粒子物理与核物理实验中的数据分析

张黎明
清华大学

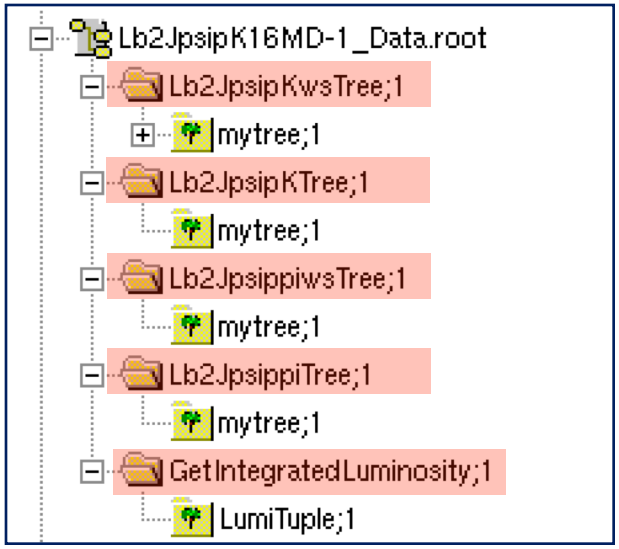
第十一讲：ROOT在数据分析中的应用(3)

本讲要点

- TTree和TChain的技巧
- 直方图的操作
  - 直方图的运算
  - 加减乘除： Add,Divide,...
  - 归一化：Scale
- ROOT中简单的拟合
  - 直方图拟合h1->Fit();
  - Graph的拟合
- 练习画一些测试统计量

ROOT文件中的子目录

```
//创建root文件，创建后默认目录就是在myfile.root目录中，
//实际上，root文件被当作一个目录。
TFile *fname=new TFile("myfile.root","recreate");

//gDirectory指向当前目录，即myfile.root
//可以调用mkdir函数在ROOT文件中创建一个子目录，如subdir
gDirectory->mkdir("subdir");

//进入到subdir子目录
gDirectory->cd("subdir");

//创建TTree
TTree *tree = new
    TTree("tree","tree in subdir");
.....
//将tree写到当前目录，即subdir中
tree->Write();
```



存在的Tree里添加新的Branch

```
void  tree3AddBranch() {
   TFile f("tree1.root" ,"update" );
   Float_t  new_v;
   TTree *t1 = (TTree*)f->Get("t1" );
   TBranch *newBranch = t1->
           Branch("new_v" ,&new_v,"new_v/F" );
//read the number of entries in the t1
   Int_t  nentries = (Int_t )t1->GetEntries();
   for  (Int_t  i = 0 ; i < nentries; i++){
      new_v= gRandom->Gaus(0 ,1 );
      newBranch->Fill();
   }
 t1->Write("" ,TObject::kOverwrite); // save only the new
 }
```

这里更新一个root，原始文件被覆盖，需要注意！

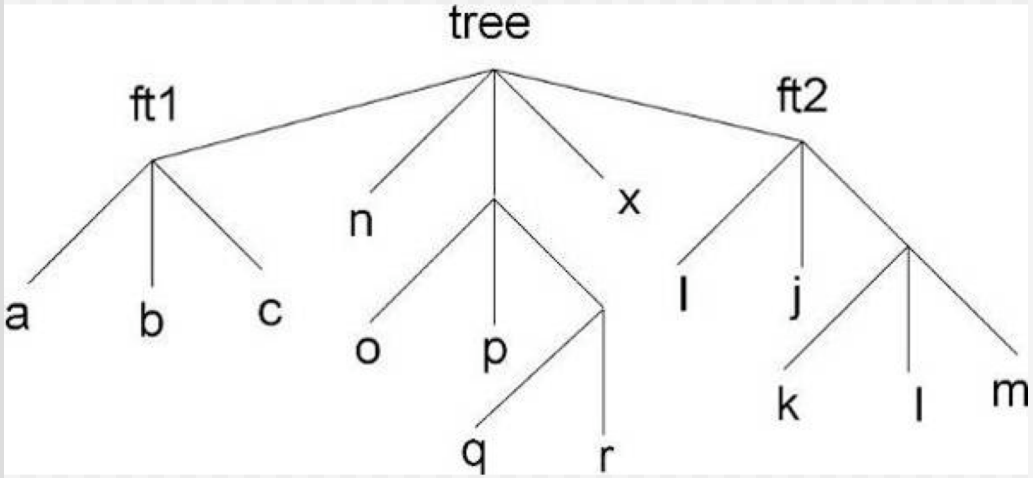
选择条件、并添加新的Branch

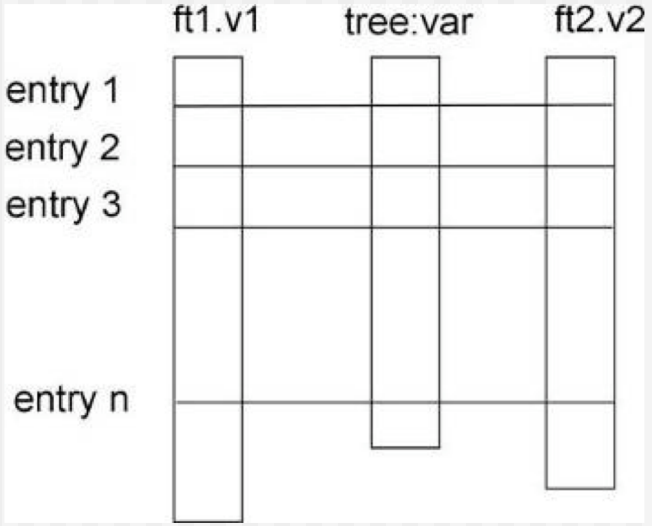
```
void  tree3AddBranch() {
   TFile f("tree1.root");
   Float_t  new_v;
   TTree *t1 = (TTree*)f->Get("t1" );
   t1->SetBranchStatus("*",1);
   TTree *h1 = t1->CloneTree(0);
   h1->Branch("new_v" ,&new_v,"new_v/F" );
//read the number of entries in the t1
   Int_t  nentries = (Int_t )t1->GetEntries();
   for  (Int_t  i = 0 ; i < nentries; i++){
      t1->GetEntry(jentry); //read from original tree t1
      new_v= gRandom->Gaus(0 ,1 );
 //can add selection here to only save passed events.
      if(!cut) continue;
      h1->Fill();
   }
   TFile f2("new.root","recreate");
   h1->Write(); // save only the new
 }
```

Save 新的tree！(事例很多时，出现问题？？)

如果只做cut，save，auto *h1 = (TTree*)t1->CopyTree(cut)

TTree::AddFriend





- 此方法可以让Tree访问友元Tree里的数据
- 比如：tree可以读取ft1,ft2里的变量
- 友元Tree的Entries个数必须>=原始Tree

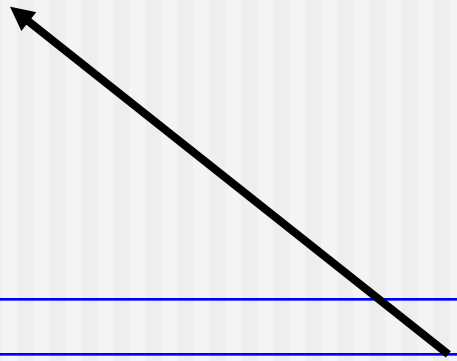
产生新的Tree和Tfile,利用AddFriend

```
void  tree1Addnew() {
   TFile f("tree1.root");
   TTree *t1 = (TTree*)f->Get("t1" );
   Int_t  nentries = (Int_t )t1->GetEntries();
   TFile f1("tree1f.root","recreate);
   TTree *t1f = new  TTree("t1f" ,"a friend Tree" );
   Float_t  new_v;
   t1f->Branch("new_v" ,&new_v,"new_v/F" );
   for  (Int_t  i = 0 ; i < nentries; i++){
       new_v= gRandom->Gaus(0 ,1 );
       t1f->Fill();
   }
  f1.cd();
  t1f->Write();
 }

void tree1r()
{
   …
   t1->AddFriend("t1f","tree1f.root");
   t1->Draw("t1f.new_v:px","t1f.new_v>0");
}
```

TChain: 分析多个root文件的利器(1)

TChain对象是包含相同tree的ROOT文件的列表。
参见手册TChain部分
http://root.cern.ch/root/html526/TChain.html

```
void exChain() {
 //定义TChain，t1为root文件中tree的名称!!!!!!!
 TChain* fChain= new TChain("t1");
 //添加所有文件至fChain，或根据需要添加部分root文件
 fChain->Add("Lec4/tree*.root");

 //画出t3的某个leaf，如ntrack
  fChain->Draw("px");
}
```



注意：此时，fChain等同于一个大root文件中的一个类"t1",该文件包含的事例数为所有文件中事例数之和（10^{12}以内）

TChain: 分析多个root文件的利器(2)

如果root文件中的类t3是在子目录subdir下，
则可以这样定义：

```
TChain* fChain= new TChain("subdir/t3");
```

或者将子目录和类的名字全部放在Add()函数中

```
TChain* fChain=new TChain();
fChain->Add("rootfiles/*.root/subdir/t3");
```

注意；从这里可以看出，

1) ROOT文件中的目录subdir与系统的子目录rootfiles同等地位;
2) ROOT文件的文件名在这里也类似于目录
3) TTree，即t3在这里也类似于目录

自动生成TTree的分析框架

- MakeClass,MakeSelector的运用

比如当前目录下有文件ex51.root，其中含有复杂的tree。可以用MakeClass或MakeSelector自动产生分析文件和头文件（非常方便）：

```
root [0] TFile f("ex51.root");
root [1] .ls
TFile**         ex51.root
 TFile*         ex51.root
  KEY: TTree    t4;1    Reconst events
root [2] t4->MakeClass();
或：          t4->MakeClass("AnaFrame");
```

自动产生以t4.h和t4.C文件，

或AnaFrame.h和AnaFrame.C文件。

类的定义以及Branch地址设定、分析框架都已经自动完成。

```
root [0] .L AnaFrame.C
root [1] AnaFrame t4
root [0] t4.Loop()
```

直接可以使用

MakeSelector的用法类似

分析框架的内容

重要的成员函数

```
virtual Int_t    Cut(Long64_t entry);
virtual Int_t    GetEntry(Long64_t entry);
virtual Long64_t LoadTree(Long64_t entry);
virtual void     Init(TTree *tree);
virtual void     Loop();
virtual Bool_t   Notify();
virtual void     Show(Long64_t entry = -1);
```

Loop的重要结构：

```
Long64_t nbytes = 0, nb = 0;
for (Long64_t jentry=0; jentry<nentries;jentry++) {
   Long64_t ientry = LoadTree(jentry);
   if (ientry < 0) break;
   nb = fChain->GetEntry(jentry);   nbytes += nb;
   // if (Cut(ientry) < 0) continue;
}
```

小结

- TTree的中添加Branch
- TChain分析含相同类结构的多个root文件
  chain->Add(...);
- 自动生成分析框架

直方图归一化(1)

https://root.cern.ch/doc/master/classTH1.html

直方图的归一化
void TH1::Scale(Double_t c1, Option_t *option)

把直方图每个区间的值(BinContent)乘以c1
假设　sum=h1->Integral()
h1->Scale(c1)之后，
　　h1->Integral()　　=　　c1*sum

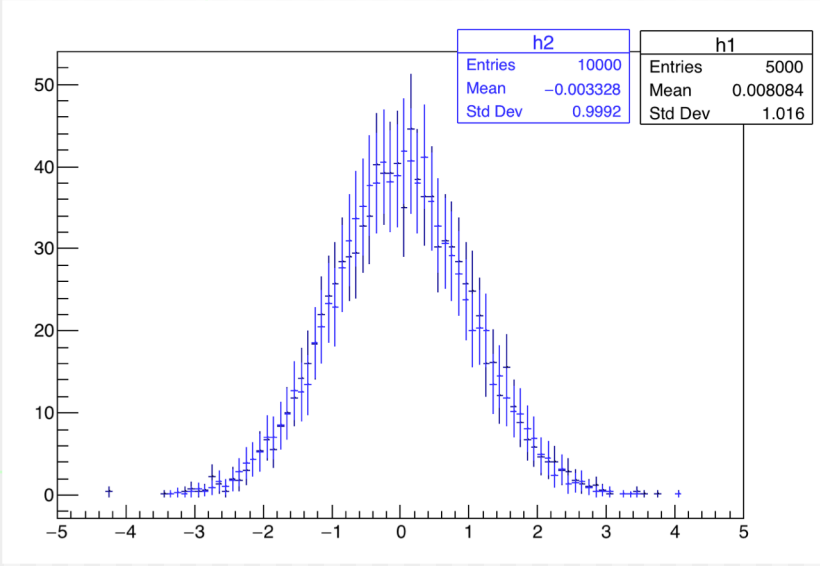
不加参数时，h1->Scale() 没有变化(默认c1=1)

"归一化"后，不仅BinContent变化了，BinError也变化了

直方图的归一化(2)

归一化常用于比较两种分布，找出区别。
所以，将2个直方图归一化到积分相同进行比较才直观。

```
root[0]TH1F *h1=new TH1F("h1","",100,-5,5);
root[1]TH1F *h2=new TH1F("h2","",100,-5,5);
root[2]h1->FillRandom("gaus",5000);
root[3]h2->FillRandom("gaus",10000);
root[4]float norm=1000;
root[5]h1->Scale(norm/h1->Integral());
root[6]h2->Scale(norm/h2->Integral());
root[7]h1->Draw("e");
root[8]h2->Draw("e,same") ;
```



注意Draw()函数的选项

"归一化"之后，h1或h2->Integral()=norm
在同一张图上可以看出比较2个分布的差别。

直方图四则运算(1)

重要提示：

1. 对直方图进行四则运算操作，一定要想明白运算的意义
   比如两个直方图的相加与两个随机变量的卷积有什么区别
2. 两个直方图的四则运算，区间大小和区间数相同才有意义
   四则运算"加减乘除"分别对应
   统计量(BinContent)的相加、相减、相乘、相除
3. 如果需要正确处理统计误差，需要在对ROOT脚本中
   调用TH1的某个静态成员函数，即
   TH1::SetDefaultSumw2();

不一定一直正确！

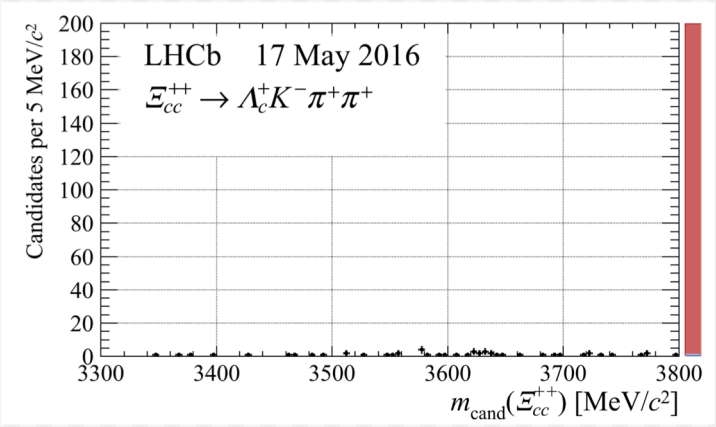
void SetDefaultSumw2(Bool_t sumw2 = kTRUE)
//static function. When this static function is called with
sumw2=kTRUE, all new histograms will automatically
activate the storage of the sum of squares of errors,
ie TH1::Sumw2 is automatically called.

直方图的四则运算(2)

相加：常用于相同实验的数据叠加，增加统计量。

```
root[1]TH1::SetDefaultSumw2();
root[1]TH1F *h3=new TH1F(*h1);
root[2]h3->Add(h1,h2,a,b);
```

结果：h3的BinContent被a*h1+b*h2替换，一般a=b=1



相减：常用于从实验测量的分布中扣除本底。

```
……
root[1]TH1F *h3=new TH1F(*h1);
root[2]h3->Sumw2();//也可在定义h3前TH1::SetDefaultSumw2();
root[3]h3->Add(h1,h2,a,-b);
```

结果：h3的BinContent被a*h1+b*h2替换，一般a=-b=1

误差：$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2} = \sqrt{n_1 + n_2}$ (假设h1和h2独立)

直方图的四则运算(3)

相除

常用于效率的计算。

```
root[1]TH1F *h3=new TH1F(*h1);
root[2]h3->Sumw2();
root[3]h3->Divide(h1,h2,a,b);
root[4]h3->Divide(h1,h2,a,b);
```

$$\sigma = \frac{n_1}{n_2}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}$$ (h1和h2独立)

思考：如果h1和h2不独立，怎么办？比如h1包含于h2

```
root[4]h3->Divide(h1,h2,a,b,"B");
```

$$\sigma = \sqrt{\frac{\frac{n_1}{n_2}(1-\frac{n_1}{n_2})}{n_2}}$$

二项分布误差

相乘

常用于对分布进行诸如效率等的修正。

```
root>TH1F *h3=new TH1F(*h1);
root>h3->Sumw2();
root>h3->Multiply(h1,h2,a,b);
```

$$\sigma = n_1 n_2\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}$$

直方图四则运算的误差处理

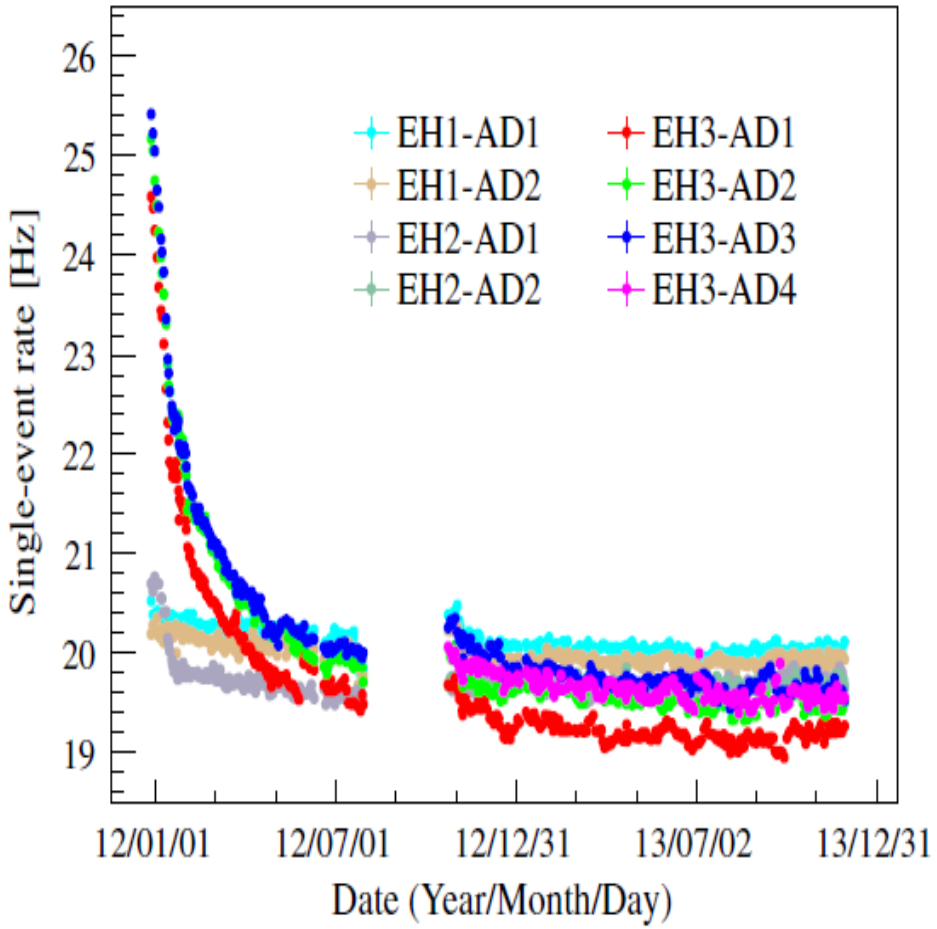
虽然ROOT都提供了较完善的一维直方图运算功能，但对最终结果的误差一定要仔细检查。很多情况下，用户需要从图中读出各频数数值与误差值，并确认运算无误。

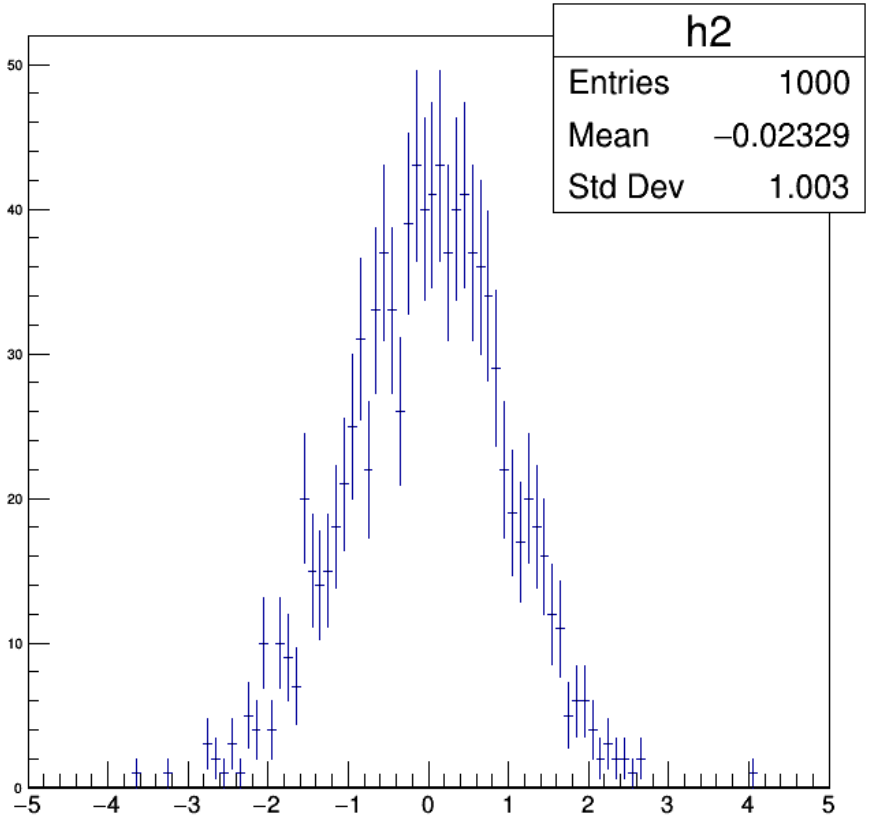
自己来进行更复杂的操作，或自己设定误差：

GetBinContent（nbin）
GetBinError（nbin）
SetBinContent（nbin,value）
SetBinError（nbin,error）

好多时候误差不是根号n那么简单！

好坏图之间的差别





字体大小，颜色，坐标轴含义，单位，图例，
额外的没用的信息

直方图画图的一些技巧

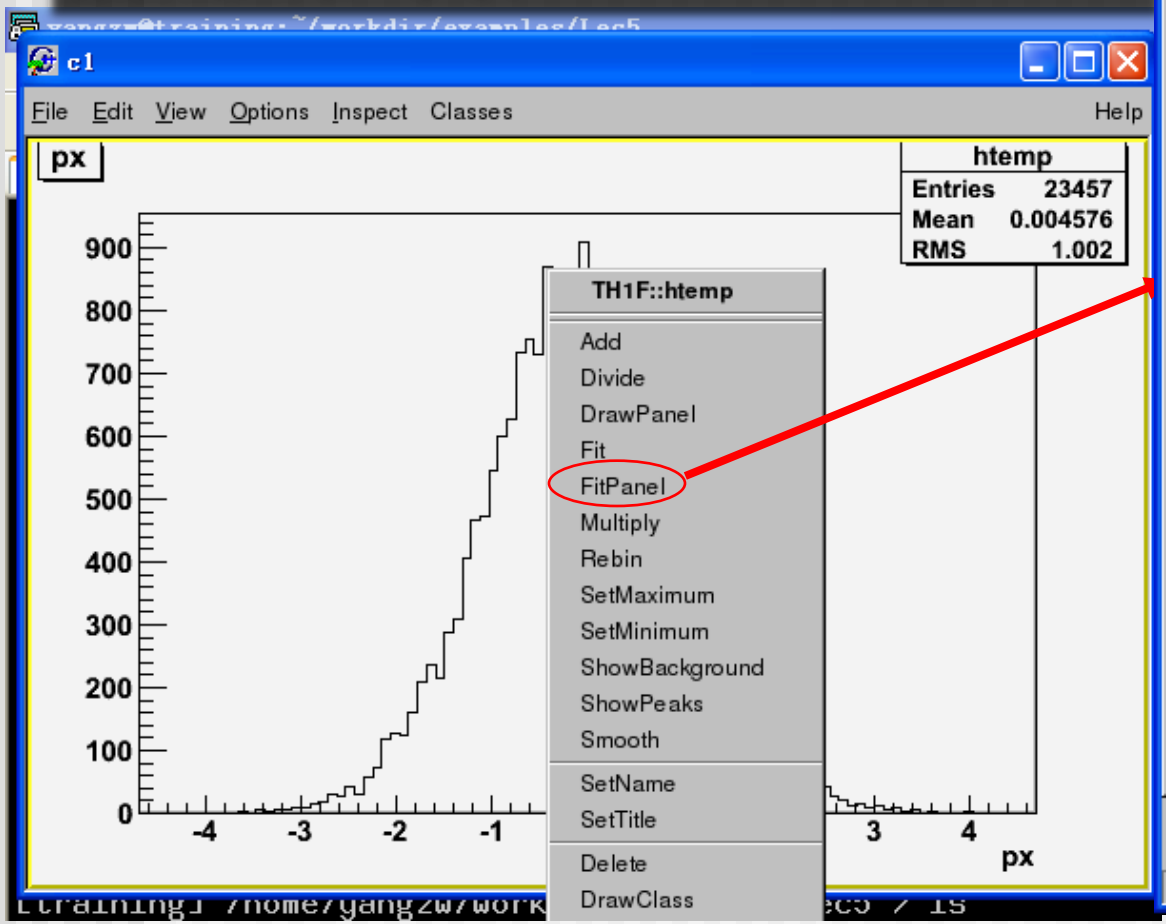
```
hSim->GetXaxis()->SetTitle("Visible Energy [MeV]");
hSim->GetYaxis()->SetTitle("Entries / 0.01 MeV");
hSim->GetXaxis()->SetRangeUser(0.1,20);
hSim->GetXaxis()->SetLableSize(0.05);
hsim->GetXaxis()->SetTitleSize(0.05);
hSim->GetXaxis()->SetTitleOffset(1);
hSim->SetTitle("Simulation and Fit Result");
hSig->SetLineColor(kRed);
hSig->SetLineWidth(2);
hSig->Rebin(10);
hSig->Draw("same");
TLegend lege(0.3,0.3,0.5,0.5,”leg”);
lege.AddEntry(hSim,”Sim”)
lege.Draw()
```

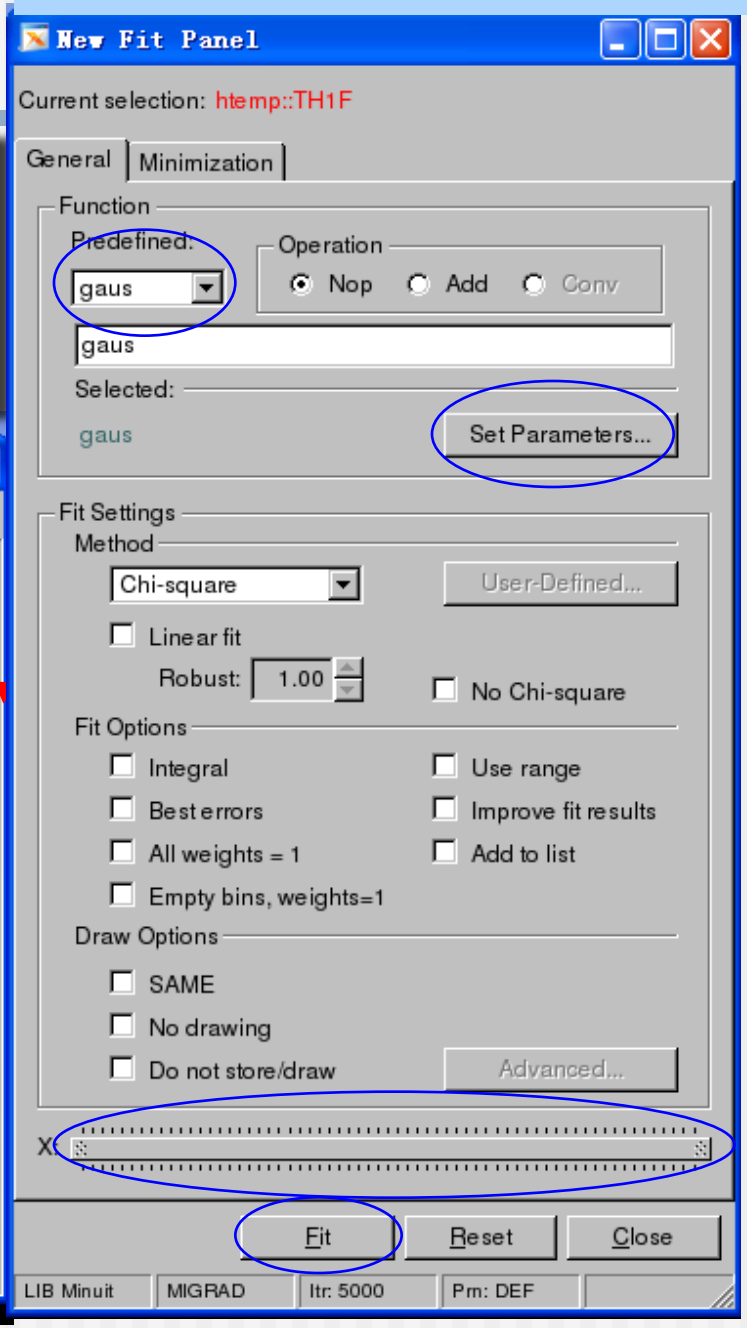
让直方图更清晰、明确、正确是你的责任，是一个合格博士生的标准。

拟合直方图(1)

将鼠标放到直方图上，右键，出现直方图操作选项，选择FitPanel，可以在FitPanel中选择拟合的各个选项，比如用什么函数拟合，拟合的区间，等等。

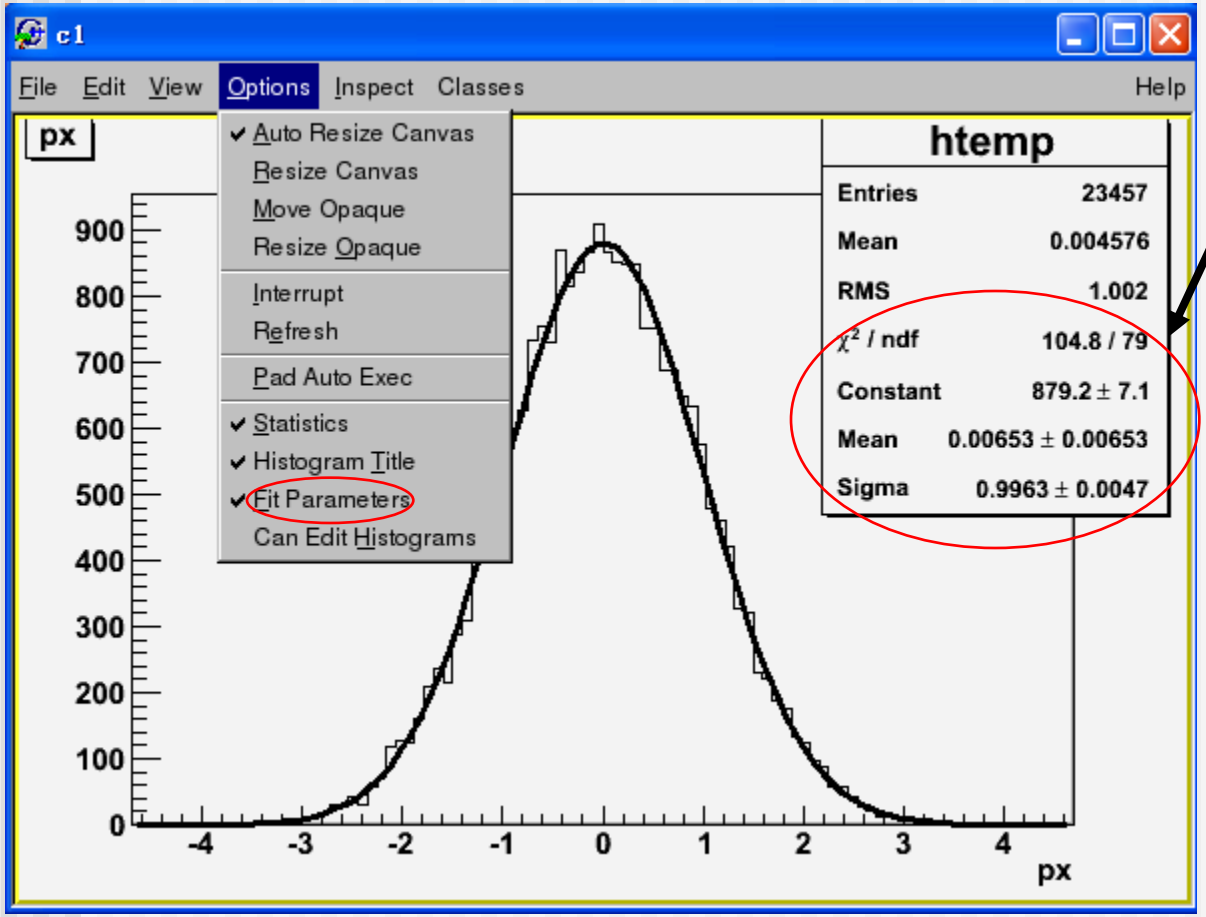
Context sensitive menu





拟合直方图(2)

用默认的高斯拟合，并在Options菜单中选上Fit Parameters选项，可以看到拟合的结果。



拟合结果给出了高斯分布的3个参数:常系数、均值、均方差，以及拟合的好坏chi2/ndf

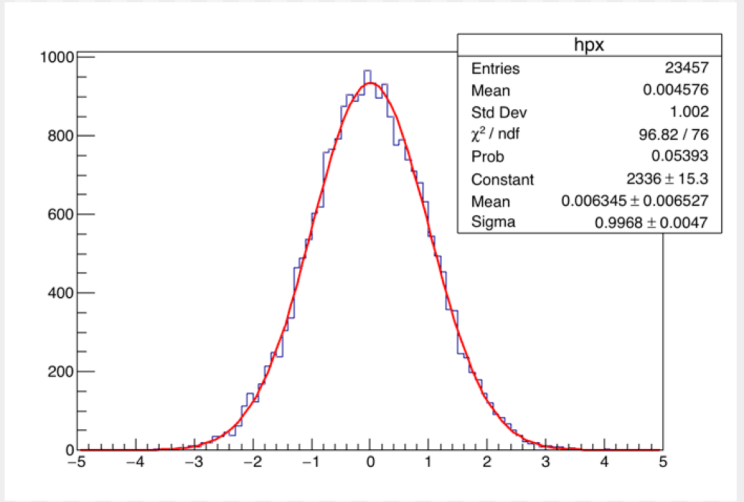
并不推荐这种拟合方式:

1)不适合自定义函数拟合

2)不适合批处理

拟合直方图(3) 简单函数

```
hpx->Fit("gausn");
hpx->Fit("gausn","","",-3,3);
TF1 *gaus = (TF1*)hpx->GetFunction("gausn");
if(gaus!=NULL) gaus->SetLineColor(2);
```



自定义拟合函数

```
TF1 *fcn = new TF1("fcn","gaus",-3,3);
hpx->Fit(fcn,"R");
```

"R" Use the Range specified in the function range

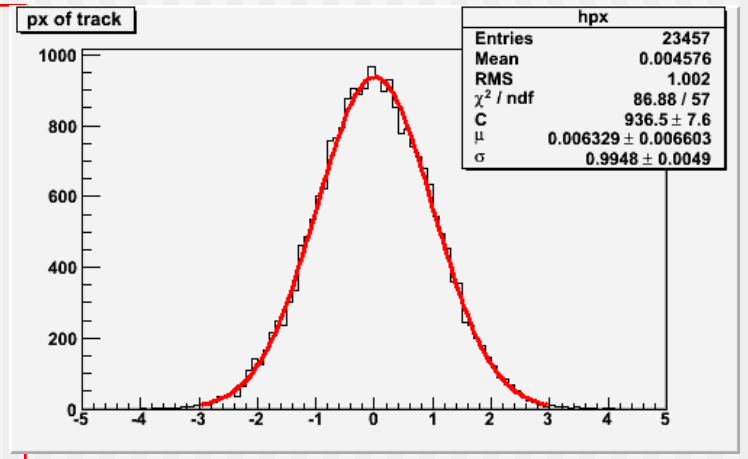
```
gStyle->SetOptFit();//设置拟合选项
```

拟合前往往需要给出合理的参数初值

```
fcn->SetParameters(500,mean,sigma);
```

拟合后取出拟合得到的参数

```
Double_t mypar[3];
fcn->GetParameters(&mypar[0]);
fcn->GetChisquare();
```



运行：root -l

root [0] .L ex51.C

用自定义的函数拟合直方图

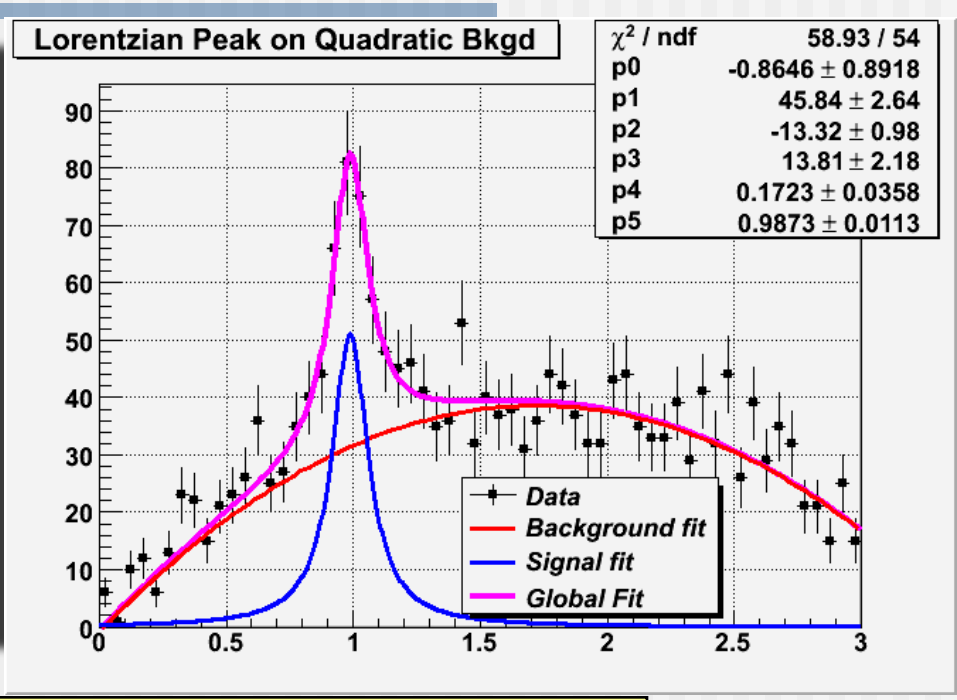
ROOT里预定义的函数

- gaus, expo, polN, ChebyshevN, landau, gausn

- "gaus" Gaussian function with 3 parameters: `f(x) = p0*exp(-0.5*((x-p1)/p2)^2)`
- "expo"An Exponential with 2 parameters: `f(x) = exp(p0+p1*x)`
- "pol*N*" A polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*x2 +...`
- "chebyshev*N*" A Chebyshev polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*(2*x2-1) +...`
- "landau" Landau function with mean and sigma. This function has been adapted from the CERNLIB routine G110 denlan (see TMath::Landau).
- "gausn Normalized form of the gaussian function with 3 parameters `f(x) = p0*exp(-0.5*((x-p1)/p2)^2)/(p2*sqrt(2PI))`

拟合直方图(3) 复杂函数
.../Lec5/ex52.C

共振峰(Breit-Wigner分布)加上二次函数本底的拟合(一共6个参数)
这是下面例子的简化版：
$ROOTSYS/tutorials/fit/FittingDemo.C
先自定义本底函数(background)和共振峰函数(lorentianPeak)，再定义这两个函数的和为拟合函数:fitFunction
利用fitFunction定义TF1函数




```
TF1 *fitFcn = new TF1("fitFcn",fitFunction,0,3,6);
```

这里指定函数区间为0-3，6个参数

```
fitFcn->SetParameter(4, 0.2); 为某个参数设初值(width)
fitFcn->SetParLimits(5, 0.6,1.4); 为某参数设置取值范围
```

运行：root -l
root [0] .L ex52.C

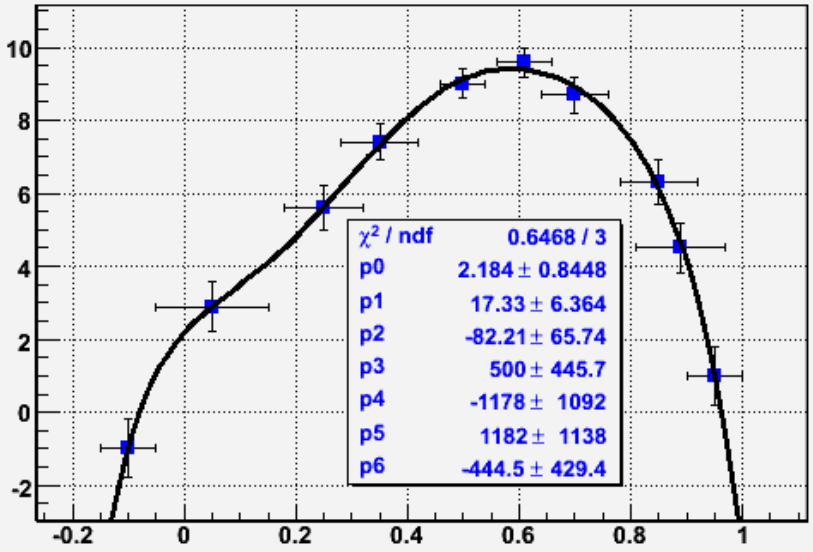
注意TLegend的使用

Chi2 Fit or Likelihood Fit

- TH1::Fit(TF1* f1, Option_t* option="", ...)
- option =
  R - Use the Range specified in the function range
  L - Use Loglikelihood method (default is chisquare method)
  For low statistics histogram, this is a must-do.

拟合任意图形

当直方图的形式已经不能满足你的需求，怎么处理任意测量结果呢？




```
const Int_t n1 = 10;
Double_t x1[]  = {-0.1,0.05,0.25,0.3
Double_t y1[]  = {-1,2.9,5.6,7.4,9,9.
Double_t ex1[] = {.05,.1,.07,.07,.04,
Double_t ey1[] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
TGraphErrors *gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
gr1->Fit("pol6","q");
gr1->Draw("Ap");
```

生成一个TGraph：定义它的数据量，各个数据点的值和误差

用一个6次多项式拟合

画出来，一定要加Ap

作业及课后自行体会内容，反馈

- 产生1000个有100个事例的高斯样本， 其中要求μ=0，σ=1

1. 用高斯函数来拟合，全部参数自由，记录每次的拟合结果，μ，σ，chi2/ndf
   - 画上述三个变量的分布
2. 仍用高斯函数拟合，固定μ=0.05，σ自由，记录得到的σ，chi2/ndf
   - 画上述两个变量的分布

- 对比上述两种拟合的σ，chi2/ndf的分布，画在一起

设置单个文件大小，自动写入新文件

- 设置文件大小要求(default 100G)
- TTree::SetMaxTreeSize(int size);
- 例子：aTree->SetMaxTreeSize(500000000); //500M, otherwise split into a new file, a new file with a name of the style "file_1.root" if the original requested file name was "file.root".
- 关闭文件
- TTree::GetCurrentFile(); //get the pointer to the current file
- OutFile->Close();