^2 of the curve w.r.t. the histogram
// If multiple curves or datasets live in the frame you can specify
// the name of the relevant curve and/or dataset in chiSquare()
cout << "chi^2 = " << frame1->chiSquare() << endl ;


// S h o w   r e s i d u a l   a n d   p u l l   d i s t s
// -------------------------------------------------------

// Construct a histogram with the residuals of the data w.r.t. the curve
RooHist* hresid = frame1->residHist() ;

// Construct a histogram with the pulls of the data w.r.t the curve
RooHist* hpull = frame1->pullHist() ;

// Create a new frame to draw the residual distribution and add the distribution to the frame
RooPlot* frame2 = x.frame(Title("Residual Distribution")) ;
frame2->addPlotable(hresid,"P") ;

// Create a new frame to draw the pull distribution and add the distribution to the frame
RooPlot* frame3 = x.frame(Title("Pull Distribution")) ;
frame3->addPlotable(hpull,"P") ;



TCanvas* c = new TCanvas("rf109_chi2residpull","rf109_chi2residpull",900,300) ;
c->Divide(3) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame1->GetYaxis()->SetTitleOffset(1.6) ; frame1->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.6) ; frame2->Draw() ;
c->cd(3) ; gPad->SetLeftMargin(0.15) ; frame3->GetYaxis()->SetTitleOffset(1.6) ; frame3->Draw() ;
}
```

To get the Value from Curve

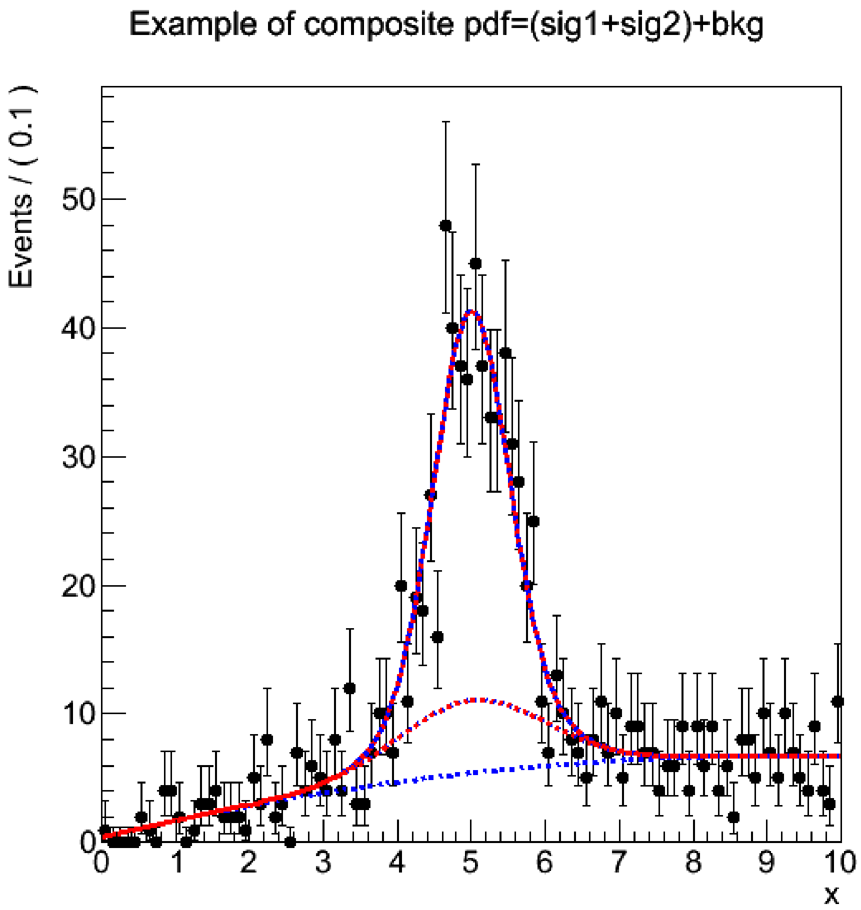
```
RooHist* histo = (RooHist*) frame->findObject("data") ;
RooCurve* func = (RooCurve*) frame->findObject("model") ;
for (Int_t i=0 ; i<histo->GetN() ; i++) {
  Double_t xdata,ydata;
  histo->GetPoint(i,xdata,ydata) ;
  Double_t yfunc = curve->interpolate(xdata) ;
  // Use xdata,ydata,yfunc here
}
```

SetSeed for RooFit Function

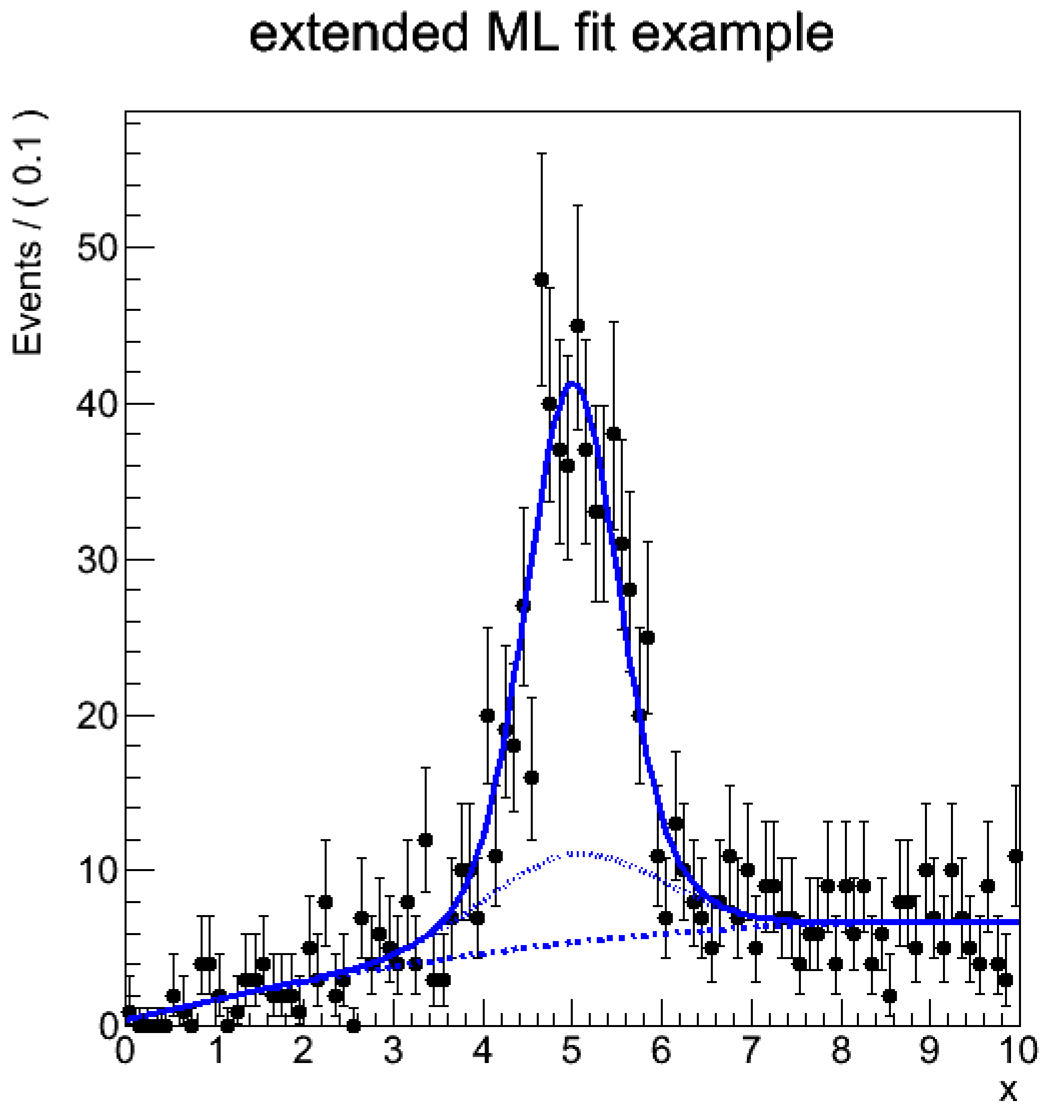
```
#include <RooRandom.h>

Int_t idx=1;

void gen_fitBK(){
  RooRandom::randomGenerator()->SetSeed(idx);
....
```

Useful examples

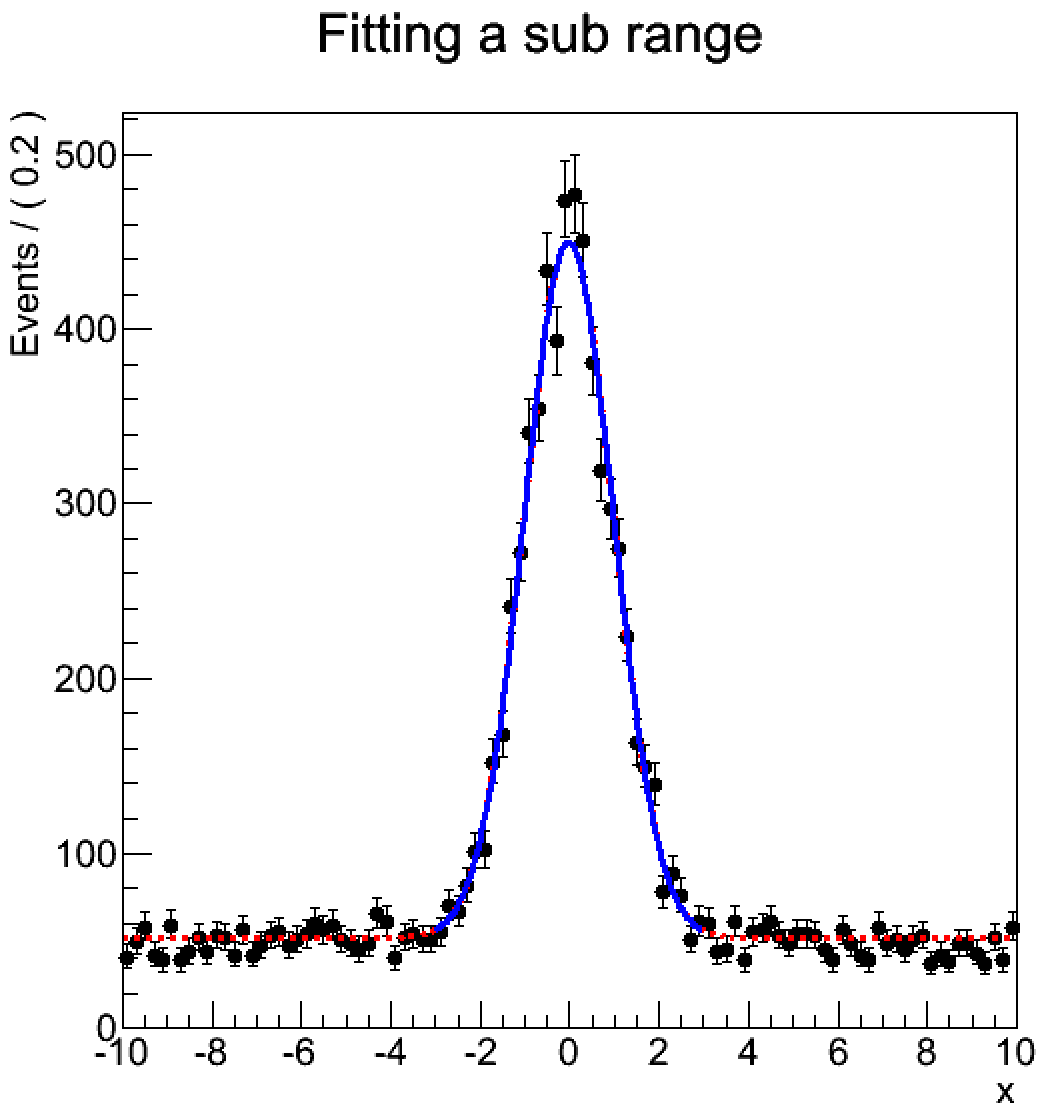
rf201_composite.C



root rf202_extendedmlfit.C



root rf203_ranges.C



siguang@pku.edu.cn

rf205_compplot.C