中国科学技术大学
University of Science and Technology of China

ROOT的基本使用

By God Bo

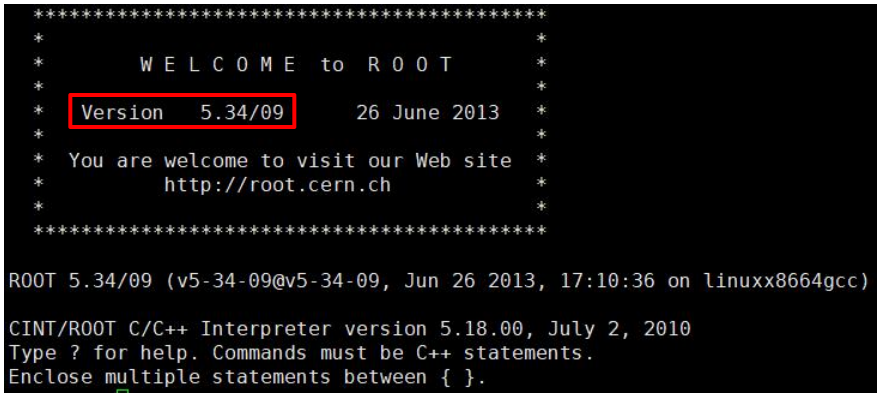
- Start:

$ root Start the ROOT
$ root –l Does not show the ROOT banner
root [0] .q quit the ROOT
root [1] .x <file_name> .x name.cxx (execute)
root [2] .L <file_name> .L name.cxx (load)

$ root –l name.root open file
root [0] .ls look list
root [1] treename->Print() list all branches in this tree
root [2] treename->Draw("branch") draw branch
root [3] treename->Scan() print message

- Calculator

root [0] 1+1
root [1] 2*(50+91)/12.
root [2] sqrt(3.)
root [3] TMath::Pi()



Creating ROOT macros

The name of the ROOT macro and the file name (without file extension) in which the macro is saved must match.

1. Create a new file in your preferred text editor.
2. Use the following general structure for the ROOT macro, preferably with a function that has the same name as the file:

```c++
void MacroName() {
  ...

  your lines of C++ code
  code line ends with ;
  ...
}
```

1. Save the file ROOT macro, using the macro name as file name: **MacroName**.C

> ***Note***
>
> *It's not necessary to* `#include` *anything in the ROOT macros. Everything in the include paths is automatically included. Note that you can type* `.I` *in the ROOT prompt to see the include paths, and* `.I [path]` *to add an extra path.*

Sample Program

You can find sample program of next few pages in

/scratchfs/bes/wwangbo/draw/draw/ (ihepc)
/ustcfs/BES3User/2021/bwang/draw/ (ustc)

draw1.cxx and draw2.cxx are used to draw figures
opetree.cxx is used to modify tree in the root file

draw1.cxx

读文件 建立直方图

```
void draw1(){

        gStyle->SetOptStat(0);
        //information in statistics via parameter mode

        TFile *File = new TFile("../rhopi3097_001.root");
        //read file
        TTree *Tree = (TTree*)File->Get("fit5c");
        //read tree in the file
        TFile *File1 = new TFile("../rhopi3097_002.root");
        TTree *Tree1 = (TTree*)File1->Get("fit5c");
        TFile *File2 = new TFile("../rhopi3097_003.root");
        TTree *Tree2 = (TTree*)File2->Get("fit5c");
        TFile *File3 = new TFile("../rhopi3097_004.root");
        TTree *Tree3 = (TTree*)File3->Get("fit5c");

        TCut cut = "";
        //selection
        //cut1+cut2=(cut1)&&(cut2)      cut1||cut2=(cut1)||(cut2)

        double bin(100), lo(0), up(3);
        //bins and the range
        TString val = "mrh0";
        //name of variable

        TH1F *h = new TH1F("h", "", bin, lo, up);
        //H for histogram, F for float
        TH1F *H[5];
        //Hist in array form
        for(int i=0; i<5 ;i++)
        {
```

输入需要绘图的root文件

输入root中待分析的tree

创建一个新的直方图，bin指的是区间数，lo,up指的是上下限

draw1.cxx

建立画布 将变量加进直方图

```
TH1F *H[5];
//Hist in array form
for(int i=0; i<5 ;i++)
{
H[i] = new TH1F(Form("H%d", i), "", bin, lo, up);
}

TCanvas *c = new TCanvas("c","c",10,10,800,600);
//define a canvas
Tree->Draw(val+">>h",cut);
//consider as signal
Tree1->Draw(val+">>H0",cut);
Tree2->Draw(val+">>H1",cut);
Tree3->Draw(val+">>H2",cut);
//consider as data

h->SetYTitle("Events");
h->SetXTitle("M_{#rho^{0}}");
h->GetYaxis()->SetLabelSize(0.043);
//size of label of Yaxis
h->GetYaxis()->SetRangeUser(0,10000);
//range of Yaxis

h->SetLineWidth(3);                    //TAttLine Class
h->SetLineColor(kRed);
h->Scale(2);                  //scale

H[2]->SetMarkerSize(0.9);                          //TAttMarker Class
H[2]->SetMarkerStyle(8);
H[2]->SetLineWidth(2);
H[2]->Scale(2*h->GetEntries()/H[2]->GetEntries());
```

将Tree中的mrho画在画布上

设置图像的具体细节，比如线的粗细，点的大小，坐标轴名称之类的

draw1.cxx

设置堆积图 一般用来描述本底

```
THStack *hs = new THStack("hs","");
H[0]->SetLineColor(kOrange);
H[0]->SetLineWidth(2);
H[0]->SetFillColor(kOrange);
H[0]->SetFillStyle(3344);
H[1]->SetLineColor(kBlue);
H[1]->SetLineWidth(2);
H[1]->SetFillColor(kBlue);
H[1]->SetFillStyle(3344);
hs->Add(H[0]);
//add hist
hs->Add(H[1]);
```

画出直方图

```
h->Draw("hist");
hs->Draw("same");
H[2]->Draw("same hist ep");
//e for error, p for point
```

draw1.cxx


```
TLegend *leg = new TLegend(0.6, 0.7, 0.85, 0.85);
//legend
leg->SetFillColor(0);
leg->SetNColumns(2);
//2 columns
leg->SetBorderSize(0);
leg->AddEntry(h,"signal","l");
//l for line, f for fill, e for error, p for point
leg->AddEntry(H[2],"data","ep");
leg->AddEntry(H[0],"back1","f");
leg->AddEntry(H[1],"back2","f");
leg->Draw();
```

设置图例

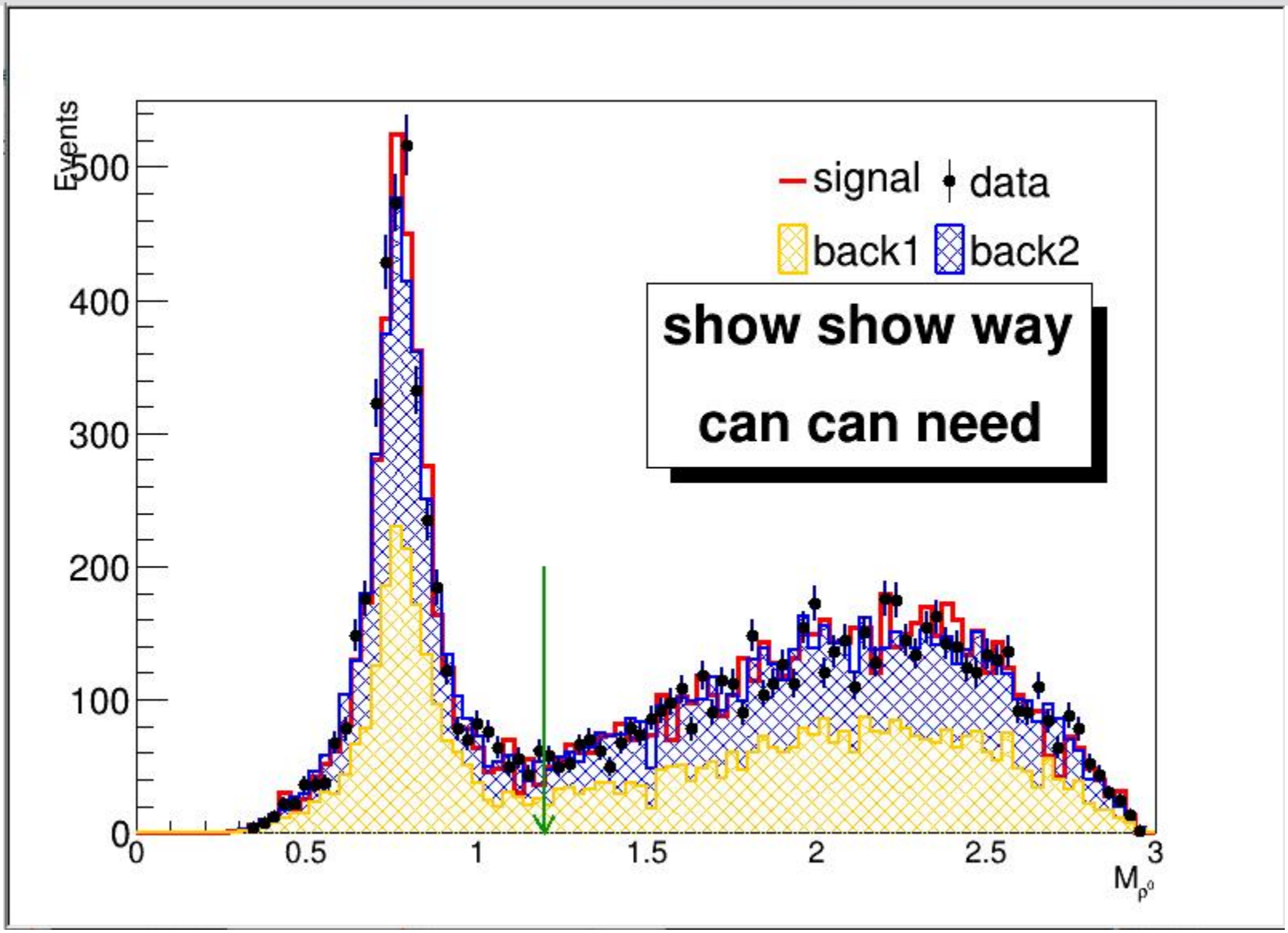
```
TArrow *arrow = new TArrow(1.2,200,1.2,0,0.02,">");
//TArrow Class
arrow->SetFillColor(1);
arrow->SetFillStyle(1001);
arrow->SetLineColor(kGreen+2);
arrow->SetLineWidth(2);
arrow->Draw();
```

设置箭头

```
TPaveText *pt = new TPaveText(0.5,0.5,0.85,0.7,"BRNDC");
pt->SetFillColor(10);
pt->SetBorderSize(10);
pt->SetTextAlign(22);
TString par1 = "show show way";
TString par2 = "can can need";
```

设置文本框

draw1.cxx


Events
500
400
300
200
100
0
0
0.5
1
1.5
2
2.5
3
M_{ρ^0}
signal
data
back1
back2
show show way
can can need

draw2.cxx

另一种读取root的方式

```
TChain *chain = new TChain("fit5c");
chain->Add("../rhopi3097_001.root");
chain->Add("../rhopi3097_005.root");
```

这样可以一次性读取多个root文件

```
double m1,m2,m3,m4,chisq;
chain->SetBranchAddress("mrh0",&m1);
chain->SetBranchAddress("chi2",&chisq);
Tree1->SetBranchAddress("mrh0",&m2);
Tree2->SetBranchAddress("mrh0",&m3);
Tree3->SetBranchAddress("mrh0",&m4);

for(int i=0; i<chain->GetEntries(); i++){
        chain->GetEntry(i);
        /*if(chisq<50)*/ sig->Fill(m1);
}
for(int i=0; i<Tree1->GetEntries(); i++){
        Tree1->GetEntry(i);
        data->Fill(m2);
}
for(int i=0; i<Tree2->GetEntries(); i++){
        Tree2->GetEntry(i);
        back1->Fill(m3);
}
for(int i=0; i<Tree3->GetEntries(); i++){
        Tree3->GetEntry(i);
        back2->Fill(m4);
```

另一种将变量填入直方图的方式，这里的data,back1,2啥的都是前面的定义过的直方图

draw2.cxx

将画布分为两部分，打开第一部分（内容与**draw1.cxx**类似）

```
TCanvas *c = new TCanvas("c","c",10,10,1200,500);
c->Divide(2,1);
c->cd(1);
```

```
c->cd(2);
double x[10], y[10];
for(int i=0; i<10; i++){
        x[i] = i+1;
        y[i] = sin(x[i])/x[i]+0.1;
}
TGraph *gr = new TGraph(10,x,y);

gr->SetLineWidth(3);
gr->SetLineColor(2);
gr->SetMarkerStyle(4);
gr->SetMarkerColor(3);

gr->GetYaxis()->SetRangeUser(-0.3,1.2);

gr->Draw("ACP");
```

打开第二部分，使用已知数组画图，这里等于画了一个函数的散点图。

draw2.cxx

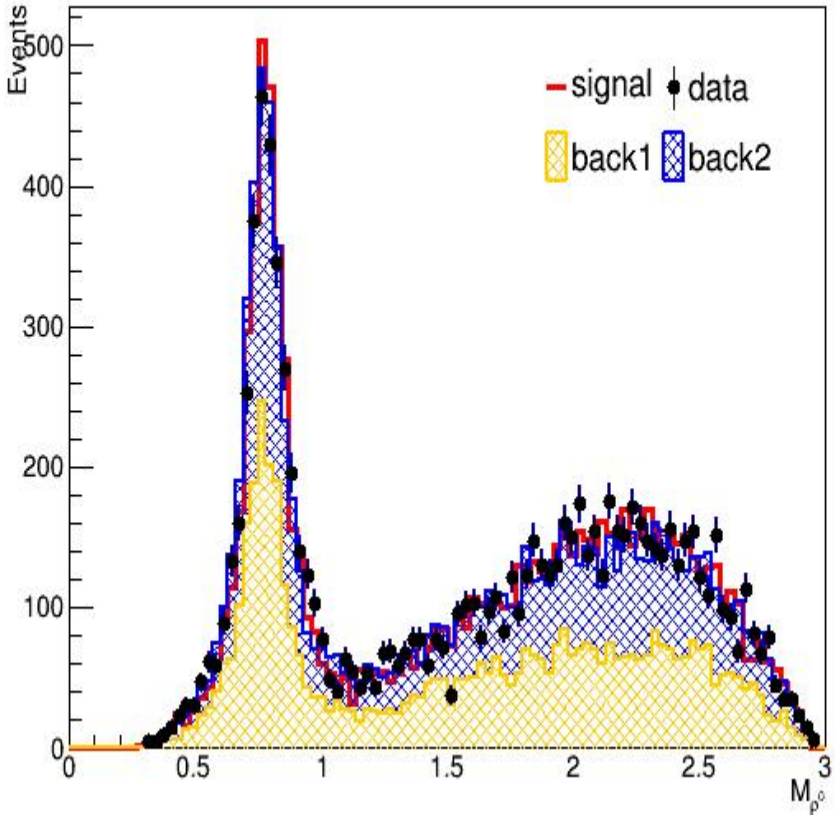
使用公式画图与超简单拟合

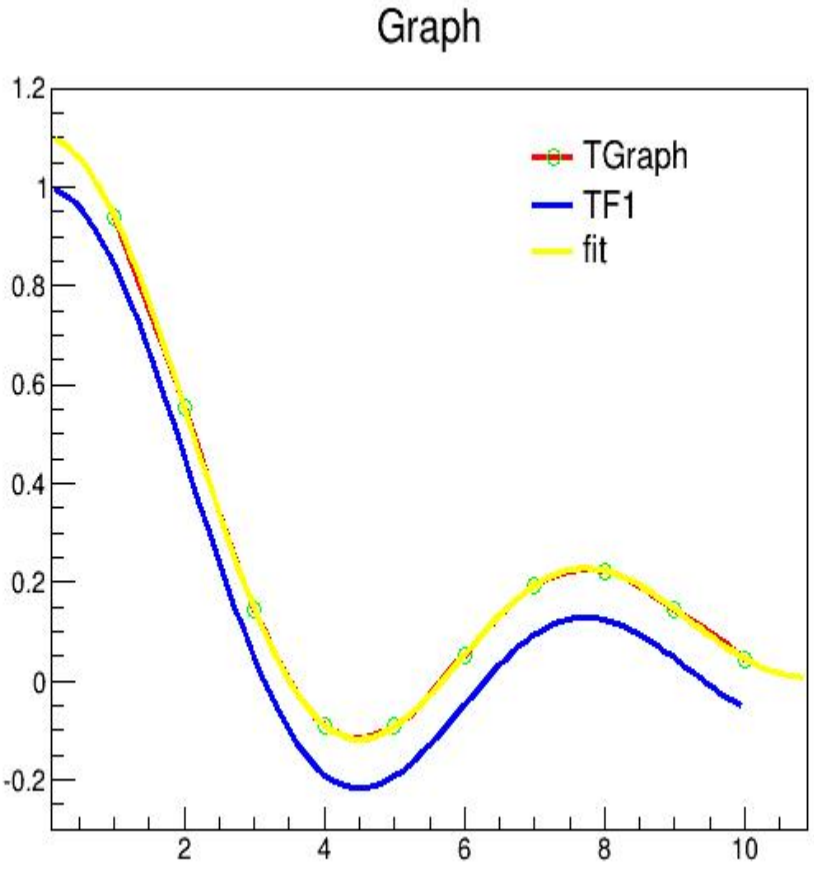
```
TF1 *f = new TF1("fun", "sin(x)/x", 0, 10);
f->SetLineWidth(3);
f->SetLineColor(4);
f->Draw("same");

TF1 *f_fit = new TF1("f_fit", "sin(x)/x+[0]",0,10);
f_fit->SetLineWidth(3);
f_fit->SetLineColor(5);
f_fit->SetParameter(0,1);
gr->Fit("f_fit");
f_fit->Draw("same");
```

新定义一个函数绘制其连续图像

对刚刚绘制的函数散点图进行拟合

draw2.cxx


Events
signal
data
back1
back2
0
100
200
300
400
500
0
0.5
1
1.5
2
2.5
3
M_{ρ^0}


Graph
TGraph
TF1
fit
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
2
4
6
8
10

opeTree.cxx

读取文件和tree

```
TFile *infile = new TFile("../rhopi3097_010.root");
TTree *intree1 = (TTree*)infile->Get("vxyz");
TTree *intree2 = (TTree*)infile->Get("photon");
TTree *intree3 = (TTree*)infile->Get("dedx");
TTree *intree4 = (TTree*)infile->Get("tof1");
TTree *intree5 = (TTree*)infile->Get("tof2");
TTree *intree6 = (TTree*)infile->Get("pid");
TTree *intree7 = (TTree*)infile->Get("etot");
TTree *intree8 = (TTree*)infile->Get("fit4c");
TTree *intree9 = (TTree*)infile->Get("fit5c");
TTree *intree10 = (TTree*)infile->Get("geff");
TTree *intree11 = (TTree*)infile->Get("tofe");
```

opeTree.cxx

综合性操作 加减branch 添加cut条件

```
//fit5c
double chisq, mrh0_1, mrhp_1, mrhm_1;
intree9->SetBranchAddress("mrh0",&mrh0_1);
//read branch
intree9->SetBranchAddress("mrhp",&mrhp_1);
intree9->SetBranchAddress("chi2",&chisq);
intree9->SetBranchStatus("mrhm",0);
//0 means deactive;  don't clone this branch while cloning in tree intree9

TFile *newfile = new TFile("../test.root", "recreate");
TTree *newfit5c = intree9->CloneTree(0);
//clone without value in the branch

double sum;
TBranch *Br1 = newfit5c->Branch("sum",&sum);
//built a new branch of tree newfit5c

for(int i=0; i<intree9->GetEntries(); i++)
{
        intree9->GetEntry(i);
        sum = mrh0_1+mrhp_1;                    //define the new variable
        if(chisq<50) newfit5c->Fill();                  //add cut
}
newfit5c->Print();
//newfit5c->Write();
```

opeTree.cxx


```
//Copy tree
intree1->SetBranchStatus("*", 0);
//all deactive
intree1->SetBranchStatus("vx0", 1);
//1 mean active

TTree *newtree1 = intree1->CloneTree();
//clone all without fill
newtree1->Print();
//newtree1->Write();


//Cut tree
TTree *newfit4c = intree8->CopyTree("chi2<50");
newfit4c->Print();
//newfit4c->Write();


newfile->Write();
//write the file
//delete newfile;
```

复制tree

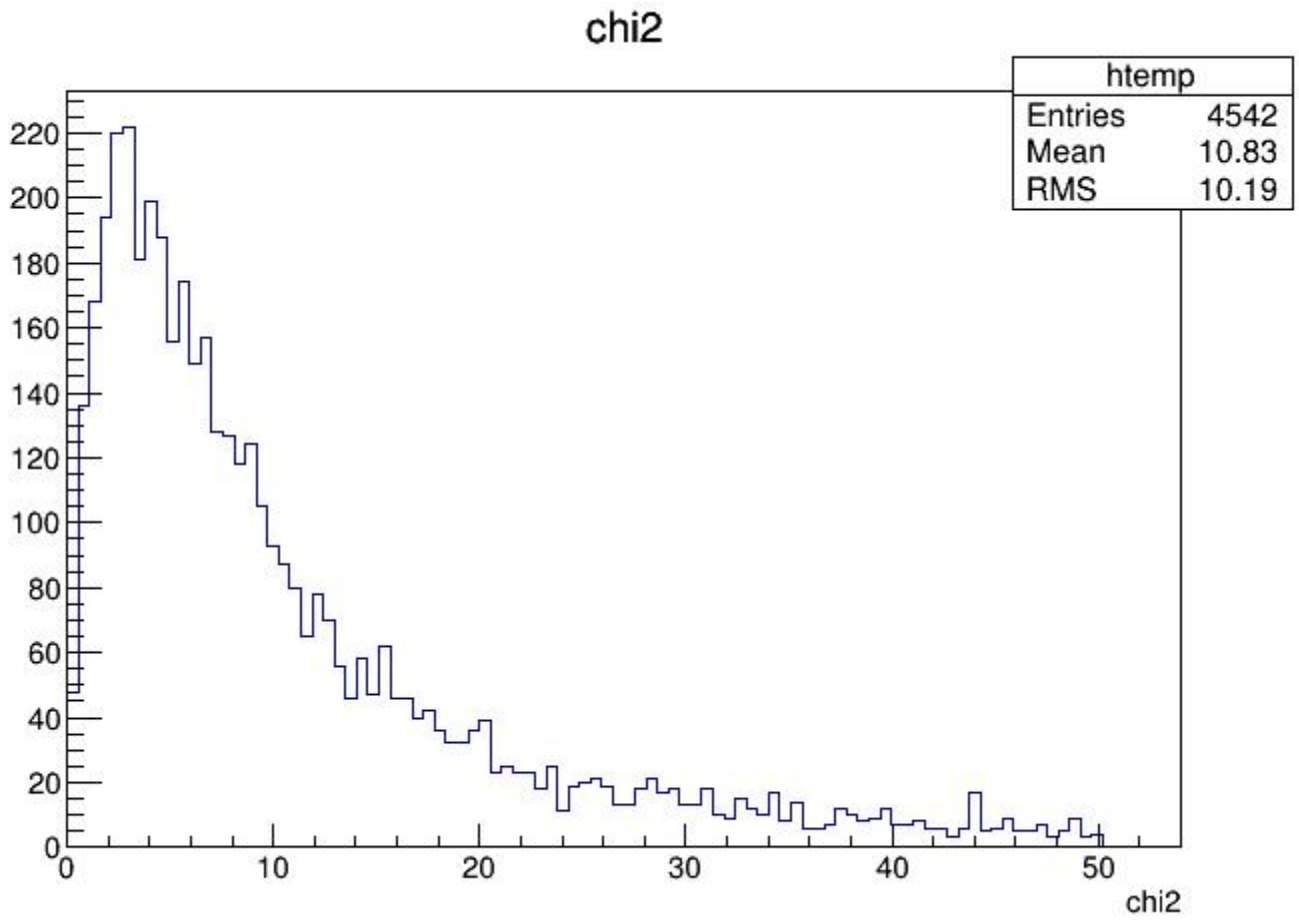
添加cut条件

写入新文件

opeTree.cxx


```
[wangyijing@lxslc705 ~/draw]$ root -l test.root
^[[Aroot [0]
Attaching file test.root as _file0...
root [1] .ls
TFile**         test.root
 TFile*         test.root
  KEY: TTree    fit5c;1 ks N-Tuple example
  KEY: TTree    vxyz;1  ks N-Tuple example
  KEY: TTree    fit4c;1 ks N-Tuple example
root [2]
```

opeTree.cxx


chi2
htemp
Entries 4542
Mean 10.83
RMS 10.19
220
200
180
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
chi2