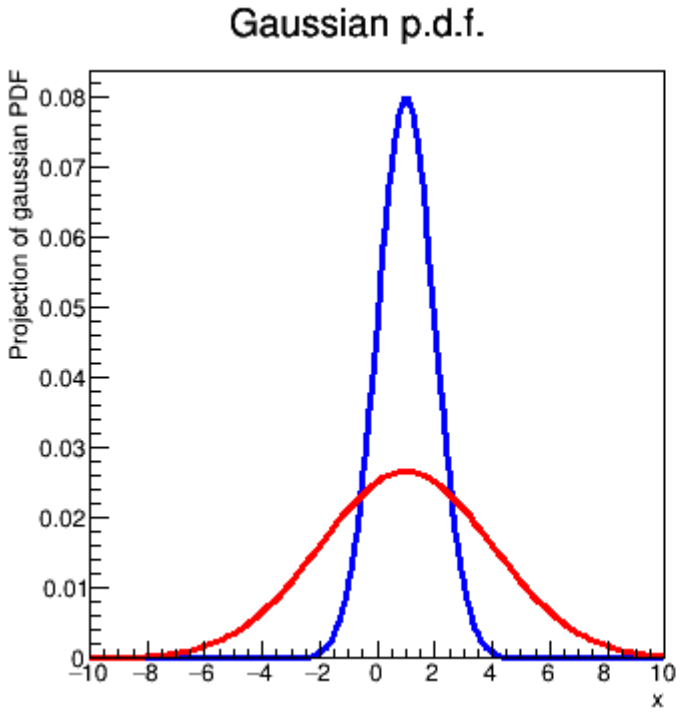
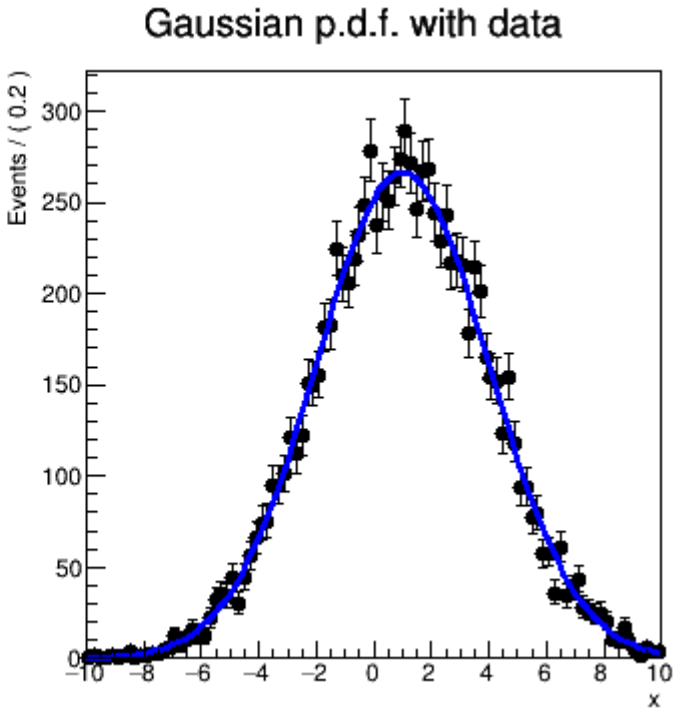
$ROOTSYS/tutorials/roofit/rf101_basics.C





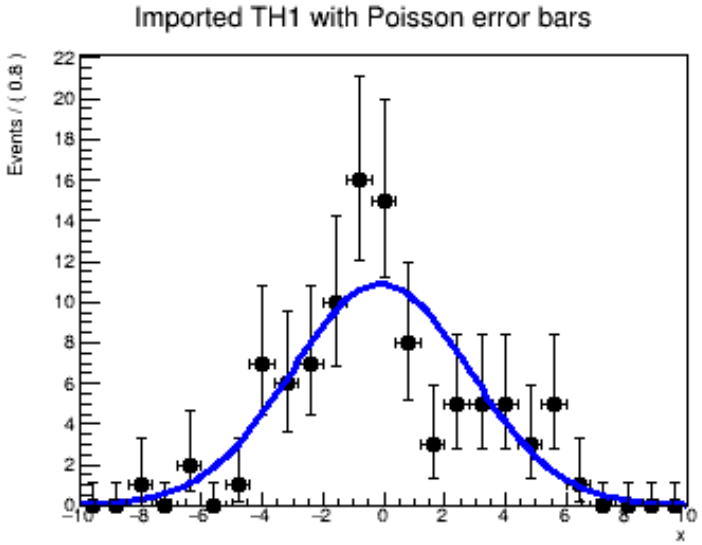
siguang@pku.edu.cn

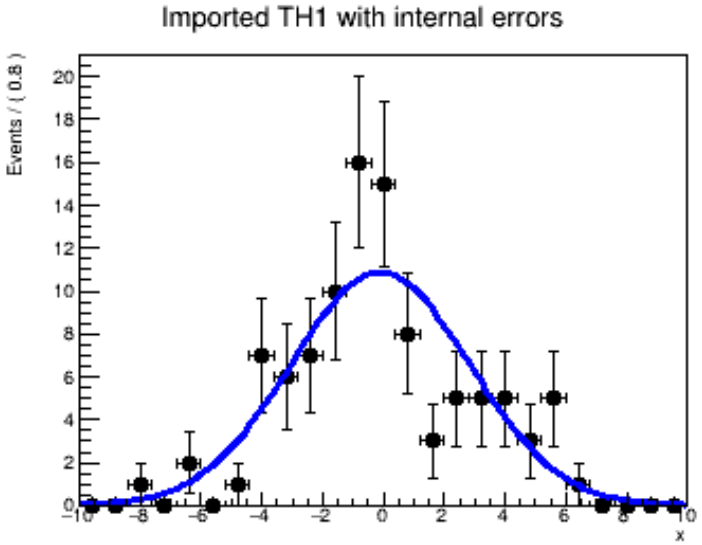
```
#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "TCanvas.h"
#include "RooPlot.h"
#include "TAxis.h"
using namespace RooFit ;


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 all
  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.")) ;
```

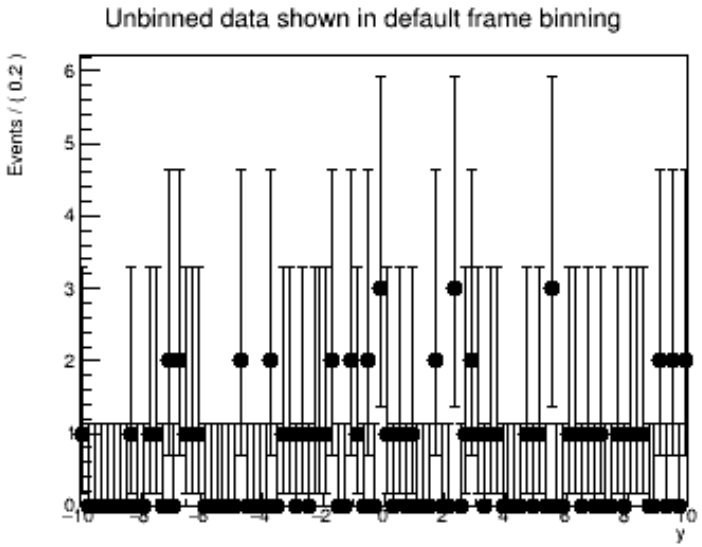
```
// Plot gauss in frame (i.e. in x)
gauss.plotOn(xframe) ;

// 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) ;
gauss.plotOn(xframe2) ;
```

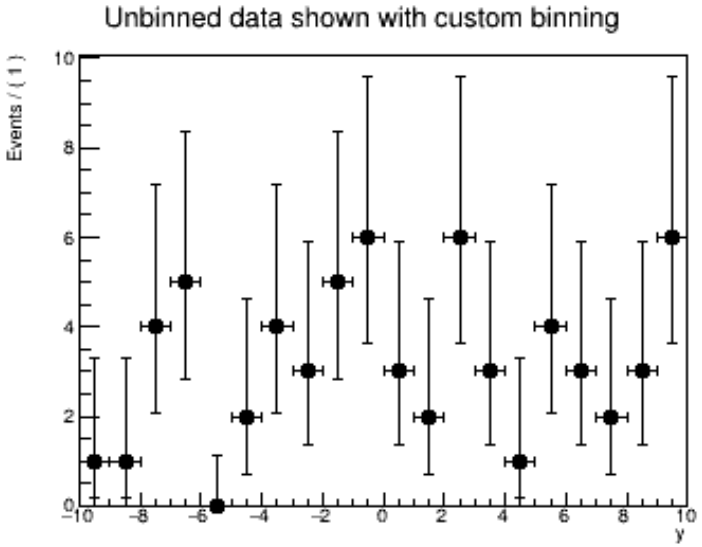

```
  // F i t   m o d e l   t o   d a t a
  // ---------------------------

  // 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() ;


}
```

$ROOTSYS/tutorials/roofit/rf102_dataimport.C









siguang@pku.edu.cn

```
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 correctly
// 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 TTree 'tree'

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

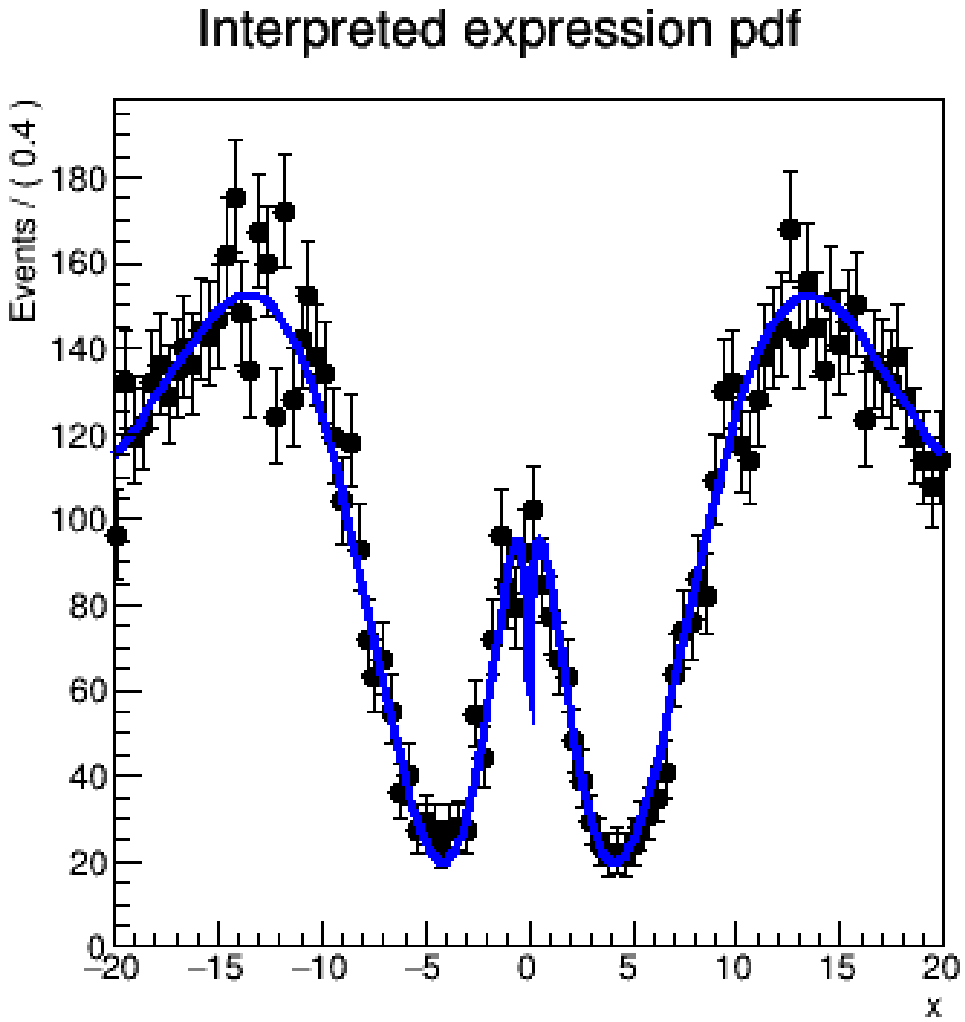
导入数据的时候自动加上x、y区间范围的cut

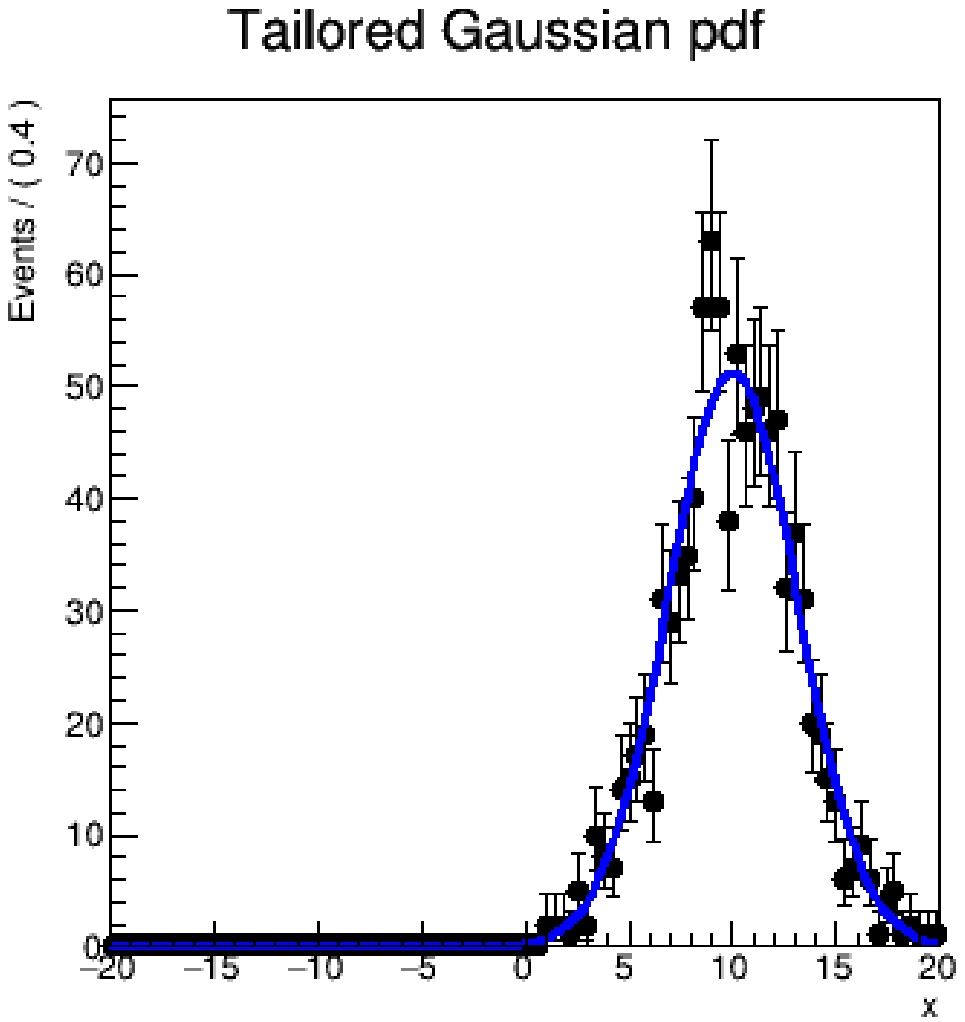
```
// P l o t   d a t a s e t   w i t h   m u l t i p l e   b i n n i n g   c h o i c e s
// ---------------------------------------------------------------------------

// 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 ;
}
```

$ROOTSYS/tutorials/roofit/rf103_interprfuncs.C





siguang@pku.edu.cn

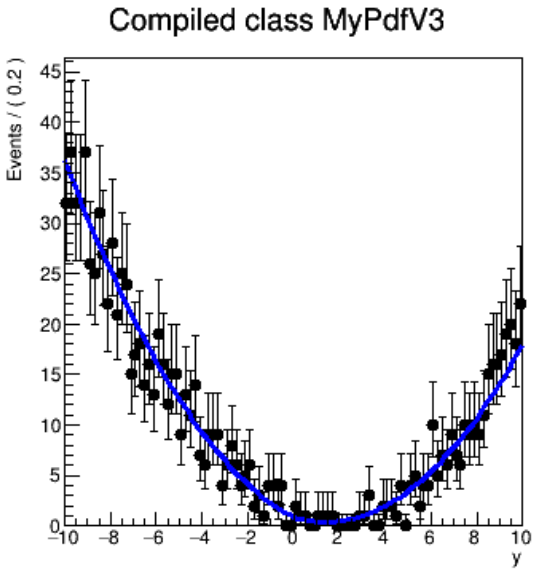
```
void rf103_interprfuncs()
{
  ////////////////////////////////////////////////////
  // G e n e r i c   i n t e r p r e t e d   p . d . f . //
  ////////////////////////////////////////////////////

  // 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)) ;
```

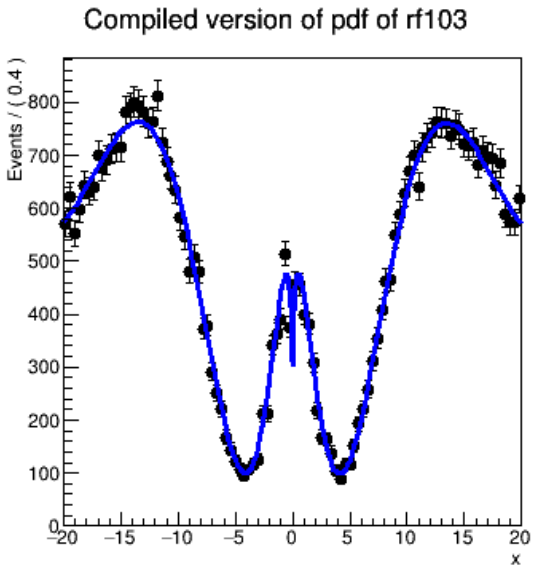
```
// S a m p l e ,   f i t   a n d   p l o t   g e n e r i c   p d f
// ---------------------------------------------------------------

// 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) ;
```

```
//////////////////////////////////////////////////////////////////////////////
// S t a n d a r d   p . d . f   a d j u s t   w i t h   i n t e r p r e t e d   h e l p e r   f u n c t i o n //
//                                                                          //
// Make a gauss(x,sqrt(mean2),sigma) from a standard RooGaussian            //
//                                                                          //
//////////////////////////////////////////////////////////////////////////////


// C o n s t r u c t   s t a n d a r d   p d f   w i t h   f o r m u l a   r e p l a c i n g   p a r a m e t e r
// ---------------------------------------------------------------------------

// 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() ;
```

$ROOTSYS/tutorials/roofit/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
```

Quit and re-start the root

siguang@pku.edu.cn

```
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)") ;
```

```
  // With  added  analytical  integral  expression
  // ---------------------------------------------------------------------

  // 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(rangeName)-B,3)-pow(x.min(ra
ngeName)-B,3))") ;



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

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

是否有解析积分式

是否加产生数据代码的框架

解析积分式

siguang@pku.edu.cn

```
// 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) ;
```

```
//////////////////////////////////////////////////////////////////////
// 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))))",RooArgSet(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() ;

}
```

MyPdfV3.h


```
/*****************************************************************************
 * Project: RooFit                                                           *
 *                                                                           *
  * This code was autogenerated by RooClassFactory                           *
 *****************************************************************************/

#ifndef MYPDFV3
#define MYPDFV3

#include "RooAbsPdf.h"
#include "RooRealProxy.h"
#include "RooCategoryProxy.h"
#include "RooAbsReal.h"
#include "RooAbsCategory.h"

class MyPdfV3 : public RooAbsPdf {
public:
  MyPdfV3() {} ;
  MyPdfV3(const char *name, const char *title,
              RooAbsReal& _x,
              RooAbsReal& _A,
              RooAbsReal&  B);
```


```
   MyPdfV3(const MyPdfV3& other, const char* name=0) ;
   virtual TObject* clone(const char* newname) const { return new MyPdfV3(*this,newname); }
   inline virtual ~MyPdfV3() { }

   Int_t getAnalyticalIntegral(RooArgSet& allVars, RooArgSet& analVars, const char* rangeName=0) const ;
   Double_t analyticalIntegral(Int_t code, const char* rangeName=0) const ;

protected:

   RooRealProxy x ;
   RooRealProxy A ;
   RooRealProxy B ;

   Double_t evaluate() const ;

private:

   ClassDef(MyPdfV3,1) // Your description goes here...
};

#endif
```

MyPdfV3.cxx

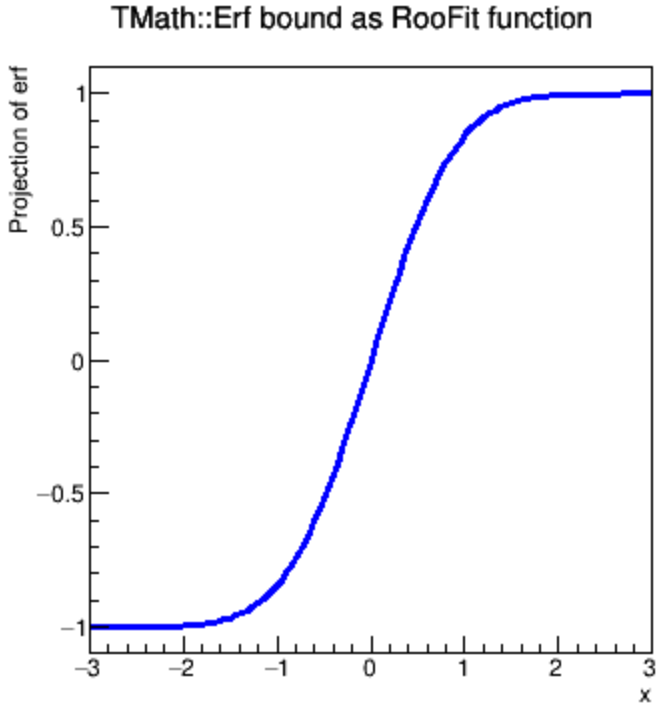
```
#include "Riostream.h"

#include "MyPdfV3.h"
#include "RooAbsReal.h"
#include "RooAbsCategory.h"
#include <math.h>
#include "TMath.h"

ClassImp(MyPdfV3)

 MyPdfV3::MyPdfV3(const char *name, const char *title,
                        RooAbsReal& _x,
                        RooAbsReal& _A,
                        RooAbsReal& _B) :
   RooAbsPdf(name,title),
   x("x","x",this,_x),
   A("A","A",this,_A),
   B("B","B",this,_B)
 {
 }
```

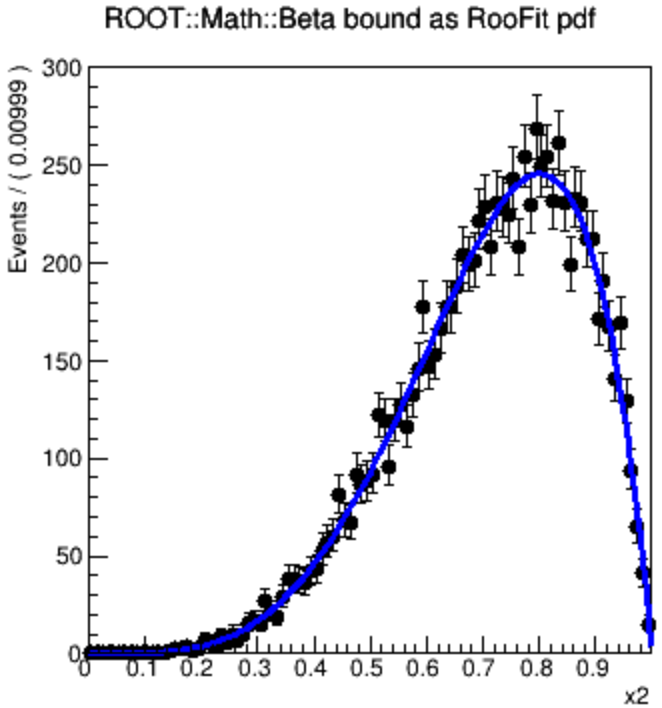
```
MyPdfV3::MyPdfV3(const MyPdfV3& other, const char* name) :
  RooAbsPdf(other,name),
  x("x",this,other.x),
  A("A",this,other.A),
  B("B",this,other.B)
{
}



Double_t MyPdfV3::evaluate() const
{
  // ENTER EXPRESSION IN TERMS OF VARIABLE ARGUMENTS HERE
  return A*fabs(x)+pow(x-B,2) ;
}
```

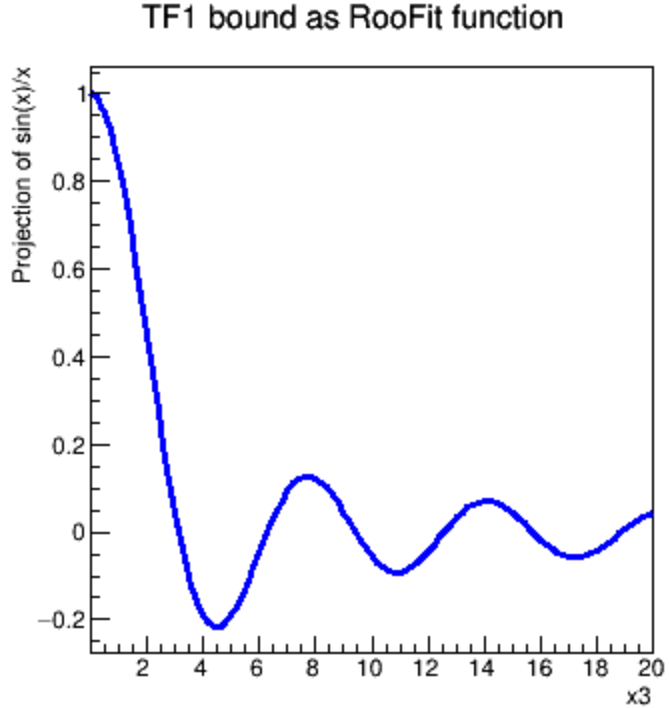
```
Int_t MyPdfV3::getAnalyticalIntegral(RooArgSet& allVars, RooArgSet& analVars, const char* /*rangeName*/) const
{
  // LIST HERE OVER WHICH VARIABLES ANALYTICAL INTEGRATION IS SUPPORTED,
  // ASSIGN A NUMERIC CODE FOR EACH SUPPORTED (SET OF) PARAMETERS
  // THE EXAMPLE BELOW ASSIGNS CODE 1 TO INTEGRATION OVER VARIABLE X
  // YOU CAN ALSO IMPLEMENT MORE THAN ONE ANALYTICAL INTEGRAL BY REPEATING THE matchArgs
  // EXPRESSION MULTIPLE TIMES

  if (matchArgs(allVars,analVars,x)) return 1 ;
  return 0 ;
}


Double_t MyPdfV3::analyticalIntegral(Int_t code, const char* rangeName) const
{
  // RETURN ANALYTICAL INTEGRAL DEFINED BY RETURN CODE ASSIGNED BY getAnalyticalIntegral
  // THE MEMBER FUNCTION x.min(rangeName) AND x.max(rangeName) WILL RETURN THE INTEGRATION
  // BOUNDARIES FOR EACH OBSERVABLE x

  if (code==1) { return ((A/2)*(pow(x.max(rangeName),2)+pow(x.min(rangeName),2))+(1./3)*(pow(x.max(rangeName)-B,3)-pow(x.min(rangeN
ame)-B,3))) ; }
  return 0 ;
}
```

有些表达式可以借用Mathematica等软件给出积分形式，保存成C代码，作为子程序被调用给出解析积分计算结果。

$ROOTSYS/tutorials/roofit/rf105_funcbinding.C







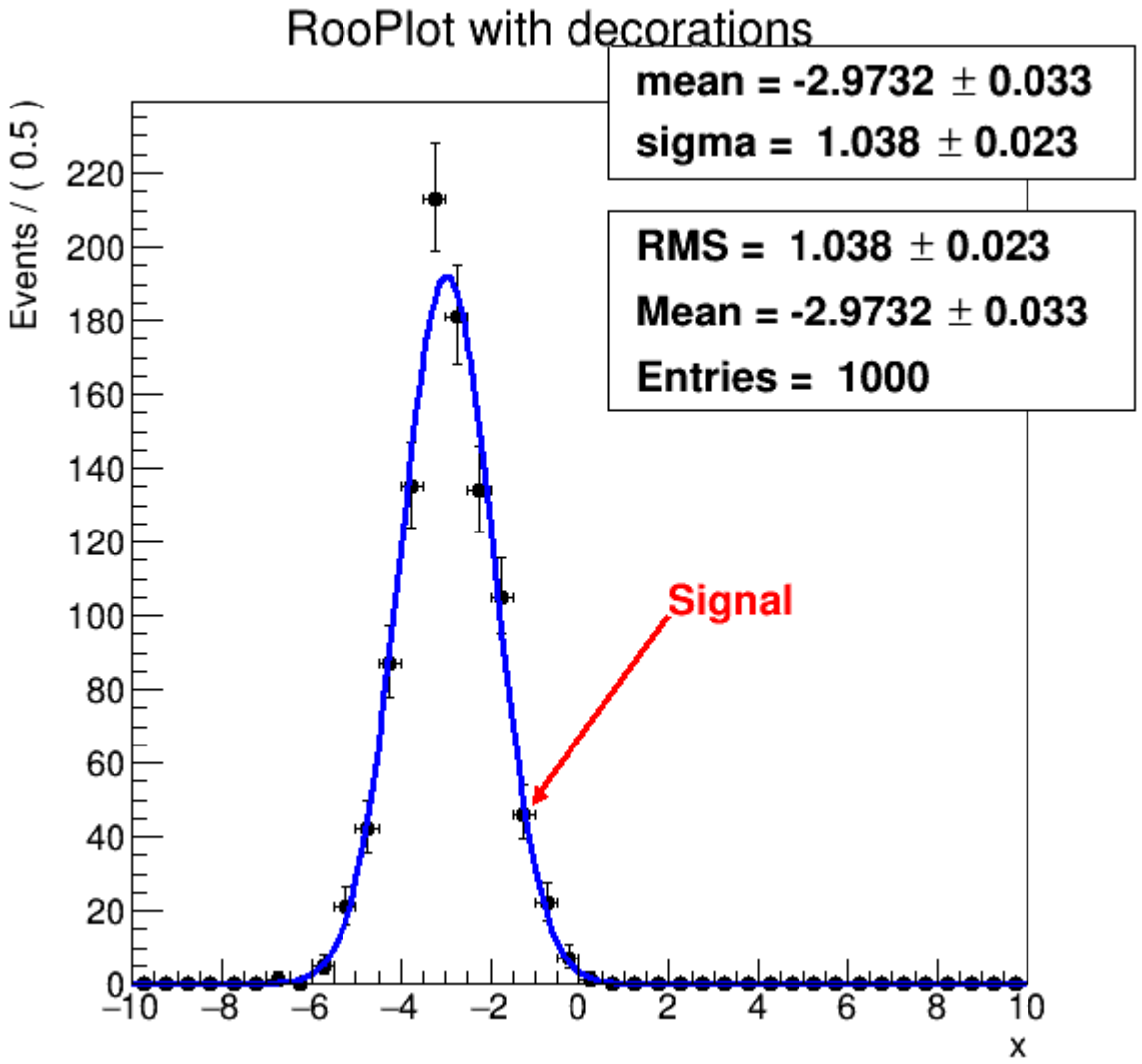
siguang@pku.edu.cn

```
void rf105_funcbinding()
{

  // B i n d   T M a t h : : E r f   C   f u n c t i o n
  // ---------------------------------------------------

  // Bind one-dimensional TMath::Erf function as RooAbsReal function
  RooRealVar x("x","x",-3,3) ;
  RooAbsReal* erf = bindFunction("erf",TMath::Erf,x) ;

  // Print erf definition
  erf->Print() ;

  // Plot erf on frame
  RooPlot* frame1 = x.frame(Title("TMath::Erf bound as RooFit function")) ;
  erf->plotOn(frame1) ;
```


```
// Bind  ROOT::Math::beta_pdf  C  function
// ---------------------------------------------------------------

// Bind pdf ROOT::Math::Beta with three variables as RooAbsPdf function
RooRealVar x2("x2","x2",0,0.999) ;
RooRealVar a("a","a",5,0,10) ;
RooRealVar b("b","b",2,0,10) ;
RooAbsPdf* beta = bindPdf("beta",ROOT::Math::beta_pdf,x2,a,b) ;

// Perf beta definition
beta->Print() ;

// Generate some events and fit
RooDataSet* data = beta->generate(x2,10000) ;
beta->fitTo(*data) ;

// Plot data and pdf on frame
RooPlot* frame2 = x2.frame(Title("ROOT::Math::Beta bound as RooFit pdf")) ;
data->plotOn(frame2) ;
beta->plotOn(frame2) ;
```

```
// Bind ROOT TF1 as RooFit function
// -------------------------------------------------------

// Create a ROOT TF1 function
TF1 *fa1 = new TF1("fa1","sin(x)/x",0,10);

// Create an observable
RooRealVar x3("x3","x3",0.01,20) ;

// Create binding of TF1 object to above observable
RooAbsReal* rfa1 = bindFunction(fa1,x3) ;

// Print rfa1 definition
rfa1->Print() ;

// Make plot frame in observable, plot TF1 binding function
RooPlot* frame3 = x3.frame(Title("TF1 bound as RooFit function")) ;
rfa1->plotOn(frame3) ;

TCanvas* c = new TCanvas("rf105_funcbinding","rf105_funcbinding",1200,400) ;
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() ;
```

$ROOTSYS/tutorials/roofit/rf106_plotdecoration.C



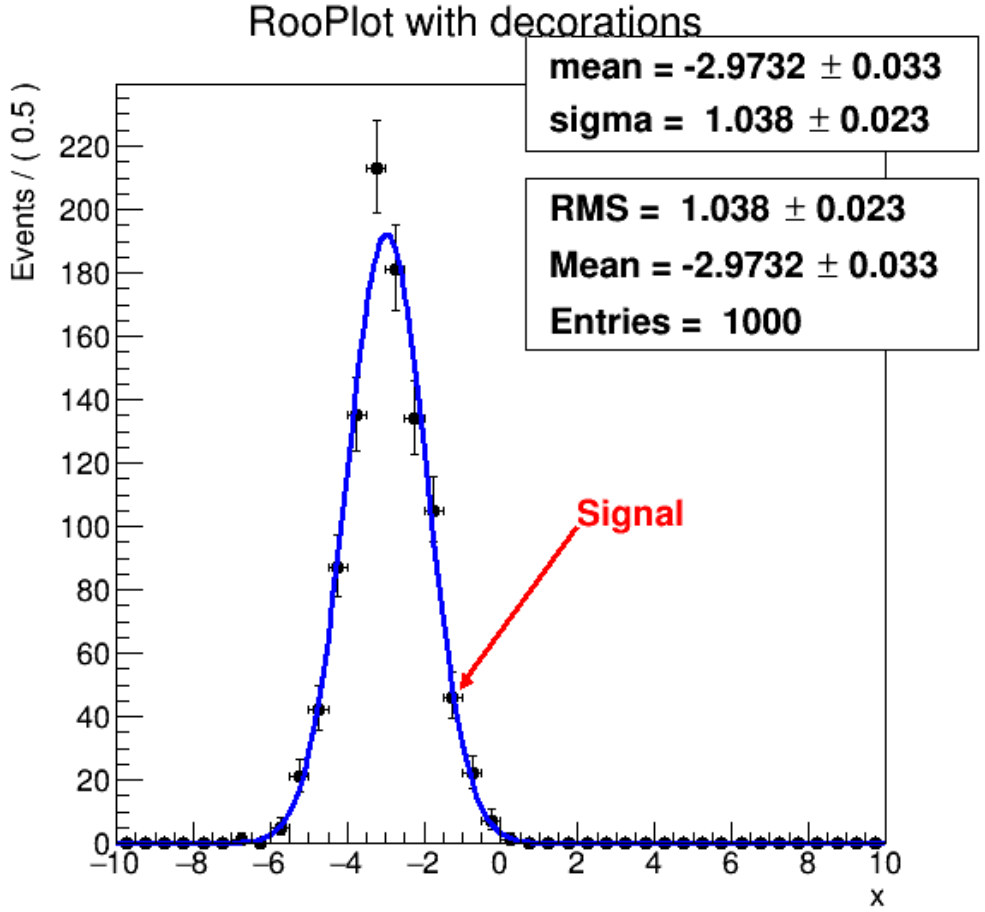
siguang@pku.edu.cn

```
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) ;


  // 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)) ;
```

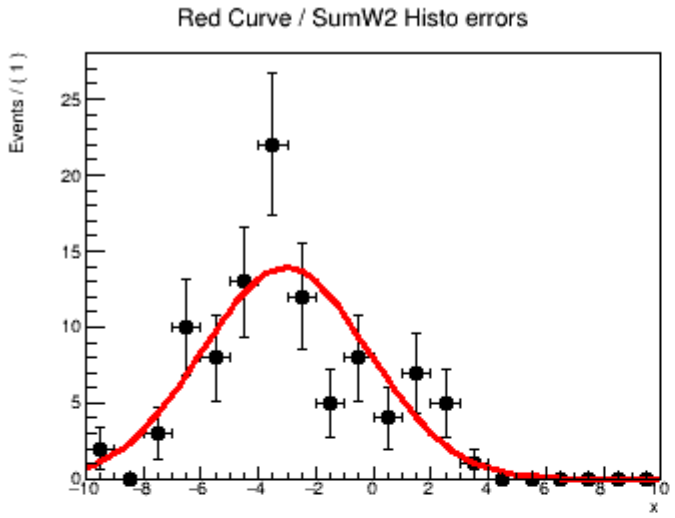
rf102_dataimport.C:
ds.plotOn(frame4,**Binning(20)**) ;

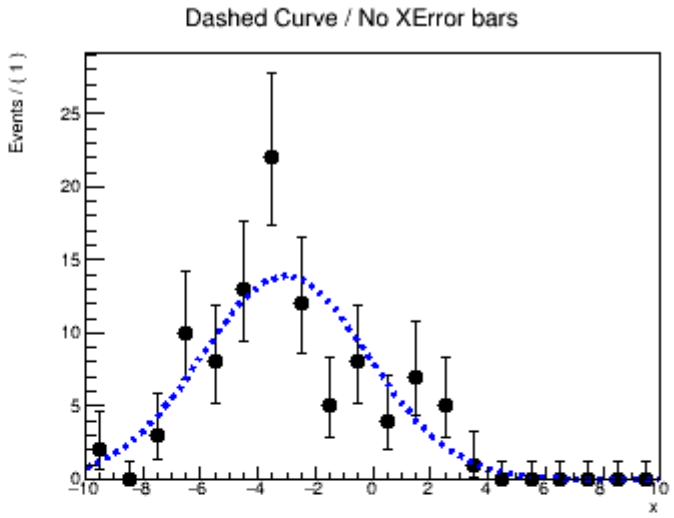
```
// A d d   b o x   w i t h   d a t a   s t a t i s t i c s
// ------------------------------------------------------

// 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) ;
```

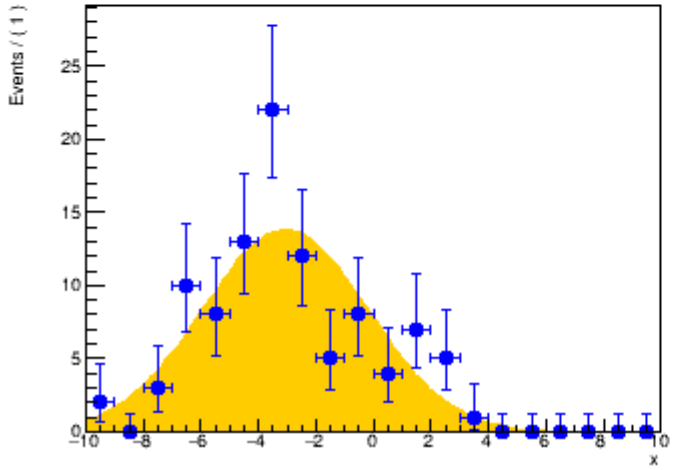


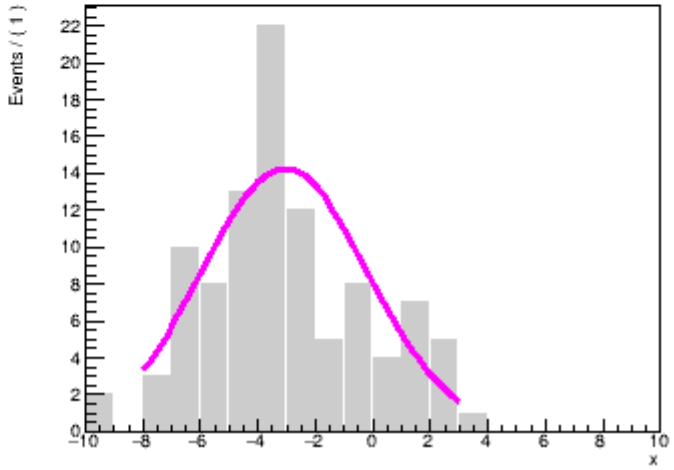

```
   // Persist frame with all decorations in ROOT file
   // ---------------------------------------------------------------------------

   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() ;

}
```

$ROOTSYS/tutorials/roofit/rf107_plotstyles.C









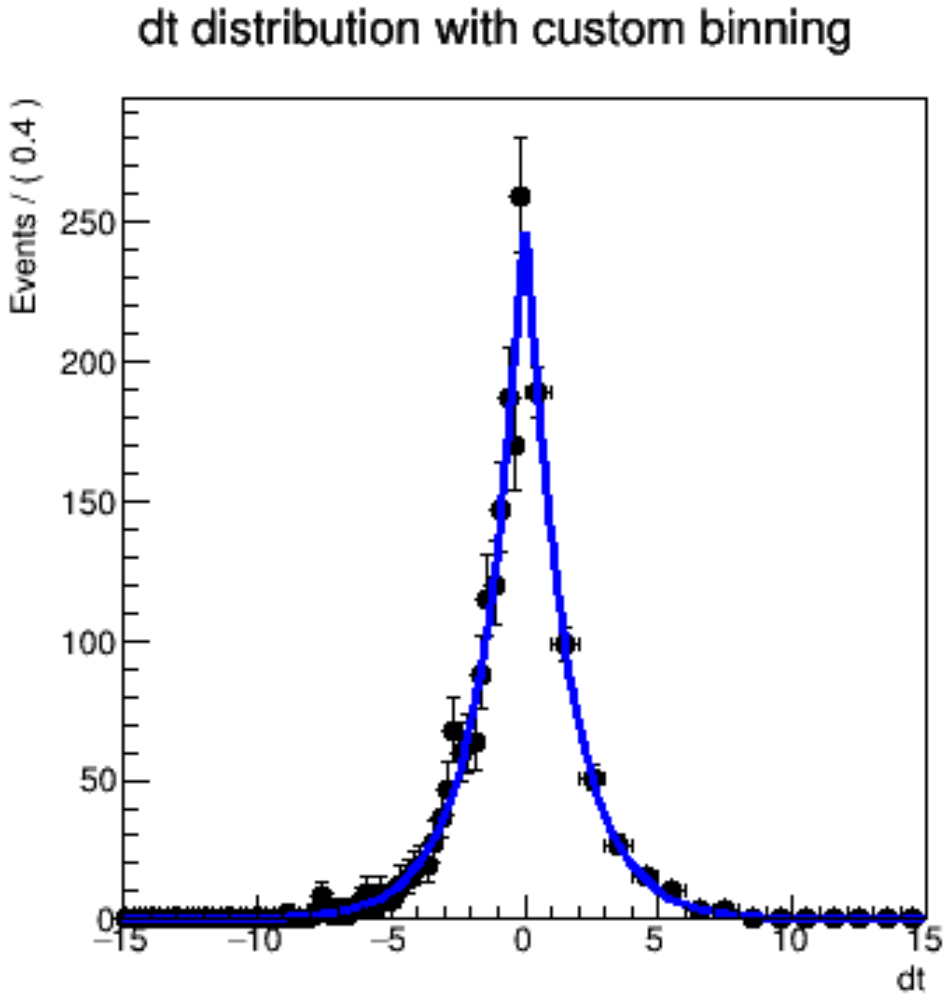
siguang@pku.edu.cn

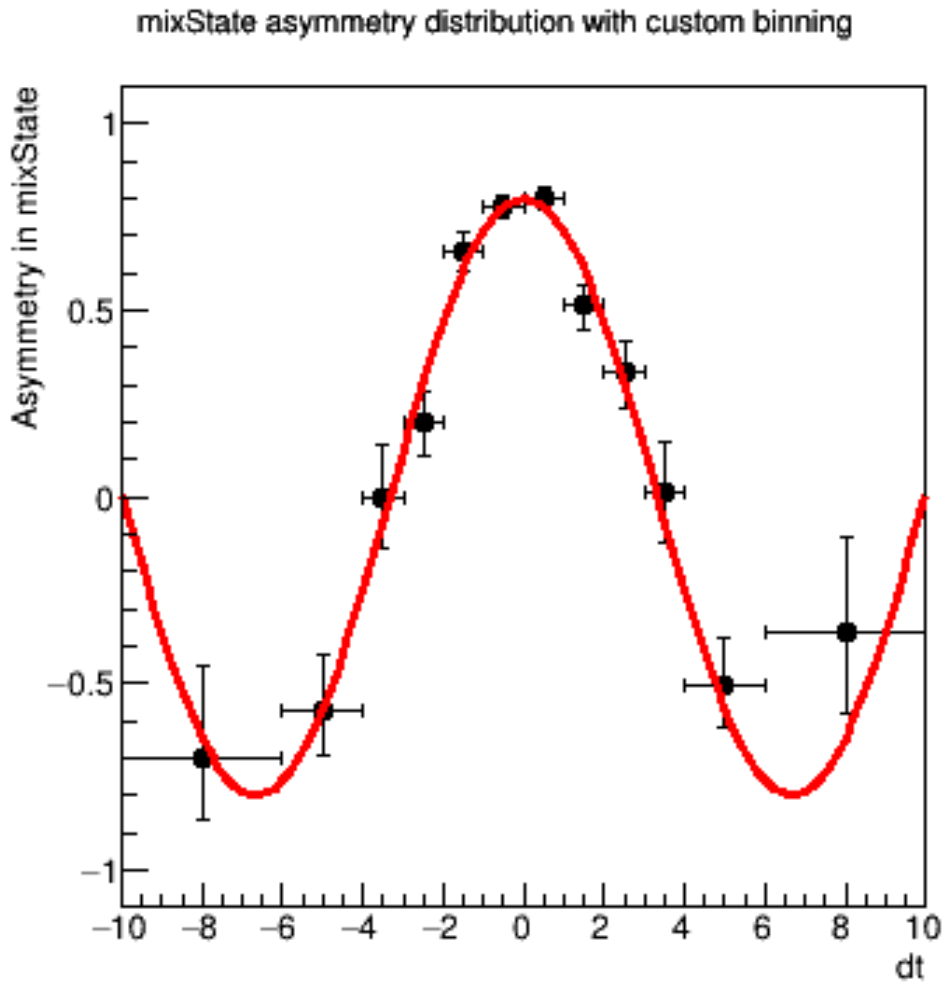
```
void rf107_plotstyles()
{

  // 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) ;
```

```
// 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)) ;
```

```
// 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() ;
```

$ROOTSYS/tutorials/roofit/rf108_plotbinning.C





（-15，0）共60个bin

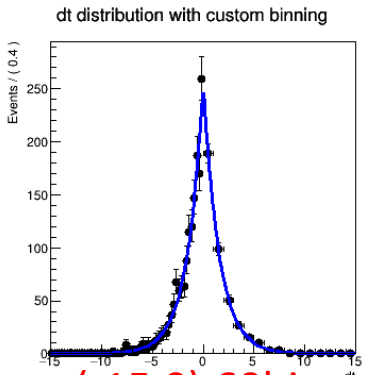
（0，15）共15个bin

siguang@pku.edu.cn

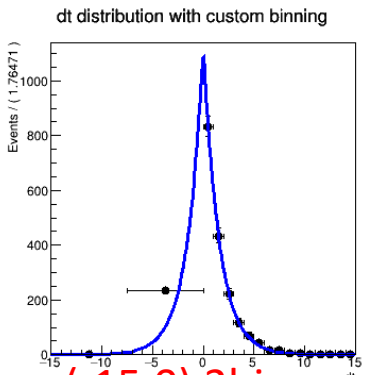
```
void rf108_plotbinning()
{

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

  // Build a B decay p.d.f with mixing
  RooRealVar dt("dt","dt",-20,20) ;
  RooRealVar dm("dm","dm",0.472) ;
  RooRealVar tau("tau","tau",1.547) ;
  RooRealVar w("w","mistag rate",0.1) ;
  RooRealVar dw("dw","delta mistag rate",0.) ;

  RooCategory mixState("mixState","B0/B0bar mixing state") ;
  mixState.defineType("mixed",-1) ;
  mixState.defineType("unmixed",1) ;
  RooCategory tagFlav("tagFlav","Flavour of the tagged B0") ;
  tagFlav.defineType("B0",1) ;
  tagFlav.defineType("B0bar",-1) ;
```

```
// Build a gaussian resolution model
RooRealVar dterr("dterr","dterr",0.1,1.0) ;
RooRealVar bias1("bias1","bias1",0) ;
RooRealVar sigma1("sigma1","sigma1",0.1) ;
RooGaussModel gm1("gm1","gauss model 1",dt,bias1,sigma1) ;

// Construct Bdecay (x) gauss
RooBMixDecay bmix("bmix","decay",dt,mixState,tagFlav,tau,dm,w,dw,gm1,RooBMixDecay::DoubleSided) ;


// S a m p l e   d a t a   f r o m   m o d e l
// ----------------------------------------

// Sample 2000 events in (dt,mixState,tagFlav) from bmix
RooDataSet *data = bmix.generate(RooArgSet(dt,mixState,tagFlav),2000) ;
```

```
// Make plot of dt distribution of data in range (-15,15) with fine binning for dt>0 and coarse binning for dt<0

// Create binning object with range (-15,15)
RooBinning tbins(-15,15) ;

// Add 60 bins with uniform spacing in range (-15,0)
tbins.addUniform(60,-15,0) ;

// Add 15 bins with uniform spacing in range (0,15)
tbins.addUniform(15,0,15) ;

// Make plot with specified binning
RooPlot* dtframe = dt.frame(Range(-15,15),Title("dt distribution with custom binning")) ;
data->plotOn(dtframe,Binning(tbins)) ;
bmix.plotOn(dtframe) ;

// NB: Note that bin density for each bin is adjusted to that of default frame binning as shown
// in Y axis label (100 bins --> Events/0.4*Xaxis-dim) so that all bins represent a consistent density distribution
```



(-15,0) 60bins

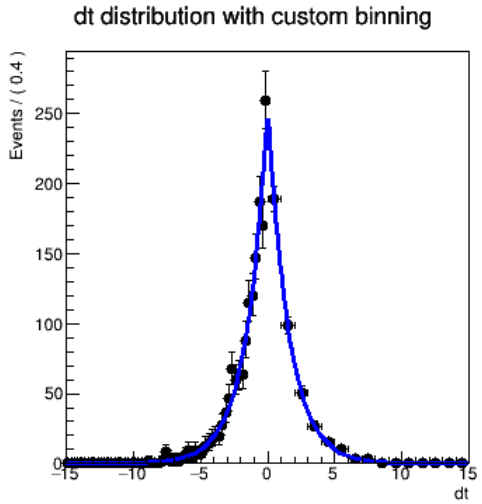


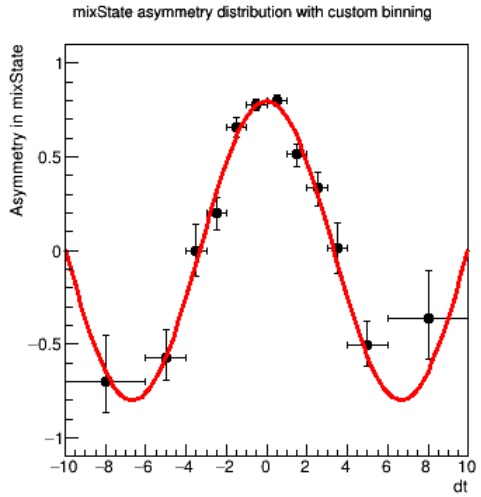
(-15,0) 2bins

From -15 to 15 total 30 and the number of bins=60+15=75
So 30/75. = 0.4, so Y axis title as Events/(0.4)

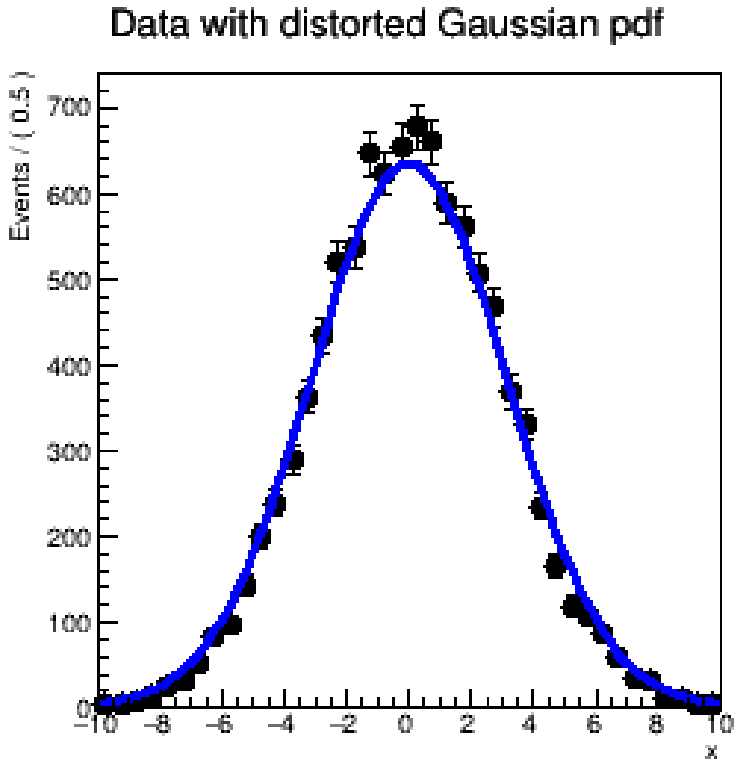
```
// Make plot of dt distribution of data asymmetry in 'mixState' with variable binning

// Create binning object with range (-10,10)
RooBinning abins(-10,10) ;

// Add boundaries at 0, (-1,1), (-2,2), (-3,3), (-4,4) and (-6,6)
abins.addBoundary(0) ;
abins.addBoundaryPair(1) ;
abins.addBoundaryPair(2) ;
abins.addBoundaryPair(3) ;
abins.addBoundaryPair(4) ;
abins.addBoundaryPair(6) ;

// Create plot frame in dt
RooPlot* aframe = dt.frame(Range(-10,10),Title("mixState asymmetry distribution with custom binning")) ;

// Plot mixState asymmetry of data with specified customg binning
data->plotOn(aframe,Asymmetry(mixState),Binning(abins)) ;

// Plot corresponding property of p.d.f
bmix.plotOn(aframe,Asymmetry(mixState)) ;
```

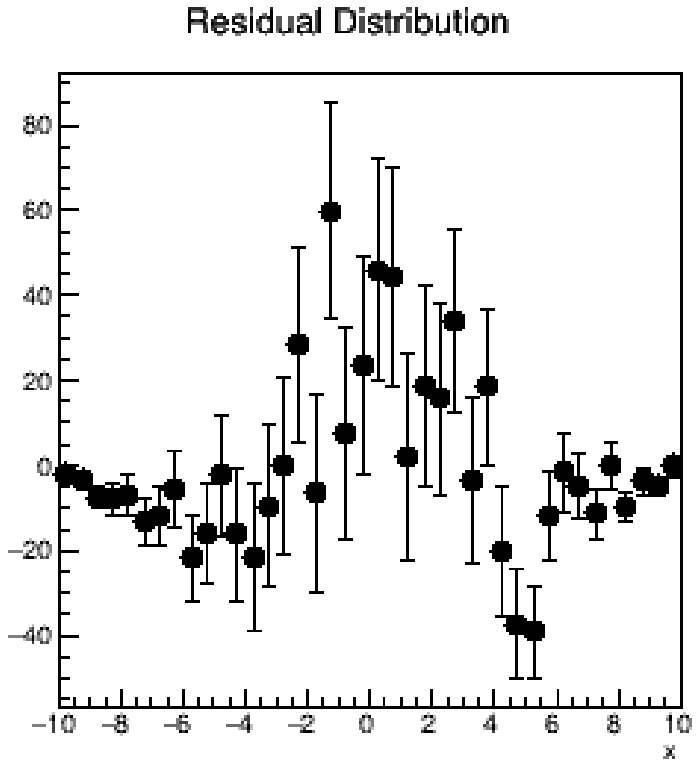
```
// Adjust vertical range of plot to sensible values for an asymmetry
aframe->SetMinimum(-1.1) ;
aframe->SetMaximum(1.1) ;

// NB: For asymmetry distributions no density corrects are needed (and are thus not applied)


// Draw plots on canvas
TCanvas* c = new TCanvas("rf108_plotbinning","rf108_plotbinning",800,400) ;
c->Divide(2) ;
c->cd(1) ; gPad->SetLeftMargin(0.15) ; dtframe->GetYaxis()->SetTitleOffset(1.6) ; dtframe->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; aframe->GetYaxis()->SetTitleOffset(1.6) ; aframe->Draw() ;
}
```

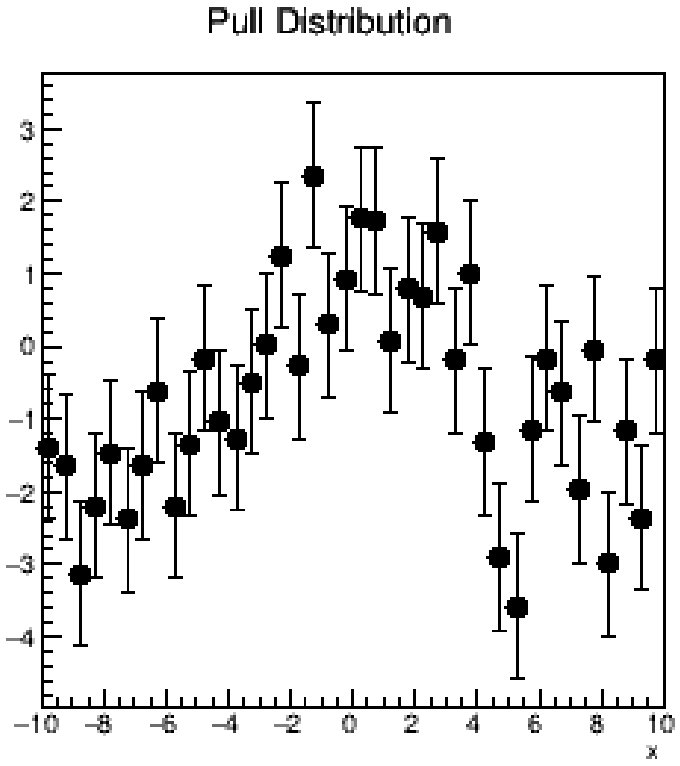




$ROOTSYS/tutorials/roofit/rf109_chi2residpull.C







siguang@pku.edu.cn

```
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 ;
```

```
// 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 ;
```

```
Double_t chiSquare (int nFitParam = 0) const
Double_t chiSquare (const char* pdfname, const char* histname, int nFitParam = 0) const
```

```
Double_t chiSquare (const char* pdfname, const char* histname, int nFitParam = 0) const

Calculate and return reduced chi-squared of curve with given name with respect
to histogram with given name. If nFitParam is non-zero, it is used to reduce the
number of degrees of freedom for a chi^2 for a curve that was fitted to the
data with that number of floating parameters
```


```
// 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() ;

}
```

$ROOTSYS/tutorials/roofit/rf110_normintegration.C

```
void rf110_normintegration()
{
  // S e t u p   m o d e l
  // ---------------------

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

  // Create p.d.f. gaussx(x,-2,3)
  RooGaussian gx("gx","gx",x,RooConst(-2),RooConst(3)) ;


  // R e t r i e v e   r a w &  n o r m a l i z e d  v a l u e s  o f  R o o F i t  p . d . f . s
  // ------------------------------------------------------------------------------------------

  // Return 'raw' unnormalized value of gx
  cout << "gx = " << gx.getVal() << endl ;
```

```
gx = 0.800737
```

X=0处（x的初始值）没有归一的gaus函数的取值。

siguang@pku.edu.cn

```
// Return value of gx normalized over x in range [-10,10]
RooArgSet nset(x) ;
cout << "gx_Norm[x] = " << gx.getVal(&nset) << endl ;
```

```
gx_Norm[x] = 0.106896
```

X=0处（x的初始值）归一化后的gaus函数的取值。

```
// Create object representing integral over gx
// which is used to calculate  gx_Norm[x] == gx / gx_Int[x]
RooAbsReal* igx = gx.createIntegral(x) ;
cout << "gx_Int[x] = " << igx->getVal() << endl ;
```

```
gx_Int[x] = 7.49084
```

整个区间（-10，10）的积分

```
// Integrate  normalized  pdf  over  subrange
// ---------------------------------------------------------


// Define a range named "signal" in x from -5,5
x.setRange("signal",-5,5) ;

// Create an integral of gx_Norm[x] over x in range "signal"
// This is the fraction of of p.d.f. gx_Norm[x] which is in the
// range named "signal"
RooAbsReal* igx_sig = gx.createIntegral(x,NormSet(x),Range("signal")) ;
cout << "gx_Int[x|signal]_Norm[x] = " << igx_sig->getVal() << endl ;
```

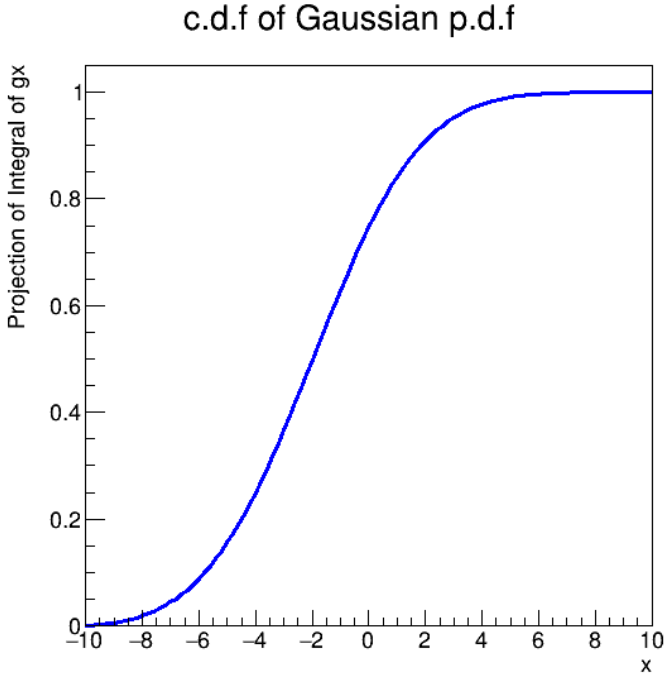
```
gx_Int[x|signal]_Norm[x] = 0.834753
```

（-5，5）区间的积分除以（-10，10）区间的积分

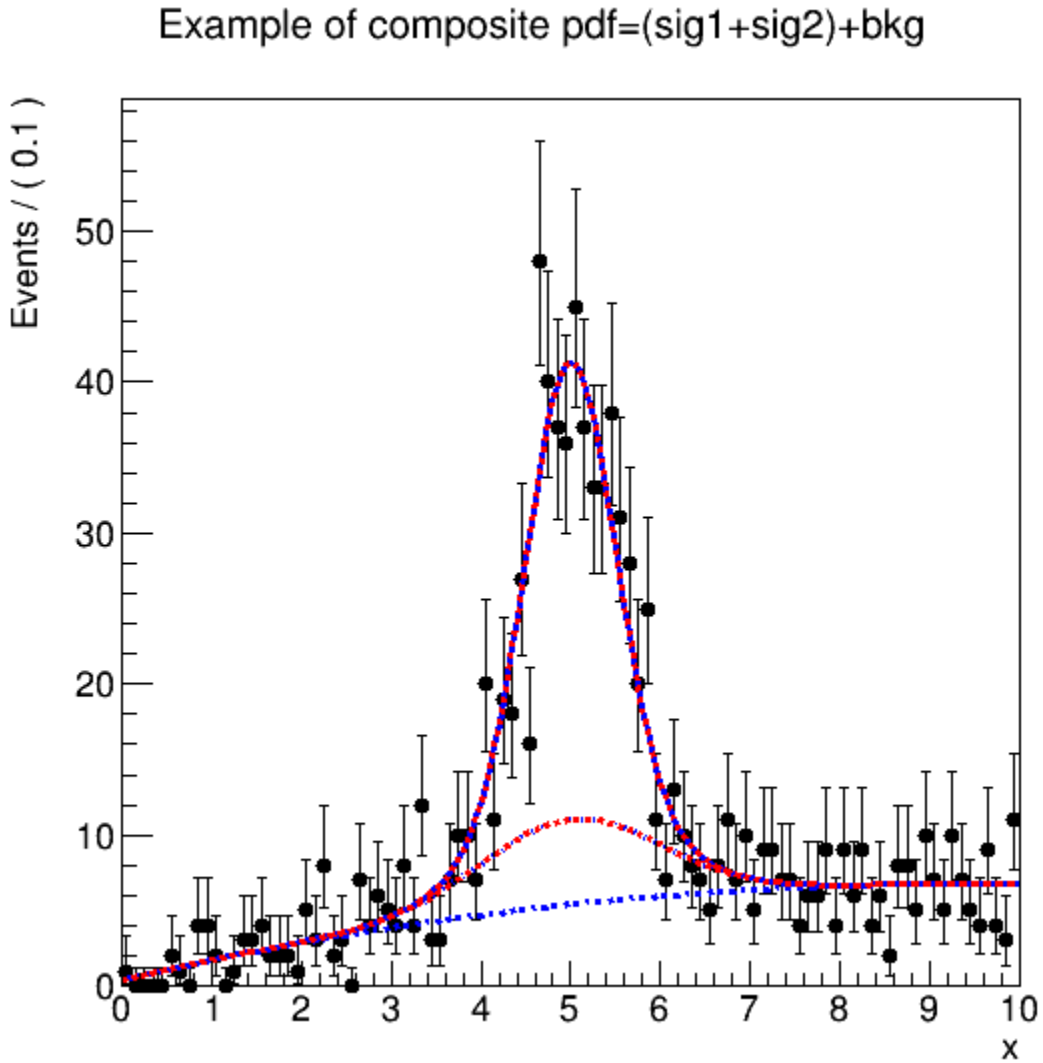
siguang@pku.edu.cn

```
// Construct cumulative distribution function from pdf
// -----------------------------------------------------------------------

// Create the cumulative distribution function of gx
// i.e. calculate Int[-10,x] gx(x') dx'
RooAbsReal* gx_cdf = gx.createCdf(x) ;

// Plot cdf of gx versus x
RooPlot* frame = x.frame(Title("c.d.f of Gaussian p.d.f")) ;
gx_cdf->plotOn(frame) ;



// Draw plot on canvas
new TCanvas("rf110_normintegration","rf110_normintegration",600,600) ;
gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.6) ;
frame->Draw() ;
```

从左到当前x的积分除以整个区间的积分



$ROOTSYS/tutorials/roofit/rf201_composite.C



siguang@pku.edu.cn

```
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 their parameters
  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.) ;
  RooRealVar a1("a1","a1",-0.2,0.,1.) ;
  RooChebychev bkg("bkg","Background",x,RooArgSet(a0,a1)) ;
```

```
//////////////////////////////////////////////////
// M E T H O D  1 - T w o  R o o A d d P d f s  //
//////////////////////////////////////////////////


// A d d  s i g n a l  c o m p o n e n t s
// ---------------------------------------

// Sum the signal components into a composite signal p.d.f.
RooRealVar sig1frac("sig1frac","fraction of component 1 in signal",0.8,0.,1.) ;
RooAddPdf sig("sig","Signal",RooArgList(sig1,sig2),sig1frac) ;
```

[f_sig1 * sig1(x,m,s1) + (1-f_sig1) * sig2(x,m,s2)]

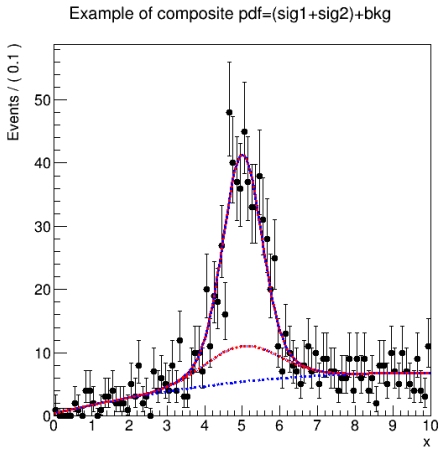
```
// A d d  s i g n a l  a n d  b a c k g r o u n d
// ----------------------------------------------

// Sum the composite signal and background
RooRealVar bkgfrac("bkgfrac","fraction of background",0.5,0.,1.) ;
RooAddPdf  model("model","g1+g2+a",RooArgList(bkg,sig),bkgfrac) ;
```

// pdf = f_bkg * bkg(x,a0,a1) + (1-fbkg) * [f_sig1 * sig1(x,m,s1) + (1-f_sig1) * sig2(x,m,s2)]

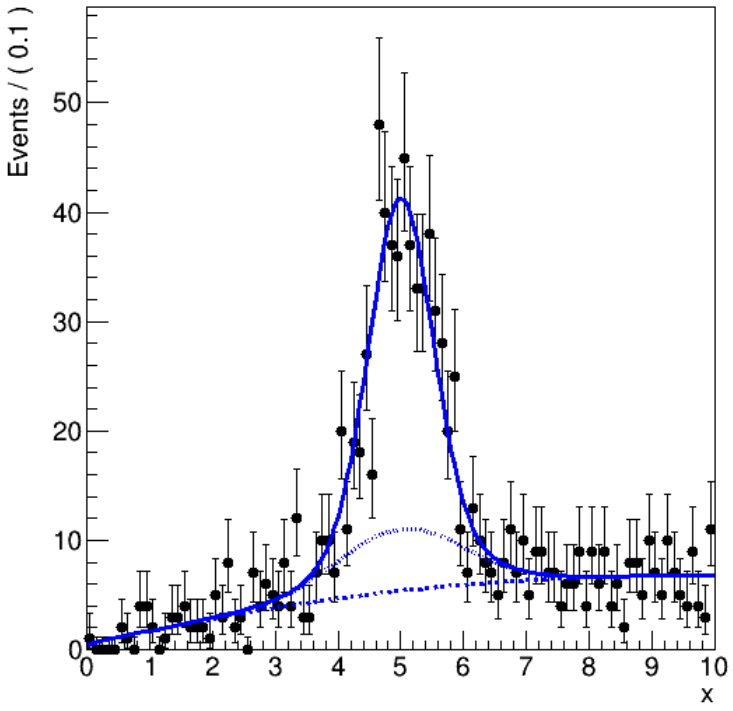
siguang@pku.edu.cn

```
// S a m p l e ,   f i t   a n d   p l o t   m o d e l
// ---------------------------------------------------

// Generate a data sample of 1000 events in x from model
RooDataSet *data = model.generate(x,1000) ;

// Fit model to data
model.fitTo(*data) ;

// Plot data and PDF overlaid
RooPlot* xframe = x.frame(Title("Example of composite pdf=(sig1+sig2)+bkg")) ;
data->plotOn(xframe) ;
model.plotOn(xframe) ;

// Overlay the background component of model with a dashed line
model.plotOn(xframe,Components(bkg),LineStyle(kDashed)) ;

// Overlay the background+sig2 components of model with a dotted line
model.plotOn(xframe,Components(RooArgSet(bkg,sig2)),LineStyle(kDotted)) ;

// Print structure of composite p.d.f.
model.Print("t") ;
```

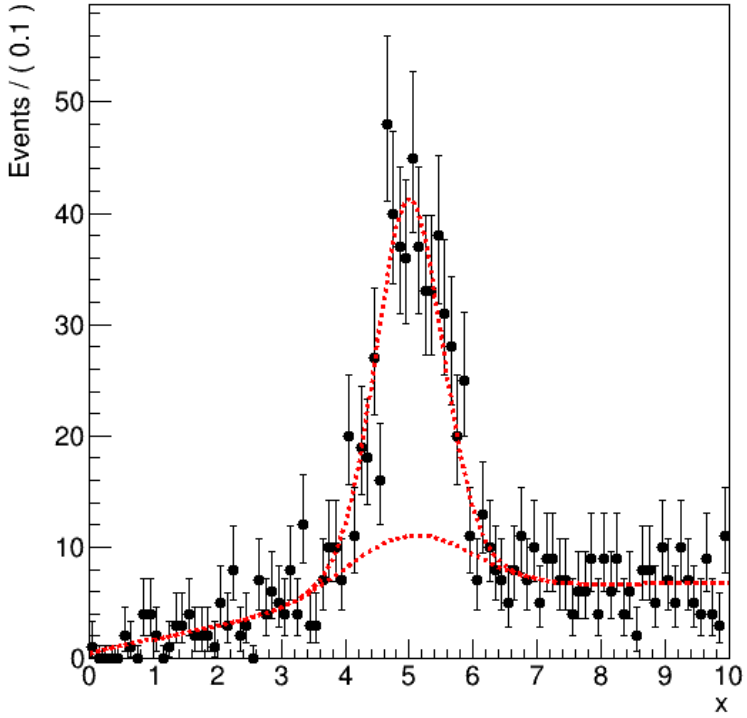
```
////////////////////////////////////////////////////////////////////////////////
// M E T H O D  2 - O n e  R o o A d d P d f  w i t h  r e c u r s i v e  f r a c t i o n s //
////////////////////////////////////////////////////////////////////////////////

// Construct sum of models on one go using recursive fraction interpretations
//
//   model2 = bkg + (sig1 + sig2)
//
RooAddPdf  model2("model","g1+g2+a",RooArgList(bkg,sig1,sig2),RooArgList(bkgfrac,sig1frac),kTRUE) ;

// NB: Each coefficient is interpreted as the fraction of the
// left-hand component of the i-th recursive sum, i.e.
//
//   sum4 = A + ( B + ( C + D))  with fraction fA, fB and fC expands to
//
//   sum4 = fA*A + (1-fA)*(fB*B + (1-fB)*(fC*C + (1-fC)*D))


// P l o t  r e c u r s i v e  a d d i t i o n  m o d e l
// ------------------------------------------------------
model2.plotOn(xframe,LineColor(kRed),LineStyle(kDashed)) ;
model2.plotOn(xframe,Components(RooArgSet(bkg,sig2)),LineColor(kRed),LineStyle(kDashed)) ;
model2.Print("t") ;

// Draw the frame on the canvas
new TCanvas("rf201_composite","rf201_composite",600,600) ;
gPad->SetLeftMargin(0.15) ; xframe->GetYaxis()->SetTitleOffset(1.4) ; xframe->Draw() ;
```



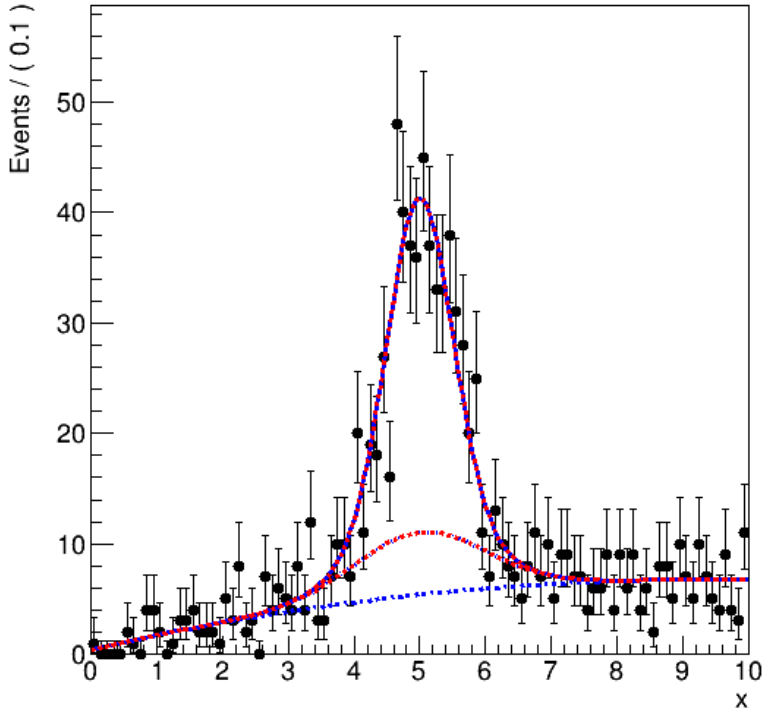
siguang@pku.edu.cn



单画model

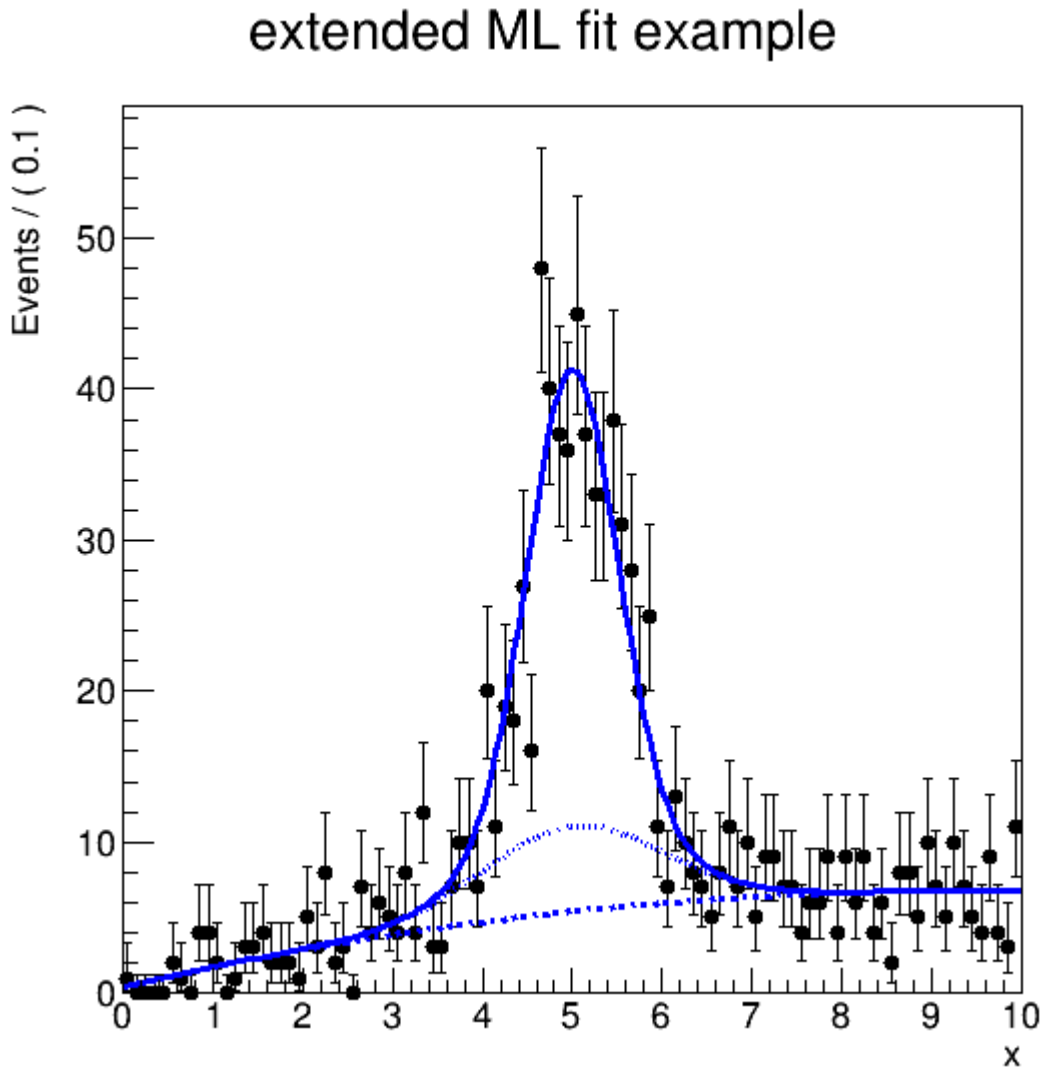


单画model2



model+model2

$ROOTSYS/tutorials/roofit/rf202_extendedmlfit.C



siguang@pku.edu.cn

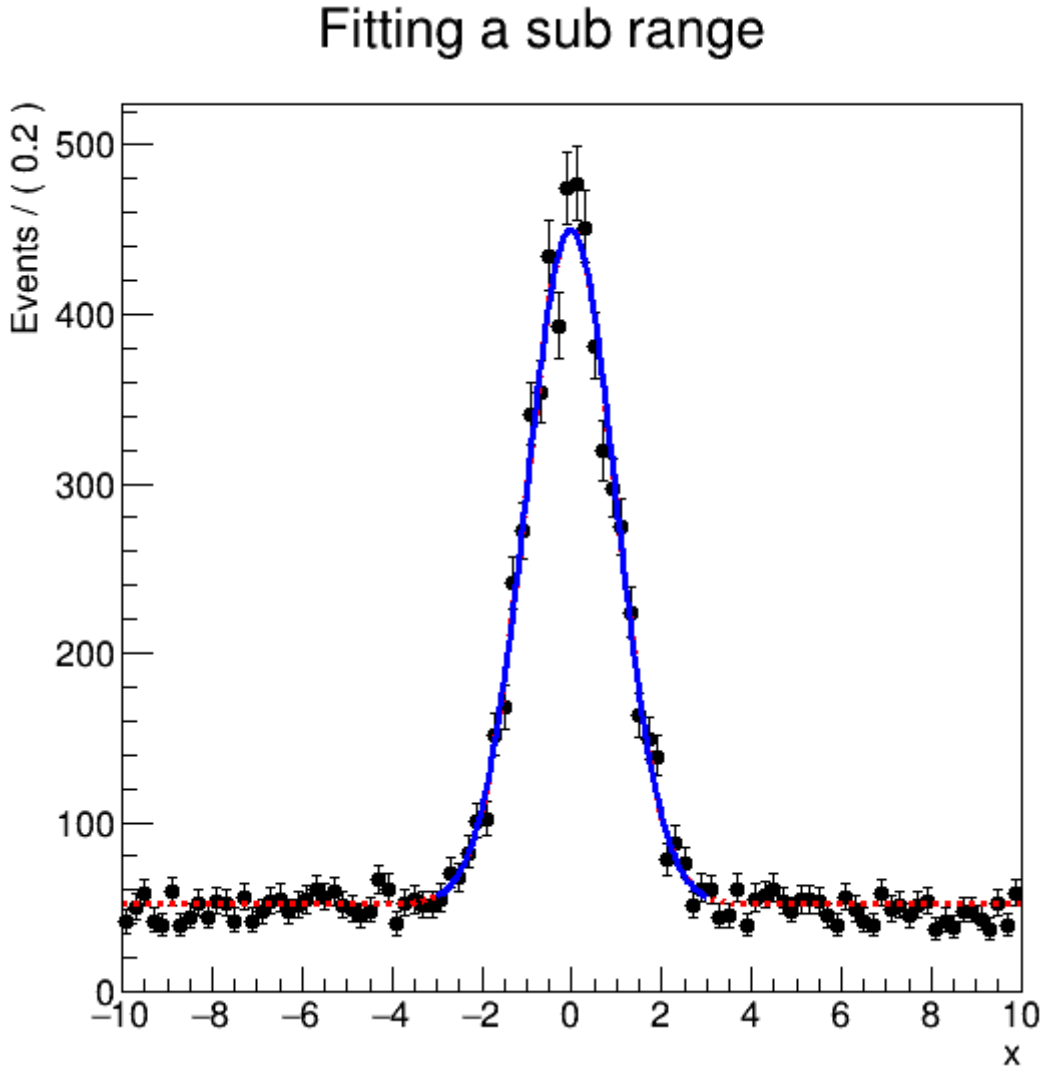
```
void rf202_extendedmlfit()
{

  // 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 their
parameters
  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.) ;
  RooRealVar a1("a1","a1",-0.2,0.,1.) ;
  RooChebychev bkg("bkg","Background",x,RooArgSet(a0,a1)) ;

  // Sum the signal components into a composite signal p.d.f.
  RooRealVar sig1frac("sig1frac","fraction of component 1 in signal",0.8,0.,1.)
;
  RooAddPdf sig("sig","Signal",RooArgList(sig1,sig2),sig1frac) ;
```

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

// Sum the composite signal and background into an extended pdf nsig*sig+nbkg*bkg
RooRealVar nsig("nsig","number of signal events",500,0.,10000) ;
RooRealVar nbkg("nbkg","number of background events",500,0,10000) ;
RooAddPdf  model("model","(g1+g2)+a",RooArgList(bkg,sig),RooArgList(nbkg,nsig)) ;



// S a m p l e ,   f i t   a n d   p l o t   e x t e n d e d   m o d e l
// ---------------------------------------------------------------------

// Generate a data sample of expected number events in x from model
// = model.expectedEvents() = nsig+nbkg
RooDataSet *data = model.generate(x) ;

// Fit model to data, extended ML term automatically included
model.fitTo(*data) ;
```

```
// Plot data and PDF overlaid, use expected number of events for p.d.f projection normalization
// rather than observed number of events (==data->numEntries())
RooPlot* xframe = x.frame(Title("extended ML fit example")) ;
data->plotOn(xframe) ;
model.plotOn(xframe,Normalization(1.0,RooAbsReal::RelativeExpected)) ;

// Overlay the background component of model with a dashed line
model.plotOn(xframe,Components(bkg),LineStyle(kDashed),Normalization(1.0,RooAbsReal::RelativeExpected)) ;

// Overlay the background+sig2 components of model with a dotted line
model.plotOn(xframe,Components(RooArgSet(bkg,sig2)),LineStyle(kDotted),Normalization(1.0,RooAbsReal::RelativeExpected)) ;

// Print structure of composite p.d.f.
model.Print("t") ;
```

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

// Associated nsig/nbkg as expected number of events with sig/bkg
RooExtendPdf esig("esig","extended signal p.d.f",sig,nsig) ;
RooExtendPdf ebkg("ebkg","extended background p.d.f",bkg,nbkg) ;


// S u m   e x t e n d e d   c o m p o n e n t s   w i t h o u t   c o e f s
// ------------------------------------------------------------------------

// Construct sum of two extended p.d.f. (no coefficients required)
RooAddPdf  model2("model2","(g1+g2)+a",RooArgList(ebkg,esig)) ;



// Draw the frame on the canvas
new TCanvas("rf202_composite","rf202_composite",600,600) ;
gPad->SetLeftMargin(0.15) ; xframe->GetYaxis()->SetTitleOffset(1.4) ; xframe->Draw() ;
```

$ROOTSYS/tutorials/roofit/rf203_ranges.C



siguang@pku.edu.cn

```
void rf203_ranges()
{
  // S e t u p   m o d e l
  // ---------------------

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

  // Construct gaussx(x,mx,1)
  RooRealVar mx("mx","mx",0,-10,10) ;
  RooGaussian gx("gx","gx",x,mx,RooConst(1)) ;

  // Construct px = 1 (flat in x)
  RooPolynomial px("px","px",x) ;

  // Construct model = f*gx + (1-f)px
  RooRealVar f("f","f",0.,1.) ;
  RooAddPdf model("model","model",RooArgList(gx,px),f) ;

  // Generated 10000 events in (x,y) from p.d.f. model
  RooDataSet* modelData = model.generate(x,10000) ;
```

```
// F i t   f u l l   r a n g e
// -------------------------

// Fit p.d.f to all data
RooFitResult* r_full = model.fitTo(*modelData,Save(kTRUE)) ;


// F i t   p a r t i a l   r a n g e
// -----------------------------------

// Define "signal" range in x as [-3,3]
x.setRange("signal",-3,3) ;

// Fit p.d.f only to data in "signal" range
RooFitResult* r_sig = model.fitTo(*modelData,Save(kTRUE),Range("signal")) ;
// P l o t   /   p r i n t   r e s u l t s
// ---------------------------------------

// Make plot frame in x and add data and fitted model
RooPlot* frame = x.frame(Title("Fitting a sub range")) ;
modelData->plotOn(frame) ;
model.plotOn(frame,Range("Full"),LineStyle(kDashed),LineColor(kRed)) ; // Add shape in full ranged dashed
model.plotOn(frame) ; // By default only fitted range is shown
```

所用的参数为最后一次拟合的数值

```
// Print fit results
cout << "result of fit on all data " << endl ;
r_full->Print() ;
cout << "result of fit in in signal region (note increased error on signal fraction)" << endl ;
r_sig->Print() ;

// Draw frame on canvas
new TCanvas("rf203_ranges","rf203_ranges",600,600) ;
gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4) ; frame->Draw() ;

return ;
```

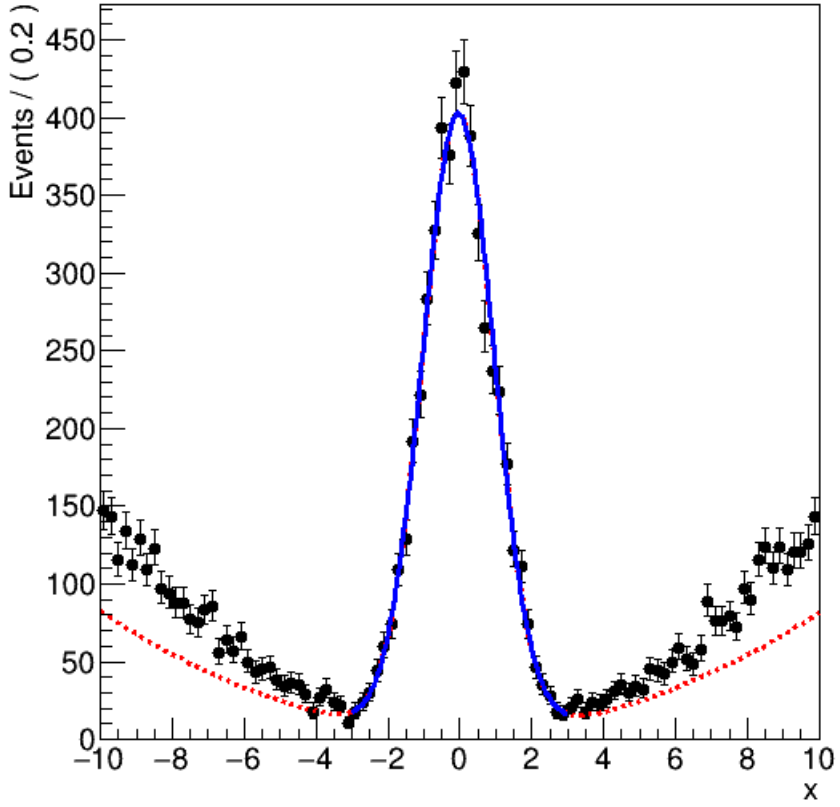
```
result of fit on all data

  RooFitResult: minimized FCN value: 25939.4, estimated distance to minimum: 7.54366e-06
                covariance matrix quality: Full, accurate covariance matrix
                Status : MIGRAD=0 HESSE=0

    Floating Parameter    FinalValue +/-  Error
  --------------------  --------------------------
                     f    5.0441e-01 +/-  6.32e-03
                    mx   -2.1605e-02 +/-  1.77e-02

result of fit in in signal region (note increased error on signal fraction)

  RooFitResult: minimized FCN value: 10339.5, estimated distance to minimum: 3.38975e-07
                covariance matrix quality: Full, accurate covariance matrix
                Status : MIGRAD=0 HESSE=0

    Floating Parameter    FinalValue +/-  Error
  --------------------  --------------------------
                     f    4.9014e-01 +/-  1.62e-02
                    mx   -2.1701e-02 +/-  1.79e-02
```

siguang@pku.edu.cn

```
// Construct px = 1 (flat in x)
// RooPolynomial px("px","px",x) ;
RooRealVar a0("a0","a0",1) ;
RooRealVar a1("a1","a1",0,-1,1) ;
RooRealVar a2("a2","a2",1,0,10) ;
RooPolynomial px("px","p2",x,RooArgList(a0,a1,a2),0) ;
```

$$f(x) = a_0 + a_1 x + a_2 x^2$$
LowestOrder=0

Fitting a sub range




```
result of fit on all data

  RooFitResult: minimized FCN value: 26442.4, estimated distance to minimum: 4.55229e-05
                covariance matrix quality: Full, accurate covariance matrix
                Status : MIGRAD=0 HESSE=0

    Floating Parameter    FinalValue +/-  Error
  --------------------  --------------------------
                    a1   -5.7140e-02 +/-  7.47e-02
                    a2    6.8082e-01 +/-  3.88e-01
                     f    5.0238e-01 +/-  5.93e-03
                    mx   -4.0218e-02 +/-  1.52e-02

result of fit in in signal region (note increased error on signal fraction)

  RooFitResult: minimized FCN value: 7577.89, estimated distance to minimum: 9.55043e-05
                covariance matrix quality: Full, accurate covariance matrix
                Status : MIGRAD=0 HESSE=0

    Floating Parameter    FinalValue +/-  Error
  --------------------  --------------------------
                    a1   -4.8898e-03 +/-  1.26e-01
                    a2    1.3139e-01 +/-  3.57e-01
                     f    6.1459e-01 +/-  2.05e-01
                    mx   -4.3672e-02 +/-  1.85e-02
```

siguang@pku.edu.cn

$ROOTSYS/tutorials/roofit/rf204_extrangefit.C

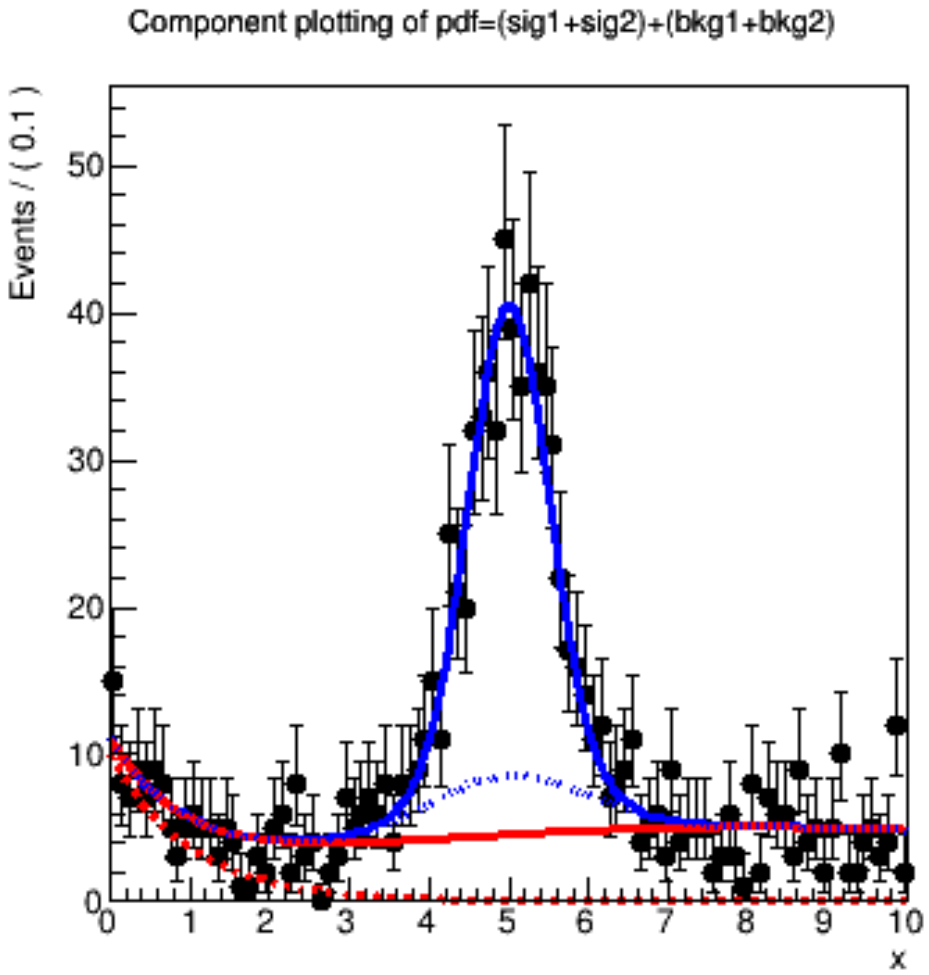
```
void rf204_extrangefit()
{


  // 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 their
parameters
  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.) ;
  RooRealVar a1("a1","a1",-0.2,0.,1.) ;
```

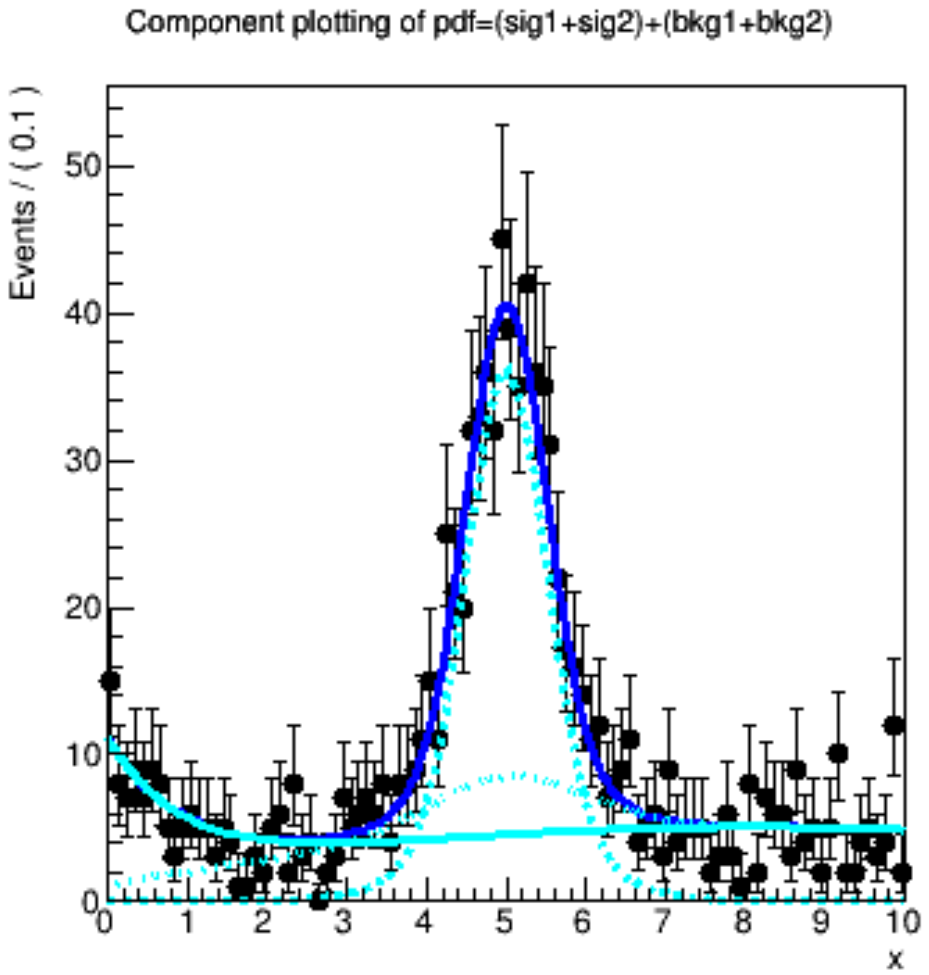
siguang@pku.edu.cn

```
RooChebychev bkg("bkg","Background",x,RooArgSet(a0,a1)) ;

// Sum the signal components into a composite signal p.d.f.
RooRealVar sig1frac("sig1frac","fraction of component 1 in signal",0.8,0.,1.) ;
RooAddPdf sig("sig","Signal",RooArgList(sig1,sig2),sig1frac) ;


// C o n s t r u c t   e x t e n d e d   c o m p s   wi t h   r a n g e   s p e c
// ---------------------------------------------------------------------

// Define signal range in which events counts are to be defined
x.setRange("signalRange",4,6) ;

// Associated nsig/nbkg as expected number of events with sig/bkg _in_the_range_ "signa
Range"
RooRealVar nsig("nsig","number of signal events in signalRange",500,0.,10000) ;
RooRealVar nbkg("nbkg","number of background events in signalRange",500,0,10000) ;
RooExtendPdf esig("esig","extended signal p.d.f",sig,nsig,"signalRange") ;
RooExtendPdf ebkg("ebkg","extended background p.d.f",bkg,nbkg,"signalRange") ;
```

```
  // S u m   e x t e n d e d   c o m p o n e n t s
  // ---------------------------------------------

  // Construct sum of two extended p.d.f. (no coefficients required)
  RooAddPdf  model("model","(g1+g2)+a",RooArgList(ebkg,esig)) ;


  // S a m p l e   d a t a ,   f i t   m o d e l
  // ------------------------------------------

  // Generate 1000 events from model so that nsig,nbkg come out to numbers <<500 in fit
  RooDataSet *data = model.generate(x,1000) ;


  // Perform unbinned extended ML fit to data
  RooFitResult* r = model.fitTo(*data,Extended(kTRUE),Save()) ;
  r->Print() ;

}
```

$ROOTSYS/tutorials/roofit/rf205_compplot.C





siguang@pku.edu.cn

```
void rf205_compplot()
{
  // S e t u p   c o m p o s i t e   p d f
  // ---------------------------------------

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

  // Create two Gaussian PDFs g1(x,mean1,sigma) anf g2(x,mean2,sigma) and their parameters
  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) ;

  // Sum the signal components into a composite signal p.d.f.
  RooRealVar sig1frac("sig1frac","fraction of component 1 in signal",0.8,0.,1.) ;
  RooAddPdf sig("sig","Signal",RooArgList(sig1,sig2),sig1frac) ;
```

```
// Build Chebychev polynomial p.d.f.
RooRealVar a0("a0","a0",0.5,0.,1.) ;
RooRealVar a1("a1","a1",-0.2,0.,1.) ;
RooChebychev bkg1("bkg1","Background 1",x,RooArgSet(a0,a1)) ;

// Build expontential pdf
RooRealVar alpha("alpha","alpha",-1) ;
RooExponential bkg2("bkg2","Background 2",x,alpha) ;

// Sum the background components into a composite background p.d.f.
RooRealVar bkg1frac("sig1frac","fraction of component 1 in background",0.2,0.,1.) ;
RooAddPdf bkg("bkg","Signal",RooArgList(bkg1,bkg2),sig1frac) ;

// Sum the composite signal and background
RooRealVar bkgfrac("bkgfrac","fraction of background",0.5,0.,1.) ;
RooAddPdf  model("model","g1+g2+a",RooArgList(bkg,sig),bkgfrac) ;
```


```
// S e t u p   b a s i c   p l o t   w i t h   d a t a   a n d   f u l l   p d f
// ---------------------------------------------------------------------------

// Generate a data sample of 1000 events in x from model
RooDataSet *data = model.generate(x,1000) ;

// Plot data and complete PDF overlaid
RooPlot* xframe  = x.frame(Title("Component plotting of pdf=(sig1+sig2)+(bkg1+bkg2)")) ;
data->plotOn(xframe) ;
model.plotOn(xframe) ;

// Clone xframe for use below
RooPlot* xframe2 = (RooPlot*) xframe->Clone("xframe2") ;


// M a k e   c o m p o n e n t   b y   o b j e c t   r e f e r e n c e
// ------------------------------------------------------------------

// Plot single background component specified by object reference
model.plotOn(xframe,Components(bkg),LineColor(kRed)) ;
```

```
// Plot single background component specified by object reference
model.plotOn(xframe,Components(bkg2),LineStyle(kDashed),LineColor(kRed)) ;

// Plot multiple background components specified by object reference
// Note that specified components may occur at any level in object tree
// (e.g bkg is component of 'model' and 'sig2' is component 'sig')
model.plotOn(xframe,Components(RooArgSet(bkg,sig2)),LineStyle(kDotted)) ;
```

Sig2是sig的组成部分

```
// M a k e   c o m p o n e n t   b y   n a m e /   r e g e x p
// ---------------------------------------------------------

// Plot single background component specified by name
model.plotOn(xframe2,Components("bkg"),LineColor(kCyan)) ;

// Plot multiple background components specified by name
model.plotOn(xframe2,Components("bkg1,sig2"),LineStyle(kDotted),LineColor(kCyan)) ;

// Plot multiple background components specified by regular expression on name
model.plotOn(xframe2,Components("sig*"),LineStyle(kDashed),LineColor(kCyan)) ;
```

```
   // Plot multiple background components specified by multiple regular expressions on name
   model.plotOn(xframe2,Components("bkg1,sig*"),LineStyle(kDashed),LineColor(kYellow),Invisible()) ;


   // Draw the frame on the canvas
   TCanvas* c = new TCanvas("rf205_compplot","rf205_compplot",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() ;


}
```

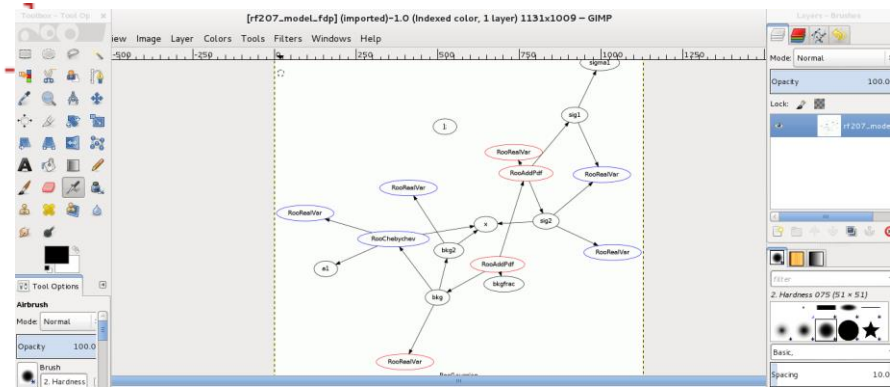
$ROOTSYS/tutorials/roofit/rf206_treevistools.C

```
void rf206_treevistools()
{
  // S e t u p   c o m p o s i t e   p d f
  // ---------------------------------------

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

  // Create two Gaussian PDFs g1(x,mean1,sigma) anf g2(x,mean2,sigma) and their parameters
  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) ;

  // Sum the signal components into a composite signal p.d.f.
  RooRealVar sig1frac("sig1frac","fraction of component 1 in signal",0.8,0.,1.) ;
  RooAddPdf sig("sig","Signal",RooArgList(sig1,sig2),sig1frac) ;
```

siguang@pku.edu.cn

```
// Build Chebychev polynomial p.d.f.
RooRealVar a0("a0","a0",0.5,0.,1.) ;
RooRealVar a1("a1","a1",-0.2,0.,1.) ;
RooChebychev bkg1("bkg1","Background 1",x,RooArgSet(a0,a1)) ;

// Build expontential pdf
RooRealVar alpha("alpha","alpha",-1) ;
RooExponential bkg2("bkg2","Background 2",x,alpha) ;

// Sum the background components into a composite background p.d.f.
RooRealVar bkg1frac("bkg1frac","fraction of component 1 in background",0.2,0.,1.) ;
RooAddPdf bkg("bkg","Signal",RooArgList(bkg1,bkg2),bkg1frac) ;

// Sum the composite signal and background
RooRealVar bkgfrac("bkgfrac","fraction of background",0.5,0.,1.) ;
RooAddPdf  model("model","g1+g2+a",RooArgList(bkg,sig),bkgfrac) ;
```

```
// P r i n t   c o m p o s i t e   t r e e   i n   A S C I I
// ------------------------------------------------------------

// Print tree to stdout
model.Print("t") ;

// Print tree to file
model.printCompactTree("","rf206_asciitree.txt") ;

// D r a w   c o m p o s i t e   t r e e   g r a p h i c a l
// ------------------------------------------------------------

// Print GraphViz DOT file with representation of tree
model.graphVizTree("rf206_model.dot") ;

// Make graphic output file with one of the GraphViz tools
// (freely available from www.graphviz.org)
//
// 'Top-to-bottom graph'
// unix> dot -Tgif -o rf207_model_dot.gif rf207_model.dot
//
// 'Spring-model graph'
// unix> fdp -Tgif -o rf207_model_fdp.gif rf207_model.dot

}
```



After run the dot or fdp command, use gimp to open the gif file.

rf207_model.dot should be rf206_model.dot

siguang@pku.edu.cn

直接运行后的结果

```
wsg@debian:~/work/root/534/tutorials/roofit/tmp$ root rf206_treevistools.C
root [0]
Processing rf206_treevistools.C...

RooFit v3.60 -- Developed by Wouter Verkerke and David Kirkby
                Copyright (C) 2000-2013 NIKHEF, University of California & Stanf
ord University
                All rights reserved, please read http://roofit.sourceforge.net/l
icense.txt
```

model.print("t") 的结果

```
0x316c060 RooAddPdf::model = 0.622695 [Auto,Dirty]
  0x3168210/V- RooAddPdf::bkg = 0.24539 [Auto,Dirty]
    0x3163910/V- RooChebychev::bkg1 = 1.2 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x3161b10/V- RooRealVar::a0 = 0.5
      0x3162510/V- RooRealVar::a1 = -0.2
    0x3166e70/V- RooRealVar::bkg1frac = 0.2
    0x3165890/V- RooExponential::bkg2 = 0.00673795 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x3164de0/V- RooRealVar::alpha = -1
  0x316acd0/V- RooRealVar::bkgfrac = 0.5
  0x315e0f0/V- RooAddPdf::sig = 1 [Auto,Dirty]
    0x3104610/V- RooGaussian::sig1 = 1 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x31022c0/V- RooRealVar::mean = 5
      0x3102f80/V- RooRealVar::sigma1 = 0.5
    0x3107860/V- RooRealVar::sig1frac = 0.8
    0x3106070/V- RooGaussian::sig2 = 1 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x31022c0/V- RooRealVar::mean = 5
      0x3103b00/V- RooRealVar::sigma2 = 1
root [1]
```

Dot的安装

```
wsg@debian:~/work/root/534/tutorials/roofit/tmp$ dot -Tgif -o rf206_model_dot.gif rf206_model.dot
bash: dot: command not found
```

按提示执行dot没有该命令！

```
wsg@debian:~/work/root/534/tutorials/roofit/tmp$ su
Password:
```

转换系统超级用户身份

```
root@debian:/home/wsg/work/root/534/tutorials/roofit/tmp# apt-cache search dot
xdot - interactive viewer for Graphviz dot files
```

找到xdot安装包

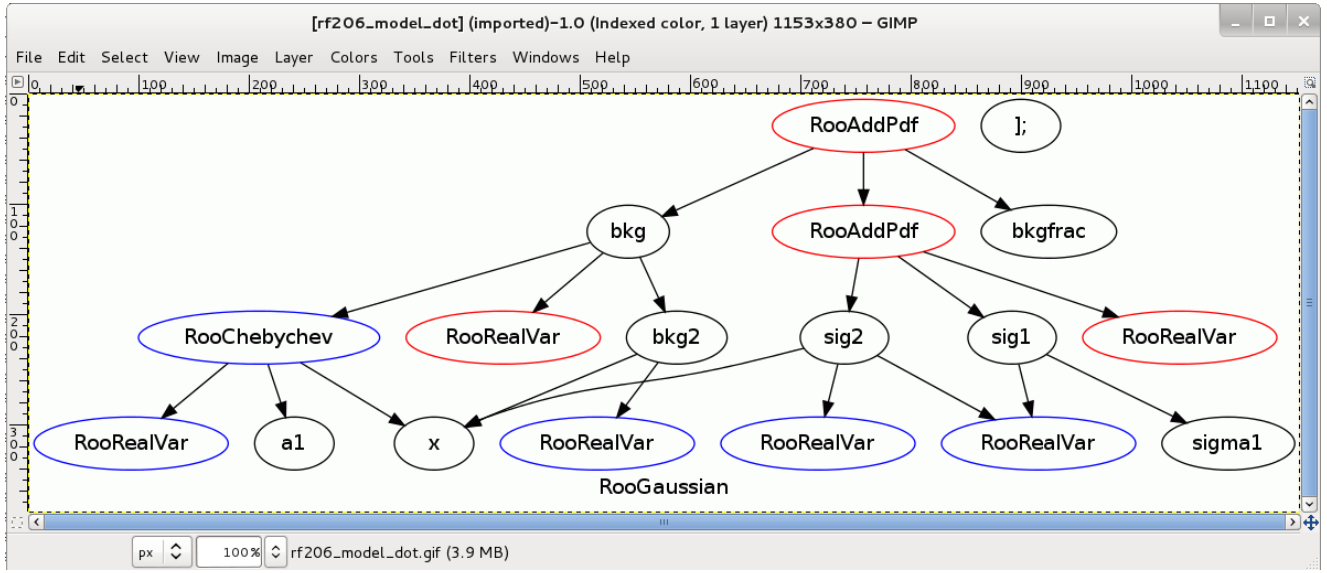
```
root@debian:/home/wsg/work/root/534/tutorials/roofit/tmp# apt-get install xdot
```

执行安装

```
root@debian:/home/wsg/work/root/534/tutorials/roofit/tmp# exit
exit
wsg@debian:~/work/root/534/tutorials/roofit/tmp$ dot -Tgif -o rf206_model_dot.gif rf206_model.dot
```

退出超级用户，重新以普通用户身份执行

用gimp打开转换后的gif文件



转换的时候发现有错误警告！
原因为每行多了个换行符合

```
digraph model{
"model" [ color=red, label="RooAddPdf
model"];
"bkg" [ color=red, label="RooAddPdf
bkg"];
"bkg1" [ color=red, label="RooChebychev
bkg1"];
"x" [ color=blue, label="RooRealVar
x"];
"a0" [ color=blue, label="RooRealVar
a0"];
"a1" [ color=blue, label="RooRealVar
a1"];
```

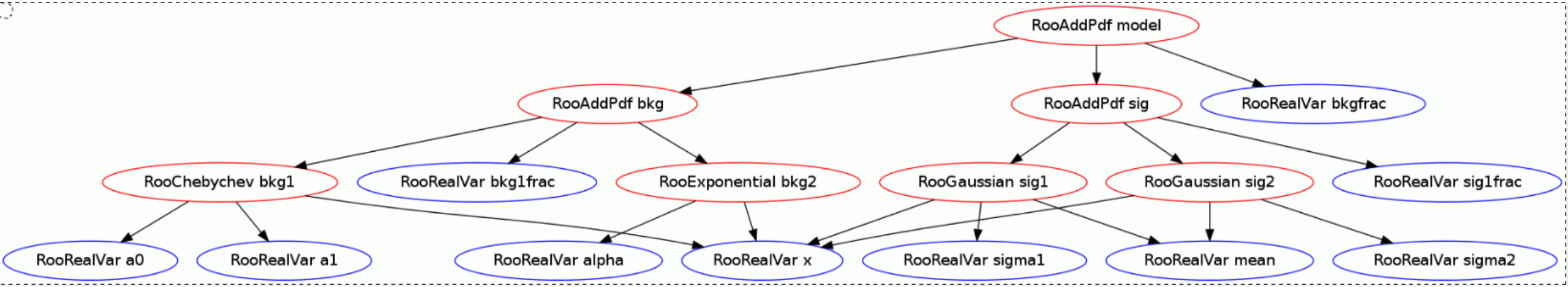
X

```
digraph model{
"model" [ color=red, label="RooAddPdf model"];
"bkg" [ color=red, label="RooAddPdf bkg"];
"bkg1" [ color=red, label="RooChebychev bkg1"];
"x" [ color=blue, label="RooRealVar x"];
"a0" [ color=blue, label="RooRealVar a0"];
"a1" [ color=blue, label="RooRealVar a1"];
"bkg1frac" [ color=blue, label="RooRealVar bkg1frac"];
"bkg2" [ color=red, label="RooExponential bkg2"];
"alpha" [ color=blue, label="RooRealVar alpha"];
"bkgfrac" [ color=blue, label="RooRealVar bkgfrac"];
"sig" [ color=red, label="RooAddPdf sig"];
"sig1" [ color=red, label="RooGaussian sig1"];
"mean" [ color=blue, label="RooRealVar mean"];
"sigma1" [ color=blue, label="RooRealVar sigma1"];
"sig1frac" [ color=blue, label="RooRealVar sig1frac"];
"sig2" [ color=red, label="RooGaussian sig2"];
"sigma2" [ color=blue, label="RooRealVar sigma2"];
"sig1" -> "x";
"sig1" -> "mean";
"sig1" -> "sigma1";
"sig2" -> "x";
"sig2" -> "mean";
"sig2" -> "sigma2";
"sig" -> "sig1";
"sig" -> "sig2";
"sig" -> "sig1frac";
"bkg1" -> "x";
"bkg1" -> "a0";
"bkg1" -> "a1";
"bkg2" -> "x";
"bkg2" -> "alpha";
"bkg" -> "bkg1";
:---  rf206_1.dot    Top L1     (Fundamental)----------
```

√

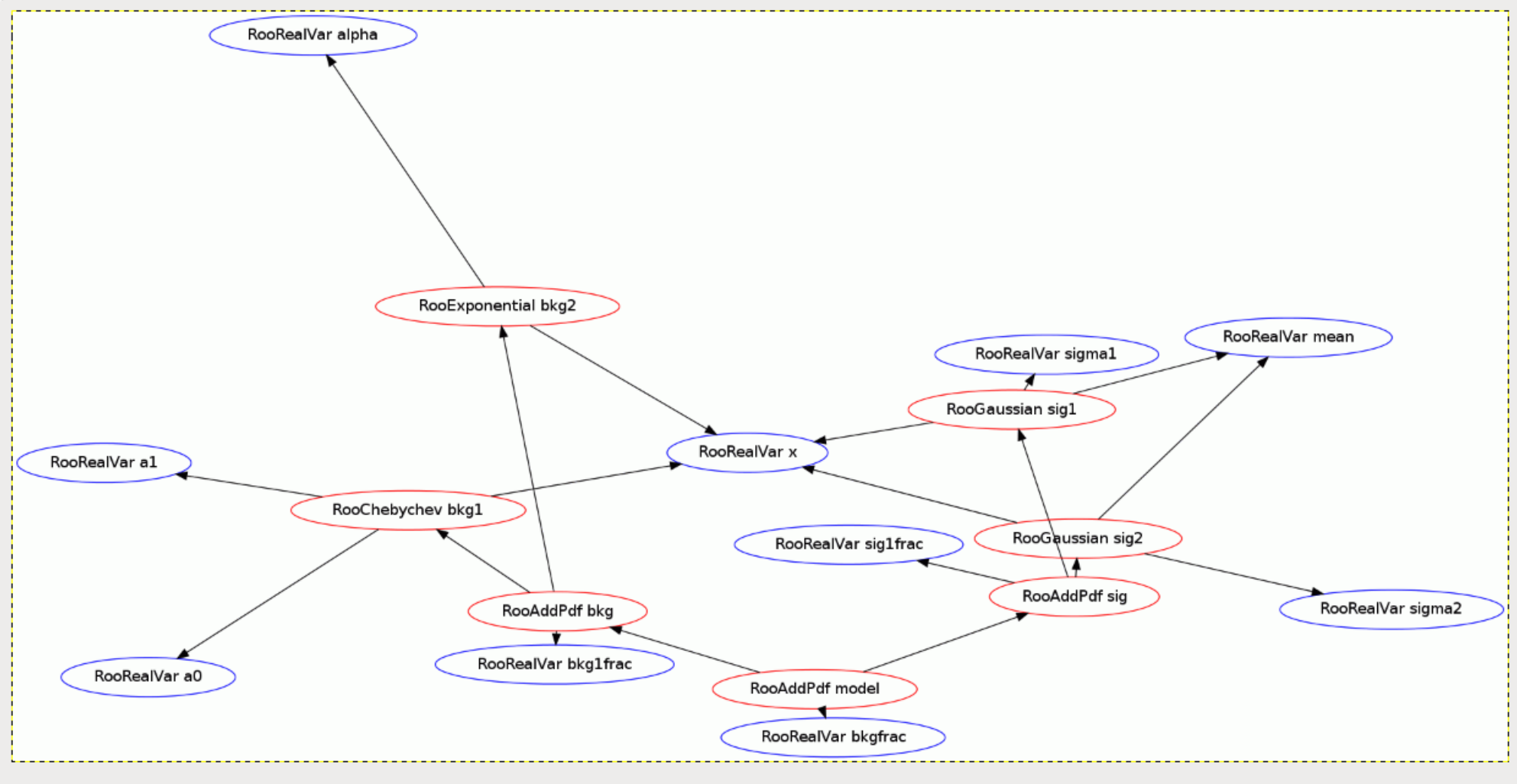
正确的输出（gimp打开dot转换的gif文件）

```
$dot -Tgif -o rf206_1.gif      rf206_1.dot
$gimp rf206_1.giv
```



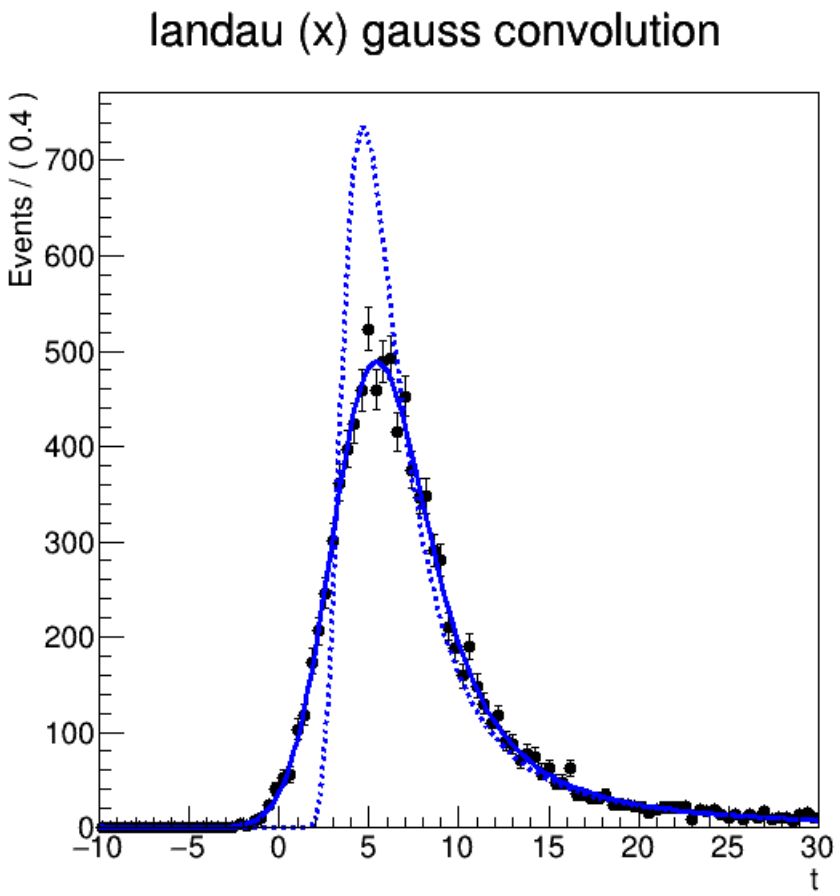
正确的输出（gimp打开fdp dot转换的gif文件）

```
$fdp -Tgif  -o    rf206_2.gif     rf206_1.dot
$gimp  rf206_2.giv
```



```
wsg@debian:~/work/root/534/tutorials/roofit/tmp$ cat rf206_asciitree.txt
0x316c060 RooAddPdf::model = 0.622695 [Auto,Dirty]
  0x3168210/V- RooAddPdf::bkg = 0.24539 [Auto,Dirty]
    0x3163910/V- RooChebychev::bkg1 = 1.2 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x3161b10/V- RooRealVar::a0 = 0.5
      0x3162510/V- RooRealVar::a1 = -0.2
    0x3166e70/V- RooRealVar::bkg1frac = 0.2
    0x3165890/V- RooExponential::bkg2 = 0.00673795 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x3164de0/V- RooRealVar::alpha = -1
  0x316acd0/V- RooRealVar::bkgfrac = 0.5
  0x315e0f0/V- RooAddPdf::sig = 1 [Auto,Dirty]
    0x3104610/V- RooGaussian::sig1 = 1 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x31022c0/V- RooRealVar::mean = 5
      0x3102f80/V- RooRealVar::sigma1 = 0.5
    0x3107860/V- RooRealVar::sig1frac = 0.8
    0x3106070/V- RooGaussian::sig2 = 1 [Auto,Dirty]
      0x30fe640/V- RooRealVar::x = 5
      0x31022c0/V- RooRealVar::mean = 5
      0x3103b00/V- RooRealVar::sigma2 = 1
wsg@debian:~/work/root/534/tutorials/roofit/tmp$
```

$ROOTSYS/tutorials/roofit/rf208_convolution.C

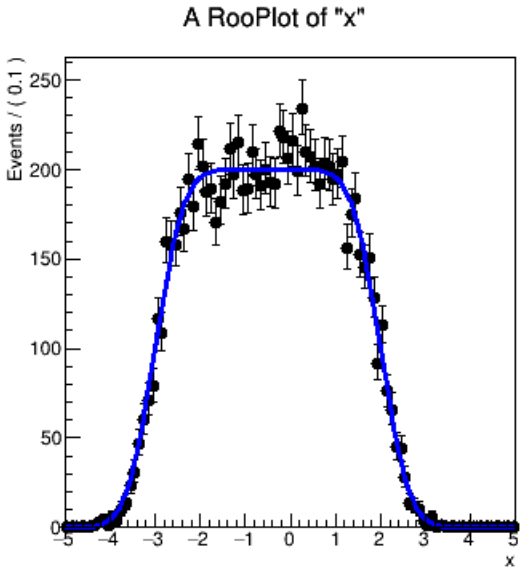


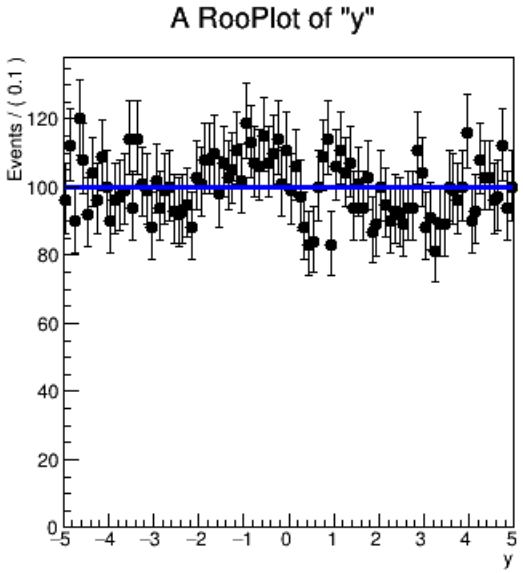
siguang@pku.edu.cn

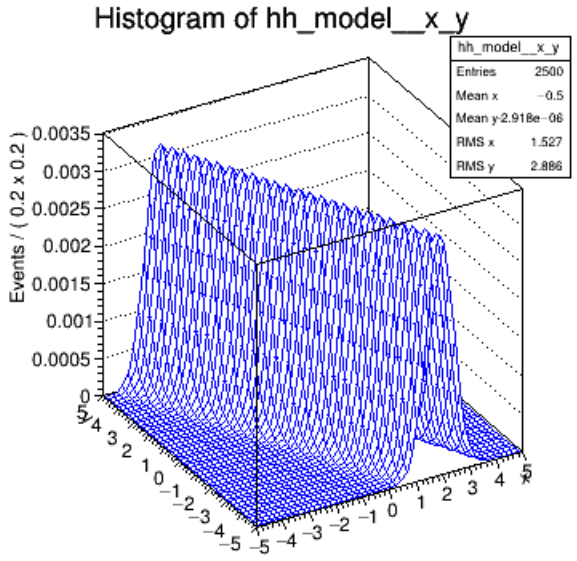
```
void rf208_convolution()
{
 // S e t u p   c o m p o n e n t   p d f s
 // ---------------------------------------

 // 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) ;


 // C o n s t r u c t   c o n v o l u t i o n   p d f
 // -------------------------------------------------

 // 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) ;
```

```
// Sample, fit and plot convoluted pdf
// ---------------------------------------------------------

// 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() ;

}
```

$ROOTSYS/tutorials/roofit/rf301_composition.C







siguang@pku.edu.cn

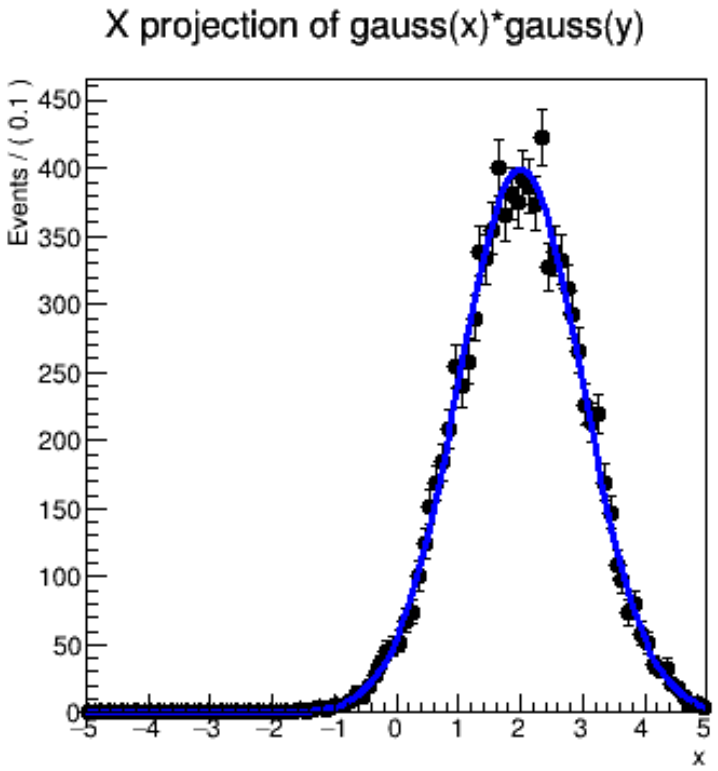
```
void rf301_composition()
{
  // S e t u p   c o m p o s e d   m o d e l   g a u s s ( x , m ( y ) , s )
  // ---------------------------------------------------------------

  // 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)) ;

  // Creat gauss(x,f(y),s)
  RooRealVar sigma("sigma","width of gaussian",0.5) ;
  RooGaussian model("model","Gaussian with shifting mean",x,fy,sigma) ;


  // S a m p l e   d a t a ,   p l o t   d a t a   a n d   p d f   o n   x   a n d   y
  // ------------------------------------------------------------------------------

  // Generate 10000 events in x and y from model
  RooDataSet *data = model.generate(RooArgSet(x,y),10000) ;
```

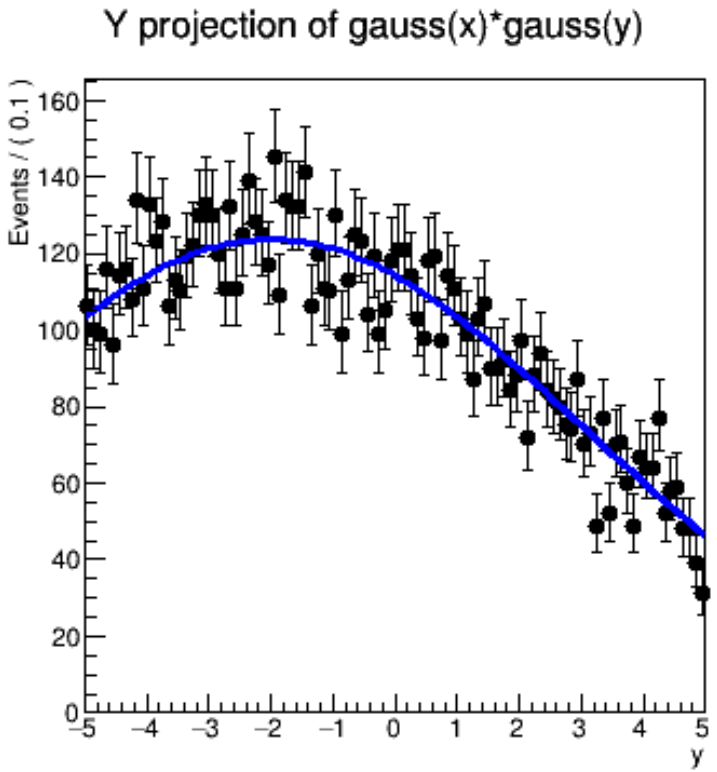
siguang@pku.edu.cn

```
// Plot x distribution of data and projection of model on x = Int(dy) model(x,y)
RooPlot* xframe = x.frame() ;
data->plotOn(xframe) ;
model.plotOn(xframe) ;

// Plot x distribution of data and projection of model on y = Int(dx) model(x,y)
RooPlot* yframe = y.frame() ;
data->plotOn(yframe) ;
model.plotOn(yframe) ;

// Make two-dimensional plot in x vs y
TH1* hh_model = model.createHistogram("hh_model",x,Binning(50),YVar(y,Binning(50))) ;
hh_model->SetLineColor(kBlue) ;



// Make canvas and draw RooPlots
TCanvas *c = new TCanvas("rf301_composition","rf301_composition",1200, 400);
c->Divide(3);
c->cd(1) ; gPad->SetLeftMargin(0.15) ; xframe->GetYaxis()->SetTitleOffset(1.4) ; xframe->Draw() ;
c->cd(2) ; gPad->SetLeftMargin(0.15) ; yframe->GetYaxis()->SetTitleOffset(1.4) ; yframe->Draw() ;
c->cd(3) ; gPad->SetLeftMargin(0.20) ; hh_model->GetZaxis()->SetTitleOffset(2.5) ; hh_model->Draw("surf") ;

}
```

$ROOTSYS/tutorials/roofit/rf304_uncorrprod.C

// pdf = gauss(x,mx,sx) * gauss(y,my,sy)





注意：x-y一定要非相关！

siguang@pku.edu.cn

```
// C r e a t e   c o m p o n e n t   p d f s   i n   x   a n d   y
// ---------------------------------------------------------

// Create two p.d.f.s gaussx(x,meanx,sigmax) gaussy(y,meany,sigmay) and its variables
RooRealVar x("x","x",-5,5) ;
RooRealVar y("y","y",-5,5) ;

RooRealVar meanx("mean1","mean of gaussian x",2) ;
RooRealVar meany("mean2","mean of gaussian y",-2) ;
RooRealVar sigmax("sigmax","width of gaussian x",1) ;
RooRealVar sigmay("sigmay","width of gaussian y",5) ;

RooGaussian gaussx("gaussx","gaussian PDF",x,meanx,sigmax) ;
RooGaussian gaussy("gaussy","gaussian PDF",y,meany,sigmay) ;



// C o n s t r u c t   u n c o r r e l a t e d   p r o d u c t   p d f
// ---------------------------------------------------------------

// Multiply gaussx and gaussy into a two-dimensional p.d.f. gaussxy
RooProdPdf  gaussxy("gaussxy","gaussx*gaussy",RooArgList(gaussx,gaussy)) ;
```

```
// Sample pdf, plot projection on x and y
// ---------------------------------------

// Generate 10000 events in x and y from gaussxy
RooDataSet *data = gaussxy.generate(RooArgSet(x,y),10000) ;

// Plot x distribution of data and projection of gaussxy on x = Int(dy) gaussxy(x,y)
RooPlot* xframe = x.frame(Title("X projection of gauss(x)*gauss(y)")) ;
data->plotOn(xframe) ;
gaussxy.plotOn(xframe) ;

// Plot x distribution of data and projection of gaussxy on y = Int(dx) gaussxy(x,y)
RooPlot* yframe = y.frame(Title("Y projection of gauss(x)*gauss(y)")) ;
data->plotOn(yframe) ;
gaussxy.plotOn(yframe) ;



// Make canvas and draw RooPlots
TCanvas *c = new TCanvas("rf304_uncorrprod","rf304_uncorrprod",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) ; yframe->GetYaxis()->SetTitleOffset(1.4) ; yframe->Draw() ;

}
```

$ROOTSYS/tutorials/roofit/rf402_datahandling.C


```
void rf402_datahandling()
{

  // Binned (RooDataHist) and unbinned datasets (RooDataSet) share
  // many properties and inherit from a common abstract base class
  // (RooAbsData), that provides an interface for all operations
  // that can be performed regardless of the data format

  RooRealVar x("x","x",-10,10) ;
  RooRealVar y("y","y", 0, 40) ;
  RooCategory c("c","c") ;
  c.defineType("Plus",+1) ;
  c.defineType("Minus",-1) ;


  // B a s i c   O p e r a t i o n s   o n   u n b i n n e d   d a t a s e t s
  // ---------------------------------------------------------

  // RooDataSet is an unbinned dataset (a collection of points in N-dimensional space)
  RooDataSet d("d","d",RooArgSet(x,y,c)) ;
```

siguang@pku.edu.cn

```
// Unlike RooAbsArgs (RooAbsPdf,RooFormulaVar,....) datasets are not attached to
// the variables they are constructed from. Instead they are attached to an internal
// clone of the supplied set of arguments

// Fill d with dummy values
Int_t i ;
for (i=0 ; i<1000 ; i++) {
  x = i/50 - 10 ;
  y = sqrt(1.0*i) ;
  c.setLabel((i%2)?"Plus":"Minus") ;

  // We must explicitly refer to x,y,c here to pass the values because
  // d is not linked to them (as explained above)
  d.add(RooArgSet(x,y,c)) ;
}
d.Print("v") ;
cout << endl ;

// The get() function returns a pointer to the internal copy of the RooArgSet(x,y,c)
// supplied in the constructor
const RooArgSet* row = d.get() ;
row->Print("v") ;
cout << endl ;
```

注意：i/50是整数除以整数，结果取整。例如40/50=0

实验数据构造方法

```
DataStore d (d)
  Contains 1000 entries
  Observables:
    1)  x = 9  L(-10 - 10)  "x"
    2)  y = 31.607  L(0 - 40)  "y"
    3)  c = Plus(idx = 1)
  "c"
```

```
1) 0x29cd000 RooRealVar:: x = 9  L(-10 - 10)  "x"
2) 0x29cdba0 RooRealVar:: y = 31.607  L(0 - 40)  "y"
3) 0x29ce4d0 RooCategory:: c = Plus(idx = 1)
"c"
```

返回当前值。即第1000个数

siguang@pku.edu.cn

```
// Get with an argument loads a specific data point in row and returns
// a pointer to row argset. get() always returns the same pointer, unless
// an invalid row number is specified. In that case a null ptr is returned
d.get(900)->Print("v") ;
cout << endl ;
```

```
1) 0x1b05c90 RooRealVar:: x = 8  L(-10 - 10)  "x"
2) 0x1b06830 RooRealVar:: y = 30  L(0 - 40)  "y"
3) 0x1b07160 RooCategory:: c = Minus(idx = -1)
"c"
```

结果是第901个数。从0开始！

siguang@pku.edu.cn

```
// Reducing, Appending and Merging
// -------------------------------------------------------

// The reduce() function returns a new dataset which is a subset of the original
cout << endl << ">> d1 has only columns x,c" << endl ;
RooDataSet* d1 = (RooDataSet*) d.reduce(RooArgSet(x,c)) ;
d1->Print("v") ;

cout << endl << ">> d2 has only column y" << endl ;
RooDataSet* d2 = (RooDataSet*) d.reduce(RooArgSet(y)) ;
d2->Print("v") ;

cout << endl << ">> d3 has only the points with y>5.17" << endl ;
RooDataSet* d3 = (RooDataSet*) d.reduce("y>5.17") ;
d3->Print("v") ;

cout << endl << ">> d4 has only columns x,c for data points with y>5.17" << endl ;
RooDataSet* d4 = (RooDataSet*) d.reduce(RooArgSet(x,c),"y>5.17") ;
d4->Print("v") ;

// The merge() function adds two data set column-wise
cout << endl << ">> merge d2(y) with d1(x,c) to form d1(x,c,y)" << endl ;
d1->merge(d2) ;
d1->Print("v") ;
```

```
// The append() function addes two datasets row-wise
cout << endl << ">> append data points of d3 to d1" << endl ;
d1->append(*d3) ;
d1->Print("v") ;
```

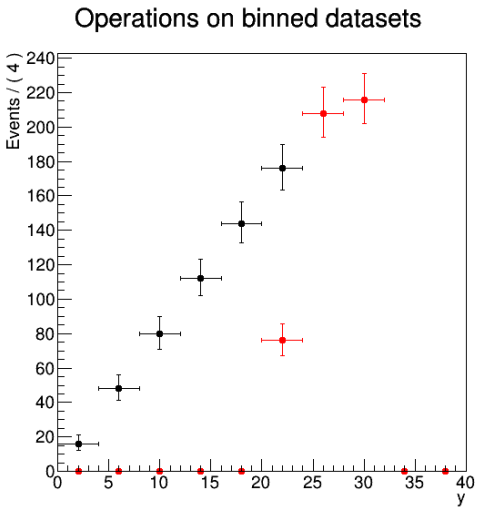
```
// The append() function addes two datasets row-wise
cout << endl << ">> append data points of d3 to d1" << endl ;

d1->Print("v") ;
d3->Print("v") ;
d1->append(*d3) ;
d1->Print("v") ;
```

```
>> append data points of d3 to d1
)ataStore d (d)
  Contains 1000 entries
  Observables:
    1)  x = 9  L(-10 - 10)  "x"
    2)  c = Plus(idx = 1)
  "c"
    3)  y = 31.607  L(0 - 40)  "y"
)ataStore d (d)
  Contains 973 entries
  Observables:
    1)  x = 9  L(-10 - 10)  "x"
    2)  y = 31.607  L(0 - 40)  "y"
    3)  c = Plus(idx = 1)
  "c"
)ataStore d (d)
  Contains 1973 entries
  Observables:
    1)  x = 9  L(-10 - 10)  "x"
    2)  c = Plus(idx = 1)
  "c"
    3)  y = 31.607  L(0 - 40)  "y"
```

siguang@pku.edu.cn

```
// O p e r a t i o n s   o n   b i n n e d   d a t a s e t s
// ---------------------------------------------------------

// A binned dataset can be constructed empty, from an unbinned dataset, or
// from a ROOT native histogram (TH1,2,3)

cout << ">> construct dh (binned) from d(unbinned) but only take the x and y dimensions," << endl
   << ">> the category 'c' will be projected in the filling process" << endl ;

// The binning of real variables (like x,y) is done using their fit range
// 'get/setRange()' and number of specified fit bins 'get/setBins()'.
// Category dimensions of binned datasets get one bin per defined category state
x.setBins(10) ;
y.setBins(10) ;
RooDataHist dh("dh","binned version of d",RooArgSet(x,y),d) ;
dh.Print("v") ;

RooPlot* yframe = y.frame(Bins(10),Title("Operations on binned datasets")) ;
dh.plotOn(yframe) ; // plot projection of 2D binned data on y

// Examine the statistics of a binned dataset
cout << ">> number of bins in dh    : " << dh.numEntries() << endl ;
cout << ">> sum of weights in dh    : " << dh.sum(kFALSE)  << endl ;
cout << ">> integral over histogram: " << dh.sum(kTRUE)   << endl ; // accounts for bin volume
```

Double_t **sum** (Bool_t correctForBinSize)

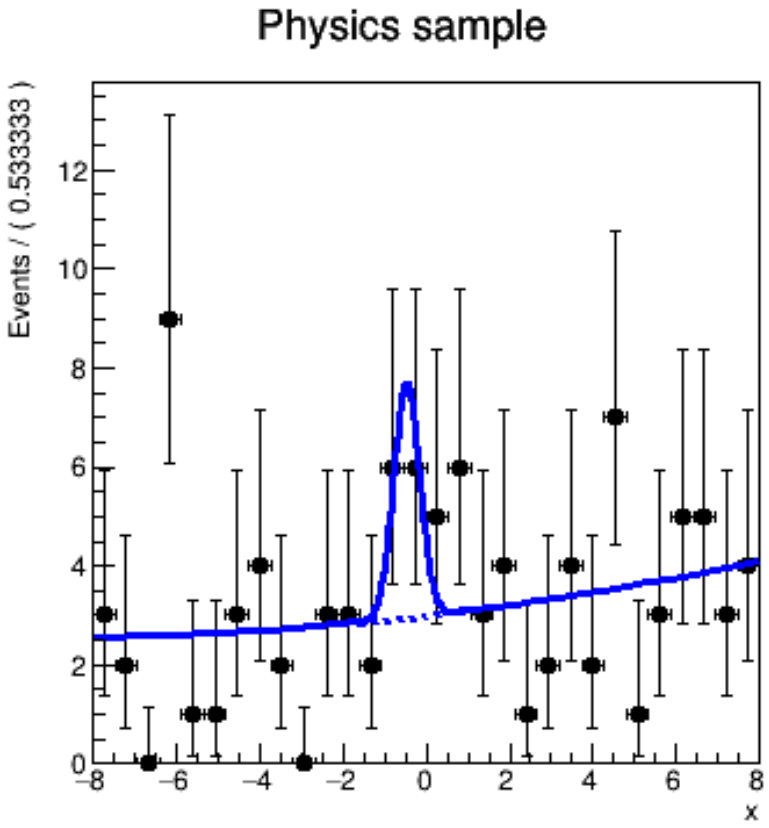


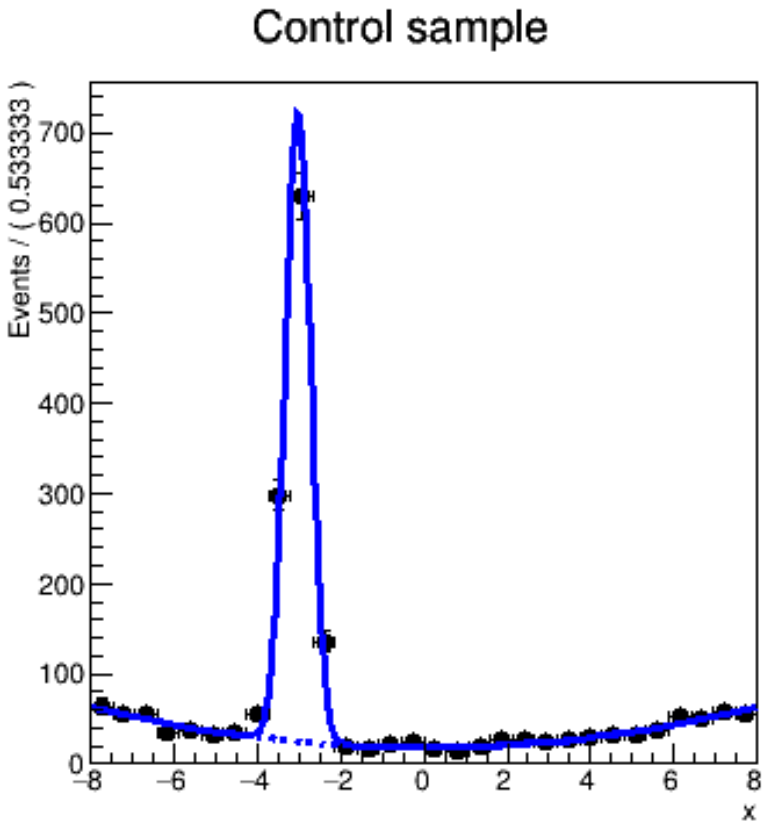
siguang@pku.edu.cn

```
// Locate a bin from a set of coordinates and retrieve its properties
x = 0.3 ; y = 20.5 ;
cout << ">> retrieving the properties of the bin enclosing coordinate (x,y) = (0.3,20.5) " << endl ;
cout << " bin center:" << endl ;
dh.get(RooArgSet(x,y))->Print("v") ; // load bin center coordinates in internal buffer
cout << " weight = " << dh.weight() << endl ; // return weight of last loaded coordinates

// Reduce the 2-dimensional binned dataset to a 1-dimensional binned dataset
//
// All reduce() methods are interfaced in RooAbsData. All reduction techniques
// demonstrated on unbinned datasets can be applied to binned datasets as well.
cout << ">> Creating 1-dimensional projection on y of dh for bins with x>0" << endl ;
RooDataHist* dh2 = (RooDataHist*) dh.reduce(y,"x>0") ;
dh2->Print("v") ;

// Add dh2 to yframe and redraw
dh2->plotOn(yframe,LineColor(kRed),MarkerColor(kRed)) ;
```


```
  // S a v i n g   a n d   l o a d i n g   f r o m   f i l e
  // ---------------------------------------------------

  // Datasets can be persisted with ROOT I/O
  cout << endl << ">> Persisting d via ROOT I/O" << endl ;
  TFile f("rf402_datahandling.root","RECREATE") ;
  d.Write() ;
  f.ls() ;

  // To read back in future session:
  // > TFile f("rf402_datahandling.root") ;
  // > RooDataSet* d = (RooDataSet*) f.FindObject("d") ;



  new TCanvas("rf402_datahandling","rf402_datahandling",600,600) ;
  gPad->SetLeftMargin(0.15) ; yframe->GetYaxis()->SetTitleOffset(1.4) ; yframe->Draw() ;

}
```

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C





siguang@pku.edu.cn

```
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) ;

  // Construct signal pdf
  RooRealVar mean("mean","mean",0,-8,8) ;
  RooRealVar sigma("sigma","sigma",0.3,0.1,10) ;
  RooGaussian gx("gx","gx",x,mean,sigma) ;

  // Construct background pdf
  RooRealVar a0("a0","a0",-0.1,-1,1) ;
  RooRealVar a1("a1","a1",0.004,-1,1) ;
  RooChebychev px("px","px",x,RooArgSet(a0,a1)) ;

  // Construct composite pdf
  RooRealVar f("f","f",0.2,0.,1.) ;
  RooAddPdf model("model","model",RooArgList(gx,px),f) ;
```

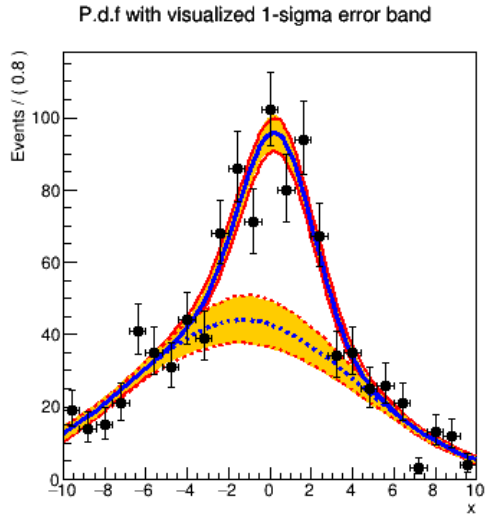
```
// C r e a t e   m o d e l   f o r   c o n t r o l   s a m p l e
// ---------------------------------------------------------

// Construct signal pdf.
// NOTE that sigma is shared with the signal sample model
RooRealVar mean_ctl("mean_ctl","mean_ctl",-3,-8,8) ;
RooGaussian gx_ctl("gx_ctl","gx_ctl",x,mean_ctl,sigma) ;

// Construct the background pdf
RooRealVar a0_ctl("a0_ctl","a0_ctl",-0.1,-1,1) ;
RooRealVar a1_ctl("a1_ctl","a1_ctl",0.5,-0.1,1) ;
RooChebychev px_ctl("px_ctl","px_ctl",x,RooArgSet(a0_ctl,a1_ctl)) ;

// Construct the composite model
RooRealVar f_ctl("f_ctl","f_ctl",0.5,0.,1.) ;
RooAddPdf model_ctl("model_ctl","model_ctl",RooArgList(gx_ctl,px_ctl),f_ctl) ;
```

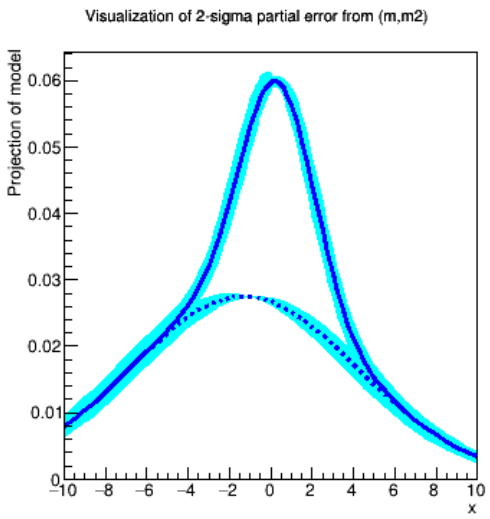
```
// G e n e r a t e   e v e n t s   f o r   b o t h   s a m p l e s
// -------------------------------------------------------------

// Generate 1000 events in x and y from model
RooDataSet *data = model.generate(RooArgSet(x),100) ;
RooDataSet *data_ctl = model_ctl.generate(RooArgSet(x),2000) ;



// C r e a t e   i n d e x   c a t e g o r y   a n d   j o i n   s a m p l e s
// ---------------------------------------------------------------------------

// Define category to distinguish physics and control samples events
RooCategory sample("sample","sample") ;
sample.defineType("physics") ;
sample.defineType("control") ;

// Construct combined dataset in (x,sample)
RooDataSet combData("combData","combined data",x,Index(sample),Import("physics",*data),Import("control",*
data_ctl)) ;
```

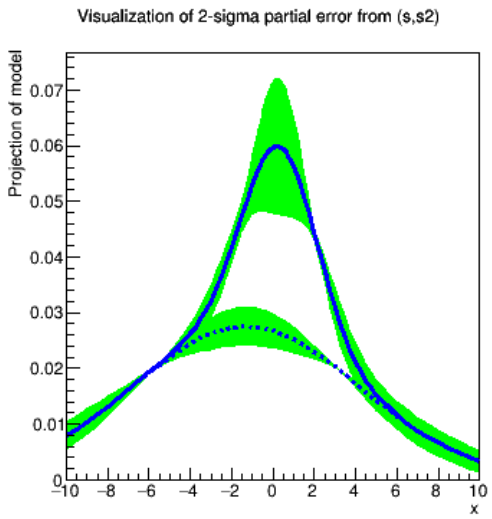
```
// C o n s t r u c t   a   s i m u l t a n e o u s   p d f   i n   ( x , s a m p l e )
// ---------------------------------------------------------------------

// Construct a simultaneous pdf using category sample as index
RooSimultaneous simPdf("simPdf","simultaneous pdf",sample) ;

// Associate model with the physics state and model_ctl with the control state
simPdf.addPdf(model,"physics") ;
simPdf.addPdf(model_ctl,"control") ;



// P e r f o r m   a   s i m u l t a n e o u s   f i t
// ------------------------------------------------

// Perform simultaneous fit of model to data and model_ctl to data_ctl
simPdf.fitTo(combData) ;
```

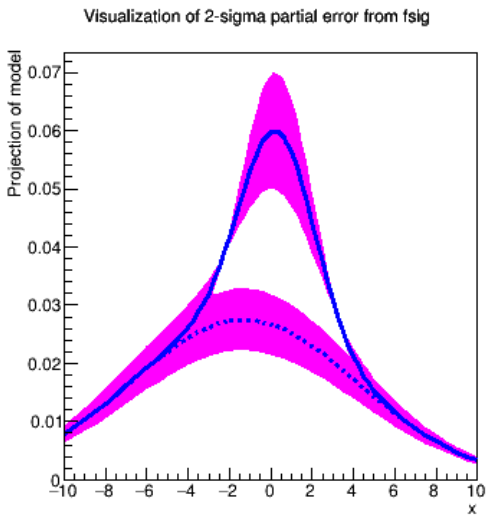
```
// Make a frame for the physics sample
RooPlot* frame1 = x.frame(Bins(30),Title("Physics sample")) ;

// Plot all data tagged as physics sample
combData.plotOn(frame1,Cut("sample==sample::physics")) ;

// Plot "physics" slice of simultaneous pdf.
// NBL You _must_ project the sample index category with data using ProjWData
// as a RooSimultaneous makes no prediction on the shape in the index category
// and can thus not be integrated
simPdf.plotOn(frame1,Slice(sample,"physics"),ProjWData(sample,combData)) ;
simPdf.plotOn(frame1,Slice(sample,"physics"),Components("px"),ProjWData(sample,combData),LineStyle(kDashed)) ;

// The same plot for the control sample slice
RooPlot* frame2 = x.frame(Bins(30),Title("Control sample")) ;
combData.plotOn(frame2,Cut("sample==sample::control")) ;
simPdf.plotOn(frame2,Slice(sample,"control"),ProjWData(sample,combData)) ;
simPdf.plotOn(frame2,Slice(sample,"control"),Components("px_ctl"),ProjWData(sample,combData),LineStyle(kDashed
)) ;



TCanvas* c = new TCanvas("rf501_simultaneouspdf","rf403_simultaneouspdf",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() ;
```

$ROOTSYS/tutorials/roofit/rf610_visualerror.C









siguang@pku.edu.cn

```
void rf610_visualerror()
{
  // S e t u p   e x a m p l e   f i t
  // ----------------------------------

  // 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()) ;
```

```
// V i s u a l i z e   f i t   e r r o r
// ------------------------------------

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

// 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)) ;
```

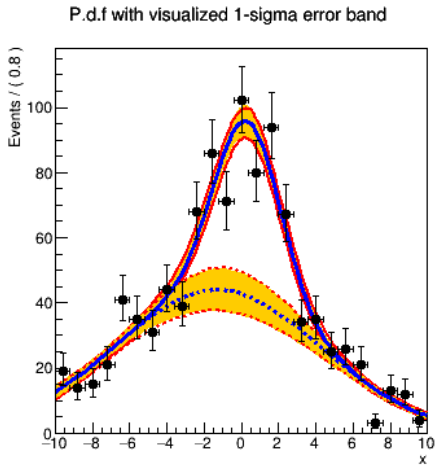
数据本身是有bin的，没看到起作用

快速，但相关性强或2σ以上的误差不够精确

```
// 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)) ;

// 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"),
LineStyle(kDashed)) ;

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

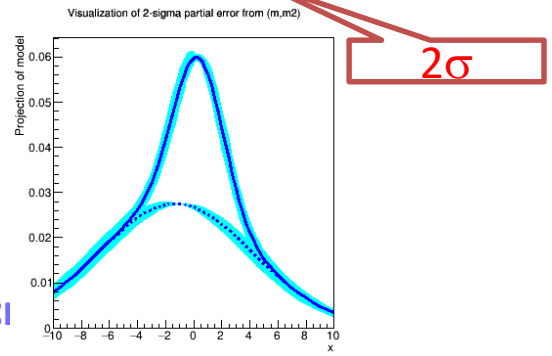
1σ

抽样方法



siguang@pku.edu.cn

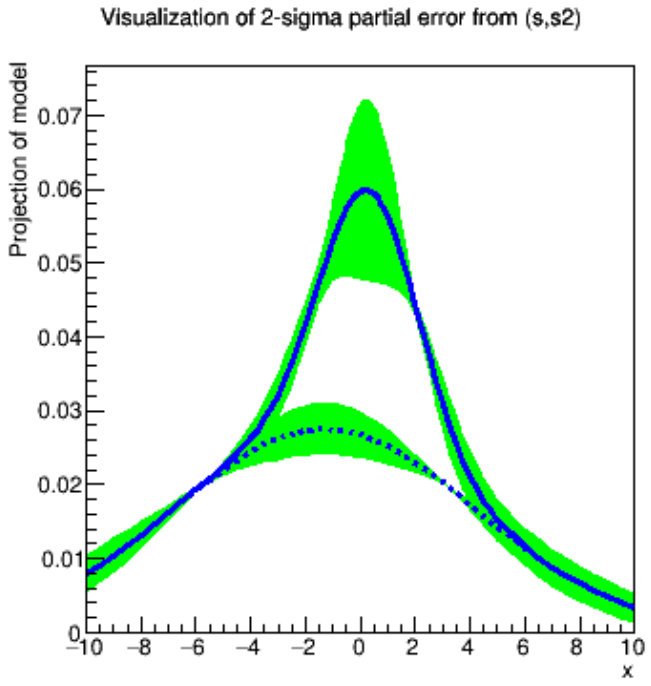
```
// V i s u a l i z e   p a r t i a l   f i t   e r r o r
// ---------------------------------------------------

// Make plot frame
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 = V22bar  = 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) ;
```



2σ

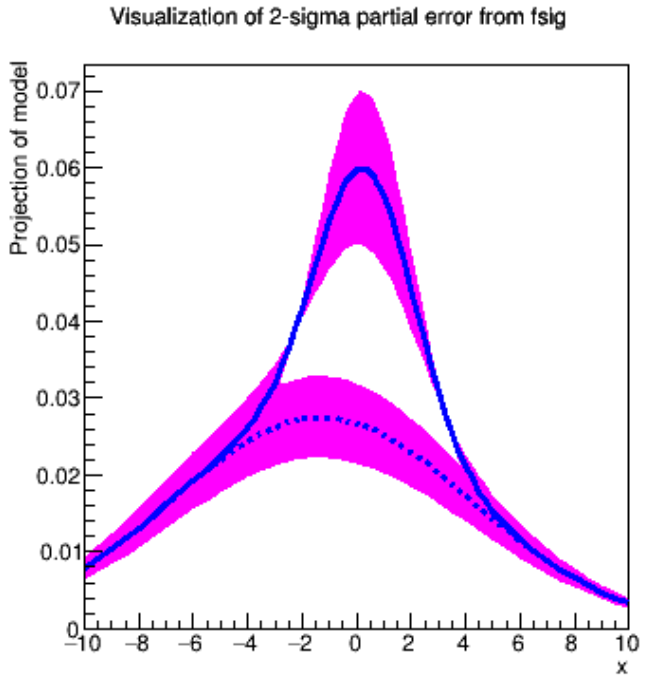
siguang@pku.edu.cn

```
// 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) ;
```



siguang@pku.edu.cn

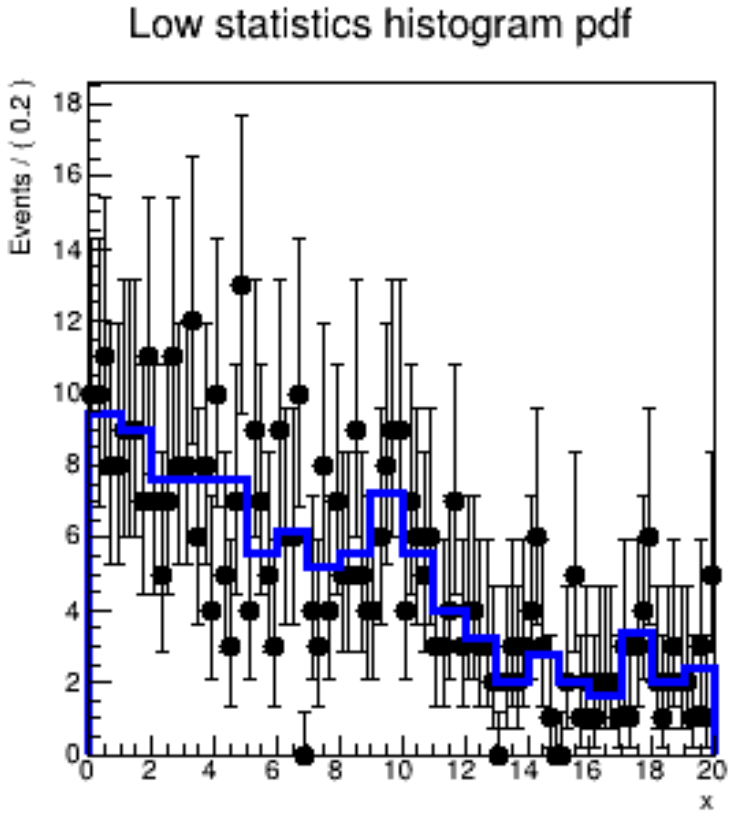
```
// Make plot frame
RooPlot* frame4 = x.frame(Bins(40),Title("Visualization of 2-sigma partial error from fsig")) ;

// 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) ;
```

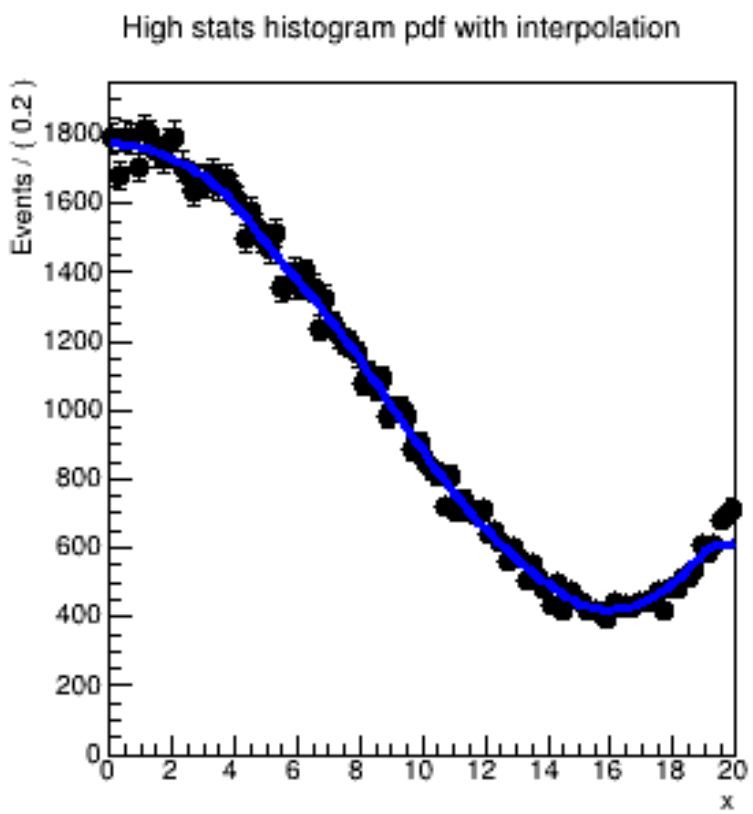


siguang@pku.edu.cn

```
  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() ;
}
```

$ROOTSYS/tutorials/roofit/rf706_histpdf.C





siguang@pku.edu.cn

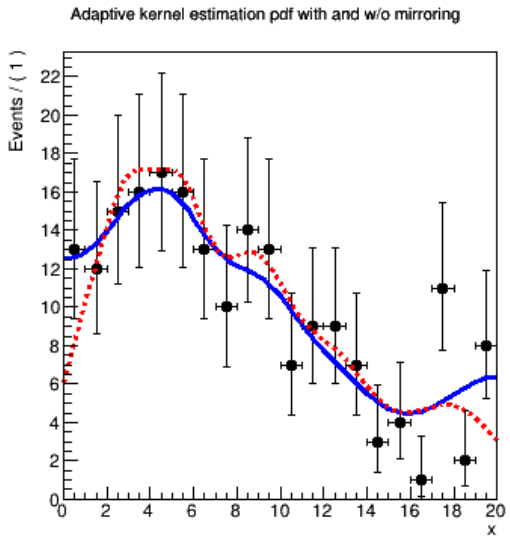
```
void rf706_histpdf()
{
 // C r e a t e   p d f   f o r   s a m p l i n g
 // ---------------------------------------

 RooRealVar x("x","x",0,20) ;
 RooPolynomial p("p","p",x,RooArgList(RooConst(0.01),RooConst(-0.01),RooConst(0.0004))) ;



 // C r e a t e   l o w   s t a t s   h i s t o g r a m
 // -----------------------------------------------

 // Sample 500 events from p
 x.setBins(20) ;
 RooDataSet* data1 = p.generate(x,500) ;

 // Create a binned dataset with 20 bins and 500 events
 RooDataHist* hist1 = data1->binnedClone() ;

 // Represent data in dh as pdf in x
 RooHistPdf histpdf1("histpdf1","histpdf1",x,*hist1,0) ;

 // Plot unbinned data and histogram pdf overlaid
 RooPlot* frame1 = x.frame(Title("Low statistics histogram pdf"),Bins(100)) ;
 data1->plotOn(frame1) ;
 histpdf1.plotOn(frame1) ;
```

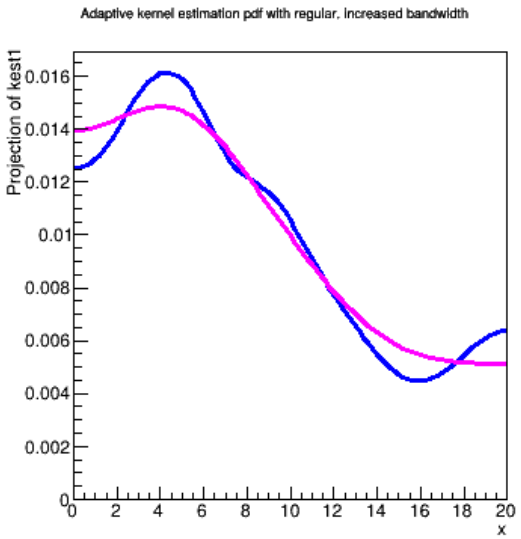
RooHistPdf (const char* name, const char* title, const RooArgSet& vars, const RooDataHist& dhist, Int_t intOrder = 0)

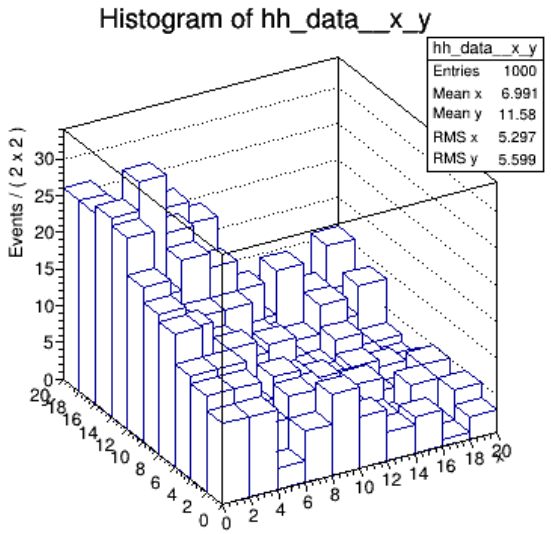
siguang@pku.edu.cn

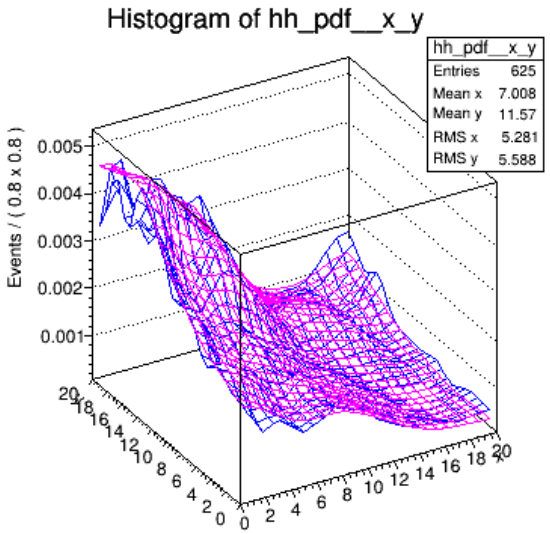
```
// C r e a t e   h i g h   s t a t s   h i s t o g r a m
// ---------------------------------------------------

// Sample 100000 events from p
x.setBins(10) ;
RooDataSet* data2 = p.generate(x,100000) ;

// Create a binned dataset with 10 bins and 100K events
RooDataHist* hist2 = data2->binnedClone() ;

// Represent data in dh as pdf in x, apply 2nd order interpolation
RooHistPdf histpdf2("histpdf2","histpdf2",x,*hist2,2) ;

// Plot unbinned data and histogram pdf overlaid
RooPlot* frame2 = x.frame(Title("High stats histogram pdf with interpolation"),Bins(100)) ;
data2->plotOn(frame2) ;
histpdf2.plotOn(frame2) ;


TCanvas* c = new TCanvas("rf706_histpdf","rf706_histpdf",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(1.8) ; frame2->Draw() ;

}
```

$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C







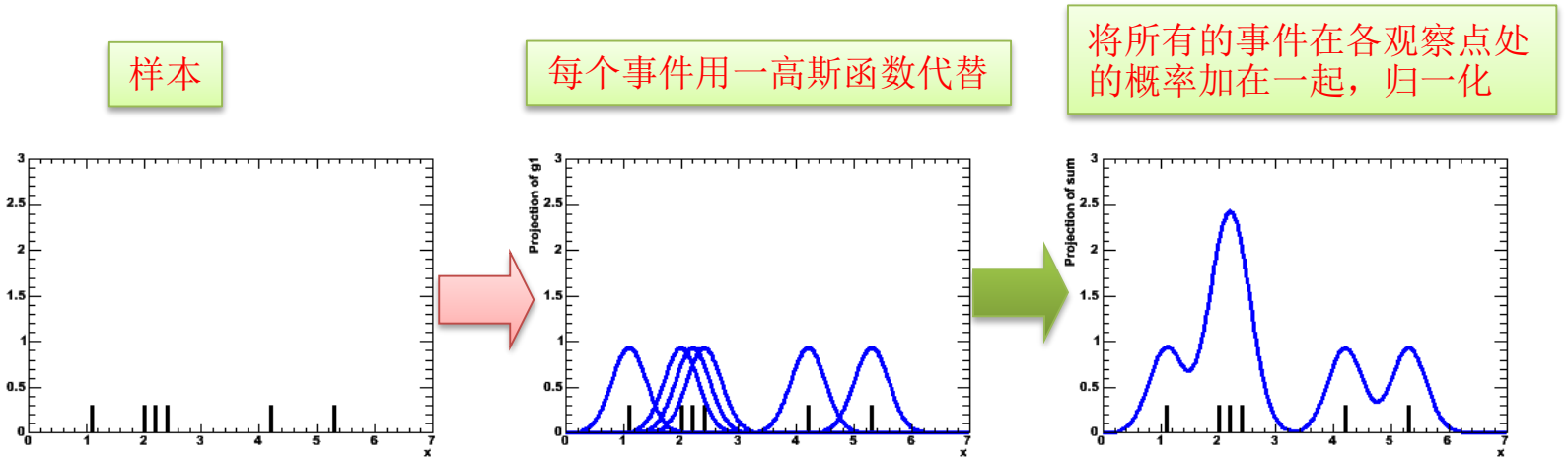


王思广
北京大学物理学院

siguang@pku.edu.cn

RooKeysPdf

- 用*unbinned* 数据
- 每个事件用一理想高斯分布代替
- 总的PDF分布：将所有的事件在各观察点处的概率加在一起，归一化

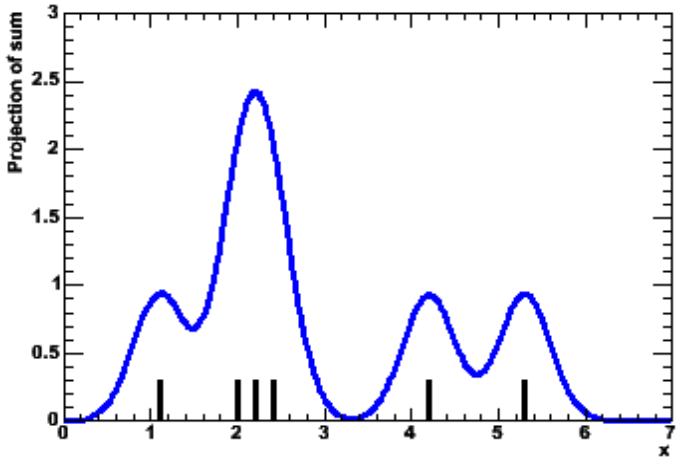


图形来自网络，下同

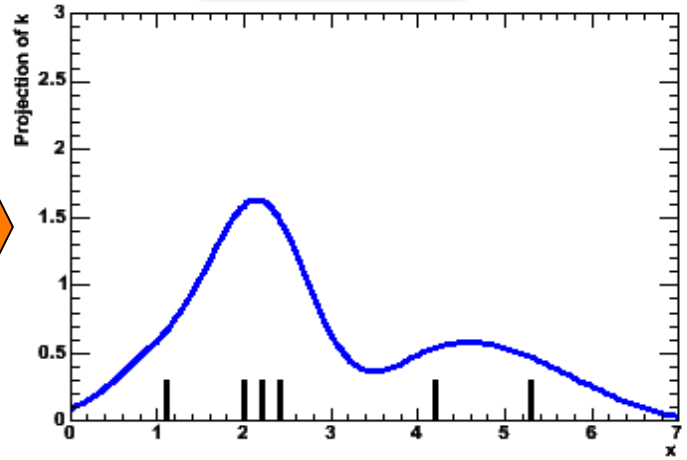
RooKeysPdf

- 只要每个事件的贡献为1/N，高斯函数的宽度可以相互之间有差异
- 自适应调节宽度技术
  - 对于低密度分布区高斯宽度加大，突出轮廓；
  - 高密度分布区域高斯函数宽度减小，突出细节；
  - 自动计算调节后的宽度

不进行调节宽度



自动调节宽度

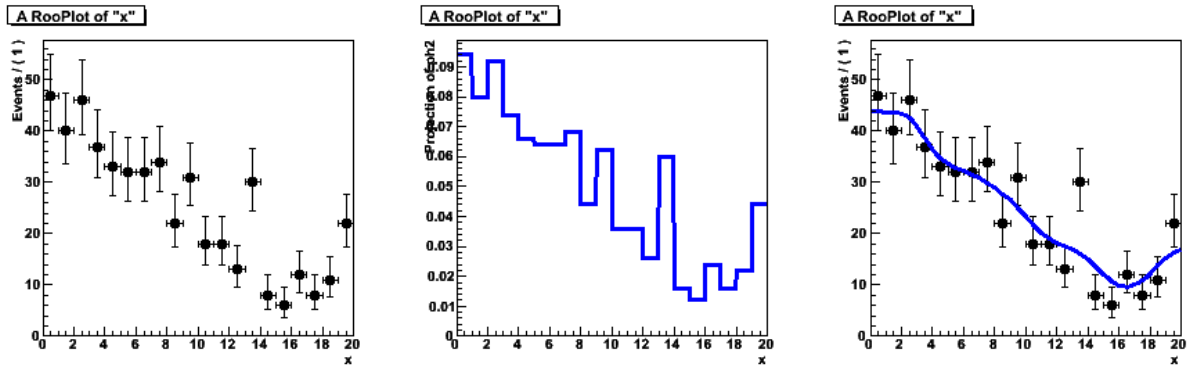


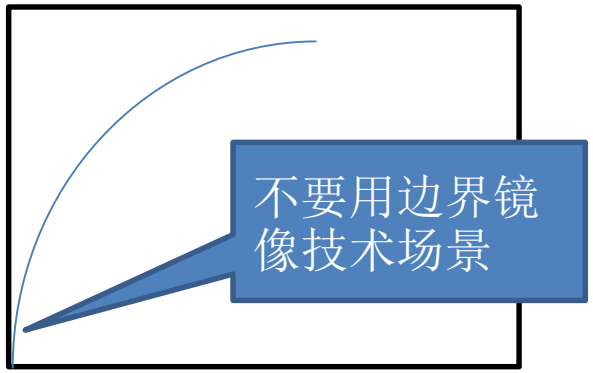
siguang@pku.edu.cn

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

- 与RooHistPdf相比较：
  - 低统计表现相当卓越
  - 边界镜像技术可减少边界泄露的影响，但不要用于本身在边界处趋于0且斜率大的情景
  - 可以用于多维(**RooNDKeysPdf**)

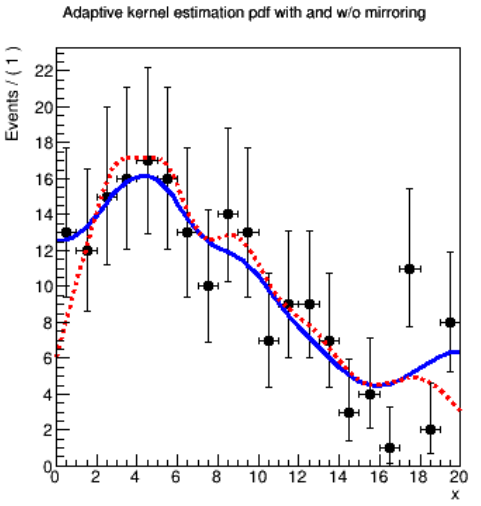
Data (N=500)　　RooHistPdf(data)　　RooKeysPdf(data)

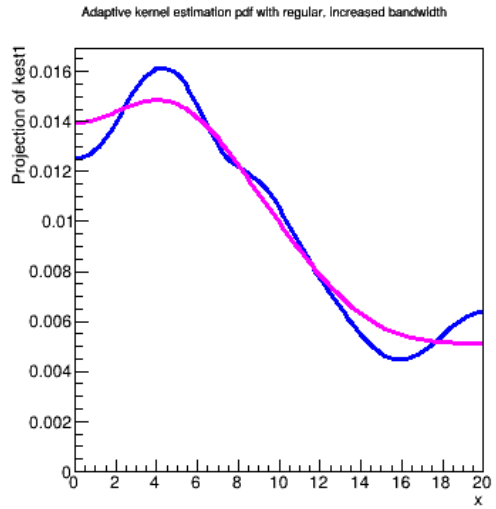




siguang@pku.edu.cn

```
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) ;
```






```
// 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)) ;
```

```
// C r e a t e   l o w   s t a t s   2 - D   d a t a s e t
// ---------------------------------------------------

// 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) ;

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

a: adaptive
m: mirroring

siguang@pku.edu.cn

```
// 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_pdf2->
Draw("surfsame") ;


}
```