root 的 fit

0. 路径

安装好root之后， 会在环境变量里面声明$ROOTSYS变量：

root的例程基本上都放在$ROOTSYS/tutorials/ 下面：

这次我们主要关注$ROOTSYS/tutorials/fit

可能的话，我们也看一下roofit目录

1. 用鼠标拟合

2. 用内置函数进行fit

比如说gaus， pol1. pol2。。。

```
   TCanvas *c1 = new TCanvas("c1");
   TH1F *h = new TH1F("h","My histogram",100,-3,3);
   h->FillRandom("gaus",6000);
   h->Fit("gaus");
   c1->Update();
```

```
void myfit1(){
  TH1F * hist = new TH1F ("hist", "hist", 10, -5,5);

  for (int i = 0; i<10; i++){
    for (int j = 0; j< i+1; j++){
      for (int k = 0; k<100; k++) hist->Fill(-5+0.5 +i*1);
    }
  }
  for (int i = 0; i<10; i++){
    hist->SetBinError(i+1, 1);
  }
  hist->Fit("pol1");
  hist->Draw("e");
}
```

3. 用内置函数定义TF1，再用TF1拟合

> 首先定义TF1，当然类似的方法也可以定义TF2

```
  TF1 f1("f1","pol1",0,10);
  f1.SetParameters(5,-0.5);
  TH1F h("background","linear background",100,0,10);
  h.FillRandom("f1",10000);
```

```
   TF1*  fM2D = new TF1("fM2D", "(1-[0]/(1+exp(([1]-x)/[2])))/(x*x)",
      xmin, xmax);
   TF1*  fM2N = new TF1("fM2N", "[0]/(1+exp(([1]-x)/[2]))/(x*x)",
      xmin, xmax);

   // First try: use a single set of parameters.
   // For each try, we need to find the overall normalization

   Double_t norm = 0.80;
   Double_t threshold = 25.0;
   Double_t width = 5.0;

   fM2D->SetParameter(0, norm);
   fM2D->SetParameter(1, threshold);
   fM2D->SetParameter(2, width);
   fM2N->SetParameter(0, norm);
   fM2N->SetParameter(1, threshold);
   fM2N->SetParameter(2, width);
   // fM2D->SetParameters(norm, threshold, width)


```

```
// Quadratic background function
Double_t background(Double_t *x, Double_t *par) {
   return par[0] + par[1]*x[0] + par[2]*x[0]*x[0];
}

// Lorenzian Peak function
Double_t lorentzianPeak(Double_t *x, Double_t *par) {
   return (0.5*par[0]*par[1]/TMath::Pi()) /
   TMath::Max( 1.e-10,(x[0]-par[2])*(x[0]-par[2]) + .25*par[1]*par[1]);
}

// Sum of background and peak function
Double_t fitFunction(Double_t *x, Double_t *par) {
  return background(x,par) + lorentzianPeak(x,&par[3]);
}



{
   // create a TF1 with the range from 0 to 3 and 6 parameters
   fitFcn = new TF1("fitFcn",fitFunction,0,3,6);
}
```

> 利用TF1产生hist

```
hM2D->FillRandom(fM2D->GetName(), nevtsD);
```

> 使用TF1拟合

```
histo->Fit(fitFcn);
```

4. 使用Miuit拟合（fitCircle.C）

```
//Generate points distributed with some errors around a circle
//Fit a circle through the points and draw
//To run the script, do, eg
//   root > .x fitCircle.C   (10000 points by default)
//   root > .x fitCircle.C(100);  (with only 100 points
//   root > .x fitCircle.C(100000);  with ACLIC
//
//Author: Rene Brun

#include "TCanvas.h"
#include "TRandom3.h"
#include "TGraph.h"
#include "TMath.h"
#include "TArc.h"
#include "TVirtualFitter.h"

TGraph *gr;

//______________________________________________________________________
void myfcn(Int_t &, Double_t *, Double_t &f, Double_t *par, Int_t) {
   //minimisation function computing the sum of squares of residuals
   Int_t np = gr->GetN();
   f = 0;
   Double_t *x = gr->GetX();
   Double_t *y = gr->GetY();
   for (Int_t i=0;i<np;i++) {
      Double_t u = x[i] - par[0];
      Double_t v = y[i] - par[1];
      Double_t dr = par[2] - TMath::Sqrt(u*u+v*v);
      f += dr*dr;
   }
}

//______________________________________________________________________
void fitCircle(Int_t n=10000) {
   //generates n points around a circle and fit them
   TCanvas *c1 = new TCanvas("c1","c1",600,600);
   c1->SetGrid();
   gr = new TGraph(n);
   if (n> 999) gr->SetMarkerStyle(1);
```

```
   else        gr->SetMarkerStyle(3);
   TRandom3 r;
   Double_t x,y;
   for (Int_t i=0;i<n;i++) {
      r.Circle(x,y,r.Gaus(4,0.3));
      gr->SetPoint(i,x,y);
   }
   c1->DrawFrame(-5,-5,5,5);
   gr->Draw("p");

   //Fit a circle to the graph points
   TVirtualFitter::SetDefaultFitter("Minuit");  //default is Minuit
   TVirtualFitter *fitter = TVirtualFitter::Fitter(0, 3);
   fitter->SetFCN(myfcn);

   fitter->SetParameter(0, "x0",   0, 0.1, 0,0);
   fitter->SetParameter(1, "y0",   0, 0.1, 0,0);
   fitter->SetParameter(2, "R",    1, 0.1, 0,0);

   Double_t arglist[1] = {0};
   fitter->ExecuteCommand("MIGRAD", arglist, 0);

   //Draw the circle on top of the points
   TArc *arc = new TArc(fitter->GetParameter(0),
      fitter->GetParameter(1),fitter->GetParameter(2));
   arc->SetLineColor(kRed);
   arc->SetLineWidth(4);
   arc->Draw();
}
```