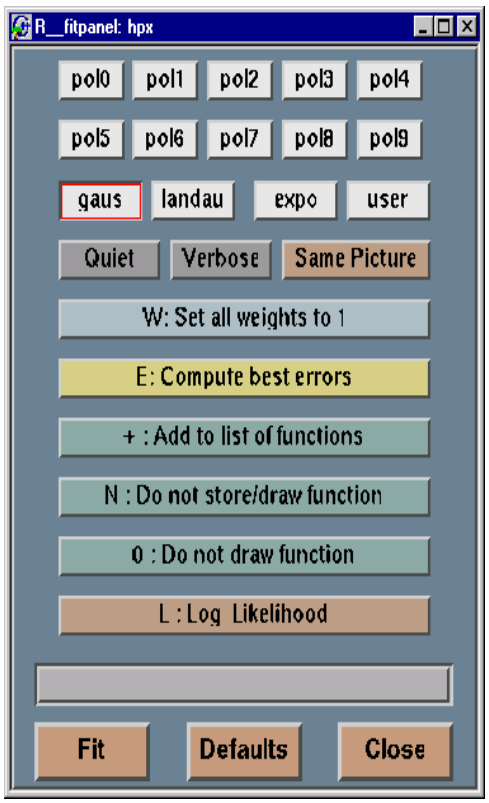
Histogram Fitting for Dummies

I) **Introduction**.

Histograms and their interpretations are central to particle physics data analysis, so it is not surprising that ROOT has a sophisticated built-in histogram fitting capability. There are 2 out of 22 chapters in the ROOT manual about histograms, and one of these is devoted solely to fitting. I am not an expert in histogram fitting, but I can share with you what I have learned and some of the pitfalls I have encountered. In particular I will use histograms from the ACD made from balloon data.

II. Simple fit.

a) The FitPanel. This is for quick and dirty fitting.



Functions: polynomials, exponentials, landau distributions, gaussians, user functions (make a TF1 function with the name "user");

Other options available for fit function (TF1 methods) by right clicking on fit line in histogram

b) In code:
Use the Fit method of histograms (TH1::Fit())

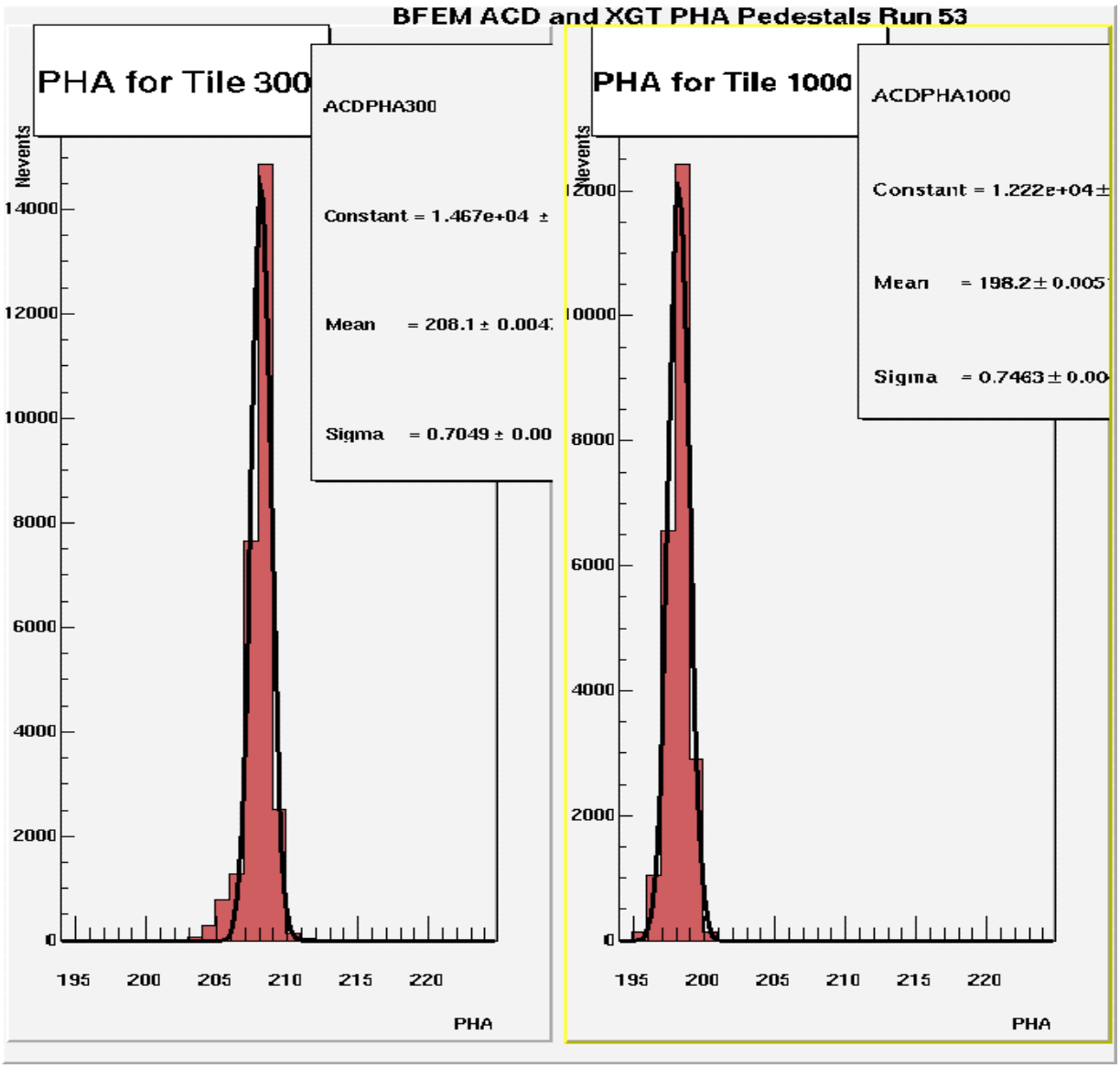
```
void Fit(const char *fname , Option_t
*option , Option_t  *goption, Axis_t xxmin,
Axis_t  xxmax)
```

function in fname pointer can be same as in fit panel or you can define your own as a TF1 Object.

More options available in code than in Fit Panel.

e.g. ACD pedestals
in loop of tiles:

```
TH1F *tile[17];
tile[8]  = (TH1F*) histFile->Get("ACDPHA300");
tile[i]->Fit("gaus","Q" ,"C" ,170,230);
```



To use example code comment out ACD pedestal subtraction and in RootTreeAnalysis.cxx

And make the binning use single channels..

Load file .L acdPlots_peds_test.cxx

AcdPlots("Histograms.root", "my psfile.ps");

c) Access to parameters in code: GetParameter method of TF1:

```
cout << tile[i]->GetName() << "    "
   << tile[i]->GetFunction("gaus")->GetParameter(0)
   << "   "
   << tile[i]->GetFunction("gaus")->GetParameter(1)
```

```
 << "   "
<< tile[i]->GetFunction("gaus")->GetParameter(2)
<< endl;
```

d) displaying parameters in stats box

gStyle->SetOptFit(0111);

`TH1::SetOptFit(mode)` method. This mode has four digits.

Mode = `pcev` (default = 0111)

- `v` = 1 print name/values of parameters
- `e` = 1 print errors (if e=1, v must be 1)
- `c` = 1 print Chi-square/number of degrees of freedom
- `p` = 1 print probability

e) defining your own functions.

```
 TF1  *f1 = new TF1("f1", "sin(x)/x", 0,10);
TF1 *f1 = new TF1("f1","[0]*x*sin([1]*x)",-3,3);
```

need to define initial parameters.

III. Fitting Sums of functions.

a) very common signal + noise + background fitting often needs to be done simultaneously.

`$ROOTSYS/tutorials/multifit.C`.

b) Fit method will take sums of canned functions.

TF1 * total = new TF1("total","gaus(0)+gaus(3)+gaus(6)",85,125);

i) separating parameters with ( ).

ii) need to initialize even canned functions (demo example)

iii) Can combine canned with user defined TF1 functions.

d) example ACD exponential noise + landau

TF1 * total = new TF1("total","expo(3)+landau(0)", 20, 800);

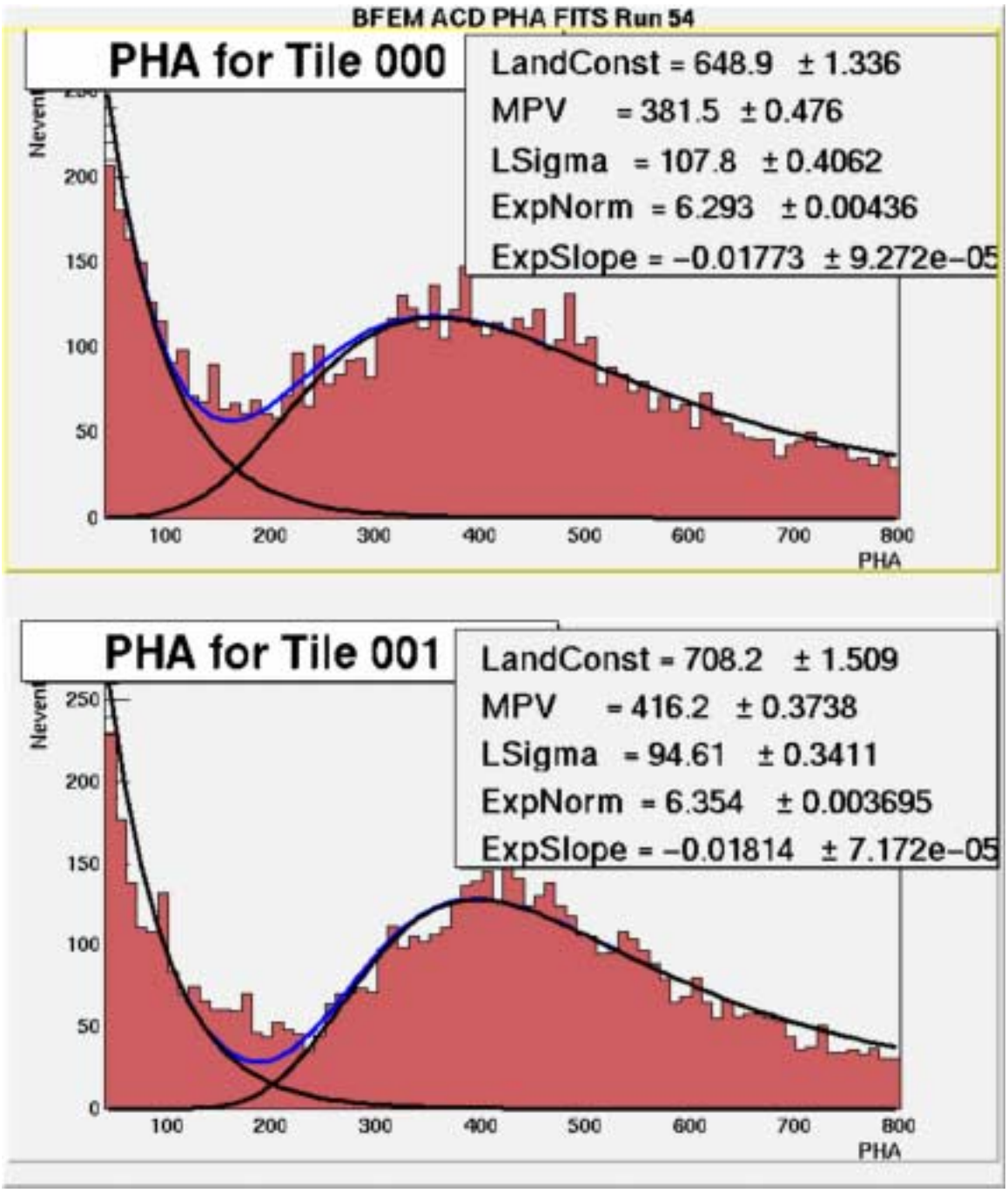
Note: landau needed to be first in Fit specification! not sure why.

```
// fitting functions
TF1 *exp1, *land, *total;
land = new TF1("land", "landau(0)",20,800);
exp1 = new TF1("exp1","expo(0)", 20, 800);
exp1->SetParLimits(1,-100000,0);
total = new TF1("total","expo(3)+landau(0)",
20, 800);

total->SetLineColor(4);

land->GetParameters(&par[0]);
land->SetRange(20,800);
exp1->GetParameters(&par[3]);
exp1->SetRange(20,800);

total->SetParameters(&par[0]);
total->SetParNames("LandConst","MPV",
                   "LSigma","ExpNorm","ExpSlope");
tile[i]->Fit(total,"WR");
```



d) displaying functions. [Draw("same")]

```
   total->GetParameters(&par[0]);

   // Feed fit parameters back for plotting final
fits.
  land->SetParameters(&par[0]);
  exp1->SetParameters(&par[3]);
```

```
tile[i]->Draw();
land->Draw("SAME");
exp1->Draw("SAME");
```

e) displaying results in stats box

```
gStyle->SetOptStat(0);
gStyle->SetOptFit(0);
.
.
```
draw histograms, fits, etc.
```
.
.
gStyle->SetOptFit(0101);
```

ACD Pedestal demo code (adapted from plotMacros/acdPlots.cxx, thanks Heather!):
Note first have to run RootTreeAnalysis code to generate Histogram file from data.

In this case pedestal subtraction code must be commented out and bin widths must be set to 1.

AcdPlots_peds_test.cxx

```
/*! acdPlots_peds
\brief This routine accepts as input a histogram file
generated via RootTreeAnalysis and generates
PHA plots for the ACDs and the XGTs.

  INPUT:  Path and filename of a histogram file generated
by RootTreeAnalysis
          Path and filename of a new file to contain the
output postscript file

  OUTPUT: Creates the postscript outputFile

  To RUN from the ROOT command line:
  .L acdPlots_peds_test.cxx
  acdPlots_peds("Histograms.root", "myNewPSFile.ps")

  */


void acdPlots_peds(char *histfile, char *outputFile) {

    gROOT->Reset();
    /// Setup some global options for the plots
    /// This will setup our Statistics box for each plot
    gStyle->SetOptStat(0000010);
    //customizing style for pedestal fits
    //gStyle->SetOptStat(111111);
    gStyle->SetOptFit(0111);

    gStyle->SetStatH(0.2);   // height
    gStyle->SetStatW(0.3);   // width
    gStyle->SetStatX(1.1);

    gStyle->SetTitleColor(0); // Title box background Color
```

```
    gStyle->SetTitleH(0.11);  // Fraction of title box
hight
    gStyle->SetTitleW(0.54);  // Fraction of title box
width
    gStyle->SetTitleX(0.05);   // X position of the title
box from left
    gStyle->SetTitleY(1.00);  // Y position of the title
box from bottom

    // setup the canvas to contain the plots
    TCanvas *c1 = new TCanvas("c1","BFEM ACD",0,0,700,850);
    c1->Divide(2,1,0.005,0.02);

    // Set canvas title
    c1->SetTitle("BFEM ACD and XGT PHA Pedestals");
    c1->GetFrame()->SetFillColor(46);
    c1->SetFillColor(19);
    c1->SetBorderMode(28);

    // Setup the title of the plots
    TText *c1title = new TText(0.35,0.98,"BFEM ACD and XGT
PHA Pedestals Run 53");
    c1title->SetTextSize(0.03);
    c1title->Draw();

    // open the histogram file
    TFile *histFile = new TFile(histfile);

    TH1F *tile[17];

    // Read in the histograms from the file
    tile[0]  = (TH1F*) histFile->Get("ACDPHA000");
    tile[1]  = (TH1F*) histFile->Get("ACDPHA001");
    tile[2]  = (TH1F*) histFile->Get("ACDPHA010");
    tile[3]  = (TH1F*) histFile->Get("ACDPHA011");
    tile[4]  = (TH1F*) histFile->Get("ACDPHA100");
    tile[5]  = (TH1F*) histFile->Get("ACDPHA110");
    tile[6]  = (TH1F*) histFile->Get("ACDPHA200");
    tile[7]  = (TH1F*) histFile->Get("ACDPHA210");
    tile[8]  = (TH1F*) histFile->Get("ACDPHA300");
    tile[9]  = (TH1F*) histFile->Get("ACDPHA310");
    tile[10] = (TH1F*) histFile->Get("ACDPHA400");
    tile[11] = (TH1F*) histFile->Get("ACDPHA410");
    tile[12] = (TH1F*) histFile->Get("ACDPHA1000");

    tile[13] = (TH1F*) histFile->Get("XGTPHA2000");
    tile[14] = (TH1F*) histFile->Get("XGTPHA2001");
```

```
    tile[15] = (TH1F*) histFile->Get("XGTPHA2010");
    tile[16] = (TH1F*) histFile->Get("XGTPHA2011");


   // c1->SetLogy(1);

    // Now draw the histograms and setup the features of
the plots
    int i;
    cout << "Gaussian fitting parameters\n"
         << " Tile  Gaus Height   Gauss Ped   width   " <<
endl;
    int c_count = 0;
    for(i=0;i<17;i++) {
      if (i== 8 || i == 12) {
        c1->cd(++c_count);
        tile[i]->SetFillColor(46);
        // The next line does not work here!
        // tile[1]->GetXaxis()->SetRange(170.230);
        tile[i]->Fit("gaus","Q" ,"C" ,170,230);
        //Double_t par =  tile[i]->GetFunction()-
>GetParameter(0);

        cout << tile[i]->GetName() << "   "
             << tile[i]->GetFunction("gaus")-
>GetParameter(0)  << "  "
             << tile[i]->GetFunction("gaus")-
>GetParameter(1)  << "  "
          << tile[i]->GetFunction("gaus")->GetParameter(2)
<< endl;
        tile[i]->SetXTitle("PHA");
        tile[i]->SetYTitle("Nevents");
        tile[i]->SetTitleOffset(1.3,"X");
        tile[i]->SetTitleOffset(1.0,"Y");
        tile[i]->GetXaxis()->SetRange(195,225);
        tile[i]->Draw();
      }
    }

    // Print out the plots to a postscript file
    c1->Print(outputFile);

}
```

ACD multiple fitting demo code (adapted from plotMacros/acdPlots.cxx, thanks Heather!):

:

acdPlots_thresh_test.cxx

```
/*! AcdPlots_thresh
\brief This routine accepts as input a histogram file
generated via RootTreeAnalysis and generates
PHA plots for the ACDs and the XGTs.

  INPUT:  Path and filename of a histogram file generated
by RootTreeAnalysis
          Path and filename of a new file to contain the
output postscript file

  OUTPUT: Creates the postscript outputFile

  To RUN from the ROOT command line:
  .L acdPlots_thresh_test.cxx
  acdPlots_thresh("Histograms.root", "myNewPSFile.ps")

  */


void acdPlots_thresh(char *histfile, char *outputFile) {

    gROOT->Reset();

    /// Setup some global options for the plots
    //  We disable the stats box nw - we only want the
final results

    gStyle->SetOptStat(0);
    gStyle->SetOptFit(0);

    gStyle->SetTitleColor(0); // Title box background Color
    gStyle->SetTitleH(0.11);  // Fraction of title box
hight
    gStyle->SetTitleW(0.54);  // Fraction of title box
width
    gStyle->SetTitleX(0.05);   // X position of the title
box from left
    gStyle->SetTitleY(1.00);  // Y position of the title
box from bottom

    // setup the canvas to contain the plots
    TCanvas *c1 = new TCanvas("c1","BFEM ACD",0,0,700,850);
```

```
    // c1->Divide(4,5,0.005,0.02);
    c1->Divide(1,2,0.005,0.02);

    // Set canvas title
    c1->SetTitle("BFEM ACD PHA");
    c1->GetFrame()->SetFillColor(46);
    c1->SetFillColor(19);
    c1->SetBorderMode(28);

    // Setup the title of the plots

    TText *c1title = new TText(0.35,0.98,"BFEM ACD PHA FITS
Run 54");
    c1title->SetTextSize(0.03);
    c1title->Draw();

    // Make labels for plots
    TPaveText *pt[17];


    // open the histogram file
    TFile *histFile = new TFile(histfile);

    TH1F *tile[17];

    // Read in the histograms from the file
    tile[0]  = (TH1F*) histFile->Get("ACDPHA000");
    tile[1]  = (TH1F*) histFile->Get("ACDPHA001");
    tile[2]  = (TH1F*) histFile->Get("ACDPHA010");
    tile[3]  = (TH1F*) histFile->Get("ACDPHA011");
    tile[4]  = (TH1F*) histFile->Get("ACDPHA100");
    tile[5]  = (TH1F*) histFile->Get("ACDPHA110");
    tile[6]  = (TH1F*) histFile->Get("ACDPHA200");
    tile[7]  = (TH1F*) histFile->Get("ACDPHA210");
    tile[8]  = (TH1F*) histFile->Get("ACDPHA300");
    tile[9]  = (TH1F*) histFile->Get("ACDPHA310");
    tile[10] = (TH1F*) histFile->Get("ACDPHA400");
    tile[11] = (TH1F*) histFile->Get("ACDPHA410");
    tile[12] = (TH1F*) histFile->Get("ACDPHA1000");

    tile[13] = (TH1F*) histFile->Get("XGTPHA2000");
    tile[14] = (TH1F*) histFile->Get("XGTPHA2001");
    tile[15] = (TH1F*) histFile->Get("XGTPHA2010");
    tile[16] = (TH1F*) histFile->Get("XGTPHA2011");


    c1->SetLogy(1);
```

```
    // Now draw the histograms and setup the features of
the plots
    int i;
    //cout << "Landau fitting parameters\n"
    //     << " Tile   Landau V, Error Size   " << endl;

    // fitting functions
    TF1 *exp1, *land, *total;
    // fitting parameters
    Double_t par[5];

    for(i=0;i<2;i++) {
        c1->cd(i+1);
        tile[i]->SetFillColor(46);

        land = new TF1("land", "landau(0)",20,800);
        exp1 = new TF1("exp1","expo(0)", 20, 800);
        exp1->SetParLimits(1,-100000,0);
        total = new TF1("total","expo(3)+landau(0)", 20,
800);

        tile[i]->Fit(land,"WQ0","",300,800);
        tile[i]->Fit(exp1,"WQ0+","",20,200);

        total->SetLineColor(4);

        land->GetParameters(&par[0]);
        land->SetRange(20,800);
        exp1->GetParameters(&par[3]);
     exp1->SetRange(20,800);

        total->SetParameters(&par[0]);
        total-
>SetParNames("LandConst","MPV","LSigma","ExpNorm","ExpSlope
");
        tile[i]->Fit(total,"WR");

        total->GetParameters(&par[0]);

        // Feed fit parameters back for plotting final
fits.
        land->SetParameters(&par[0]);
        exp1->SetParameters(&par[3]);

        // Find bin with pak value
        int startbin = 30;
```

```
        Double_t lymax, y;
        Double_t lxmax = (Double_t) tile[i]-
>GetBinCenter(startbin);
        Double_t x = lxmax;
        lymax = 0;

        for (x = lxmax; x< 1300.; x += (Double_t) tile[i]-
>GetBinWidth(startbin)) {
       if( (y = land->Eval(x,0,0)) > lymax) { lymax =y;
lxmax = x; }
        }
        cout << " max of landau occurs at " << lxmax <<
endl;

        land->DrawCopy("SAME");
        exp1->DrawCopy("SAME");

        pt[i] = new TPaveText(500., lymax+30, 800.,
lymax+90., "br");
        char valuestr[30];
        sprintf(valuestr,"Peak %6.0f",lxmax);
        TText * t1 = pt[i]->AddText(valuestr);

        char valuestr2[30];

        sprintf(valuestr2,"MPV %6.0f",tile[i]-
>GetFunction("total")->GetParameter(1));
        TText * t2 = pt[i]->AddText(valuestr2);


     // cout << tile[i]->GetName() << "   "
        //     << tile[i]->GetFunction("landau")-
>GetParameter(0)  << "  "
        //     << tile[i]->GetFunction("landau")-
>GetParameter(1)  << "  "
     //    << tile[i]->GetFunction("landau")-
>GetParameter(2)  << endl;
        cout << tile[i]->GetName() << "   "
             << tile[i]->GetFunction("total")-
>GetParameter(0)  << "  "
            << tile[i]->GetFunction("total")-
>GetParameter(1)  << "  "
         << tile[i]->GetFunction("total")->GetParameter(2)
<< "  "
         << tile[i]->GetFunction("total")->GetParameter(3)
<< "  "
```

```
          << tile[i]->GetFunction("total")->GetParameter(4)
<< "   "
             << endl;
         tile[i]->SetXTitle("PHA");
         tile[i]->SetYTitle("Nevents");
         tile[i]->SetTitleOffset(1.3,"X");
         tile[i]->SetTitleOffset(1.0,"Y");
         if(i == 7) { tile[i]->GetXaxis()->SetRange(5,160);
}
         else { tile[i]->GetXaxis()->SetRange(5,80);}
         tile[i]->Draw();
         land->Draw("SAME");
         exp1->Draw("SAME");
         pt[i]->Draw("same");
   }

    // uncomment the next line to get fitting parameters
    // gStyle->SetOptFit();

    gStyle->SetOptFit(0101);

    // Print out the plots to a postscript file
    c1->Print(outputFile);

}
```