Introduction to ROOT: part 3

Mirco Dorigo
mirco.dorigo@ts.infn.it

LACD 2024-2025
March 26th, 2025

Take home messages from last class

1. We learnt how to read a txt file to take input data (formatted as a table “columns of variables, rows of events”).
    - **Always double check what you are reading.**
2. Convert (immediately) your data into a `TTree`
    - It enables **easier inspections. Check the data** in an interactive root session.
3. We learnt also how to plot an histogram.
   This is usually done after knowing what we want/expect to see.

Exercises

1. We still have to see a signal peak… Modify the macro to plot the histogram of $p_x(K)$:

   A. For each event, using the K and π momenta, their known masses, and the CM energy, calculate the invariant mass M defined in slide 4. You can either do the calculation by hand or use the class TLorentzVector, which deals with 4-vectors.
   Plot the distribution of M.

   B. A key variable is the difference between the measured B energy (in the CM) and half of the collision energy, $\Delta E = E^* - \sqrt{s}/2$. Calculate the variable for each event and plot the distribution.

   C. Describe the M and ΔE distributions (mean, standard dev…): do they look as expected?

2. Modify makeTree.C to add these two new variables to the `TTree` and save the tree in a file.

3. Have a look at the macro `computeP.C` from the lesson material — see next slides.
   Try to understand and run it, see the use of standard `C++` libraries (vector, numeric) and another ROOT class (TVector3). Modify the macro to add the plot of the momentum variable calculated there.

Breaking exercise 1-2

From the K and π momenta in the CM, we need to calculate two variables:

$$M = \sqrt{s/4 - |\vec{p}_B^*|^2}$$

$$\Delta E = E^* - \sqrt{s}/2$$

where $\sqrt{s} = 10.5794\ \mathrm{GeV}$ and $\vec{p}_B^*$ and E^* are the B-candidate momentum and energy in the CM.

We will do using the TLorentzVector class and save the variable directly in a `TTree`. Let's take the macro `makeTree.C` and modify it.

Breaking the exercise

```
#include "Riostream.h"
#include "TString.h"
#include "TH1D.h"
#include "TTree.h"
#include "TFile.h"
#include "TLorentzVector.h"
```

Check the class

```
    int icand = 0;
    double k_px,  k_py,  k_pz;
    double pi_px, pi_py, pi_pz;

    //define the new variables
    double B_m, B_de;
    //and those needed for the calculation
    const double pi_m = 0.1396;
    const double k_m = 0.4937;
    const double sqrt_s = 10.5794;

```

New variables

Taken from PDG

```
    TTree* dataTree = new TTree("dataTree","B0toKpi data");

    //the K momentum components
    dataTree->Branch("k_px",&k_px,"k_px/D");
    dataTree->Branch("k_py",&k_py,"k_py/D");
    dataTree->Branch("k_pz",&k_pz,"k_pz/D");
    //the pi momentum components
    dataTree->Branch("pi_px",&pi_px,"pi_px/D");
    dataTree->Branch("pi_py",&pi_py,"pi_py/D");
    dataTree->Branch("pi_pz",&pi_pz,"pi_pz/D");
    //add the new variables to the tree
    dataTree->Branch("B_m",&B_m,"B_m/D");
    dataTree->Branch("B_de",&B_de,"B_de/D");
```

Breaking the exercise

```
    while(file_in.is_open()){

        file_in >> k_px  >> k_py  >> k_pz
                >> pi_px >> pi_py >> pi_pz;

        if(file_in.eof()) break;

        h_px->Fill(k_px);

        //define the 4-momenta of the kaon and pion
        TLorentzVector k_p, pi_p;
        k_p.SetXYZM(k_px,k_py,k_pz,k_m);
        pi_p.SetXYZM(pi_px,pi_py,pi_pz,pi_m);

        //Compute the B 4-momentum
        TLorentzVector B_p = k_p + pi_p;

        //take the B energy with B_p.E(), and calcuate DeltaE
        B_de = B_p.E() - sqrt_s/2;
        //take the B momentum vector with B_p.Vect()
        //calculate the magnitude squared with .Mag2()
        //and compute the mass
        B_m = sqrt( sqrt_s*sqrt_s/4 - B_p.Vect().Mag2() );

        //fill the tree
        dataTree->Fill();

        ++icand;
    }
```

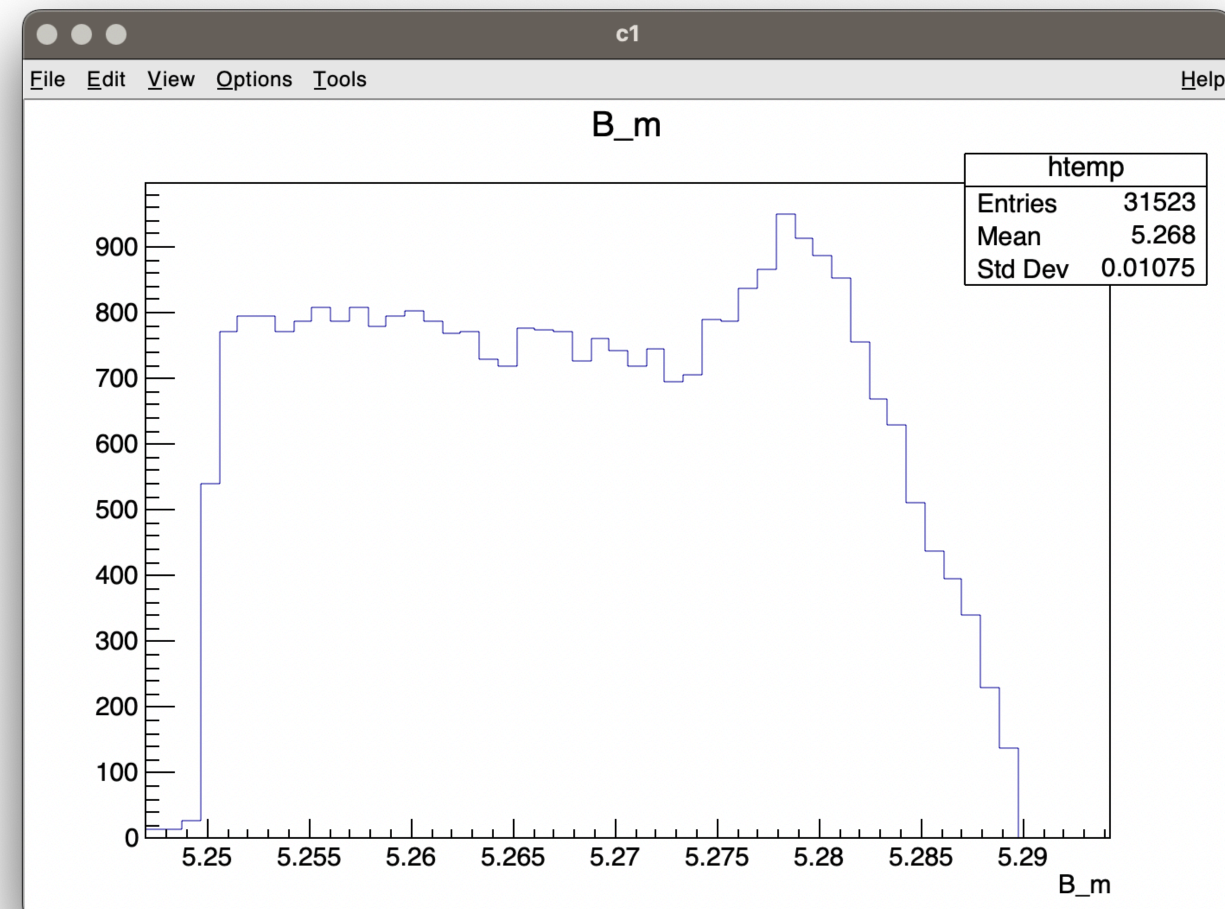
Breaking the exercise

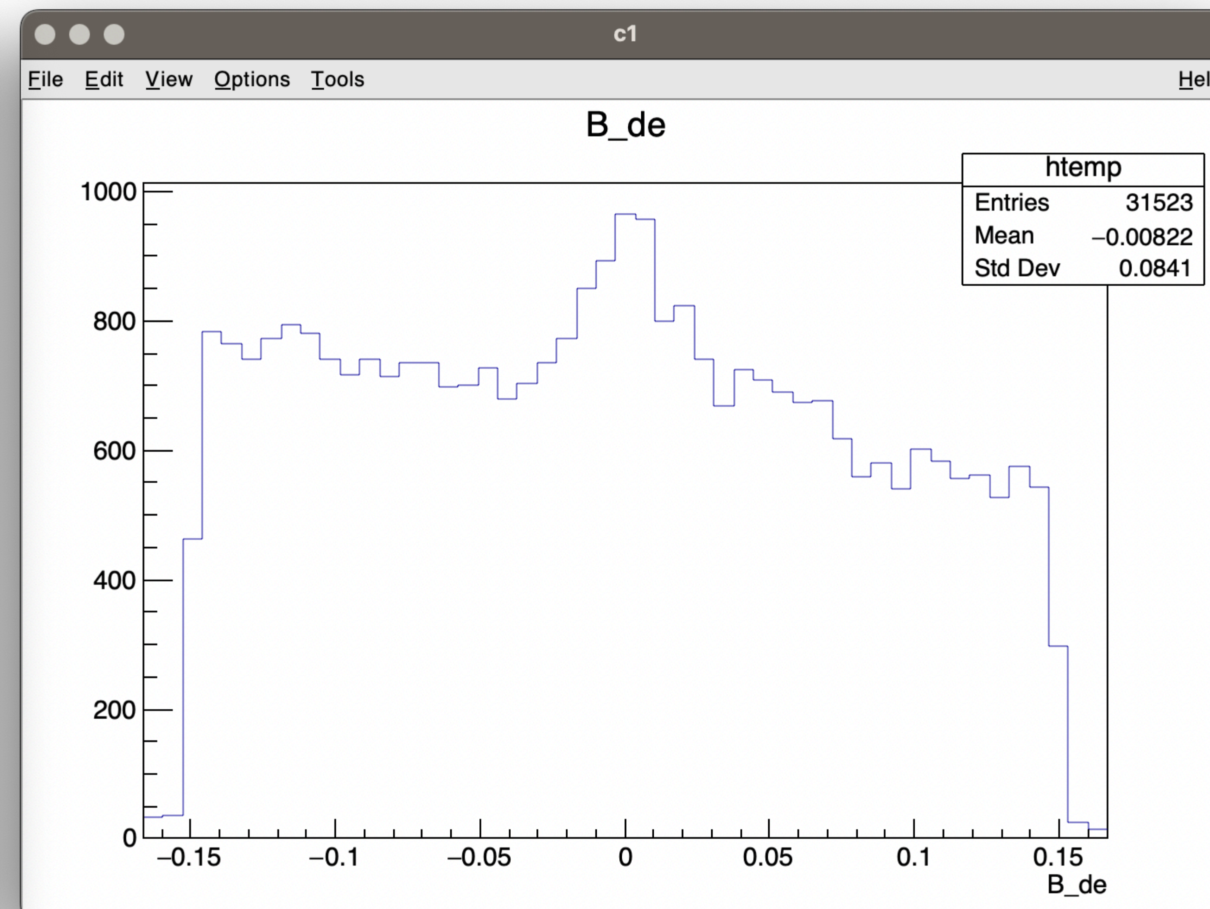
```
    cout << "Number of candidates: " << ica

    //make a trivial check...
    cout << "Candidates in the tree: " << c
    //look at the content
    dataTree->Print();

    //store now in a root file
    TFile* dataFile = new TFile("data_B0toK
    dataTree->Write();
    h_px->Write();
    dataFile->Close();

    return;
```

```
root [0]
Processing makeTree.C...
Number of candidates: 31523
Candidates in the tree: 31523
******************************************************************************
*Tree    :dataTree  : B0toKpi data                                           *
*Entries :    31523 : Total =         2031886 bytes  File  Size =          0 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :k_px      : k_px/D                                                 *
*Entries :    31523 : Total  Size=     253945 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    1 :k_py      : k_py/D                                                 *
*Entries :    31523 : Total  Size=     253945 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    2 :k_pz      : k_pz/D                                                 *
*Entries :    31523 : Total  Size=     253945 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    3 :pi_px     : pi_px/D                                                *
*Entries :    31523 : Total  Size=     253965 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    4 :pi_py     : pi_py/D                                                *
*Entries :    31523 : Total  Size=     253965 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    5 :pi_pz     : pi_pz/D                                                *
*Entries :    31523 : Total  Size=     253965 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    6 :B_m       : B_m/D                                                  *
*Entries :    31523 : Total  Size=     253925 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    7 :B_de      : B_de/D                                                 *
*Entries :    31523 : Total  Size=     253945 bytes  All baskets in memory   *
*Baskets :        7 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
root [1]
```

The distributions

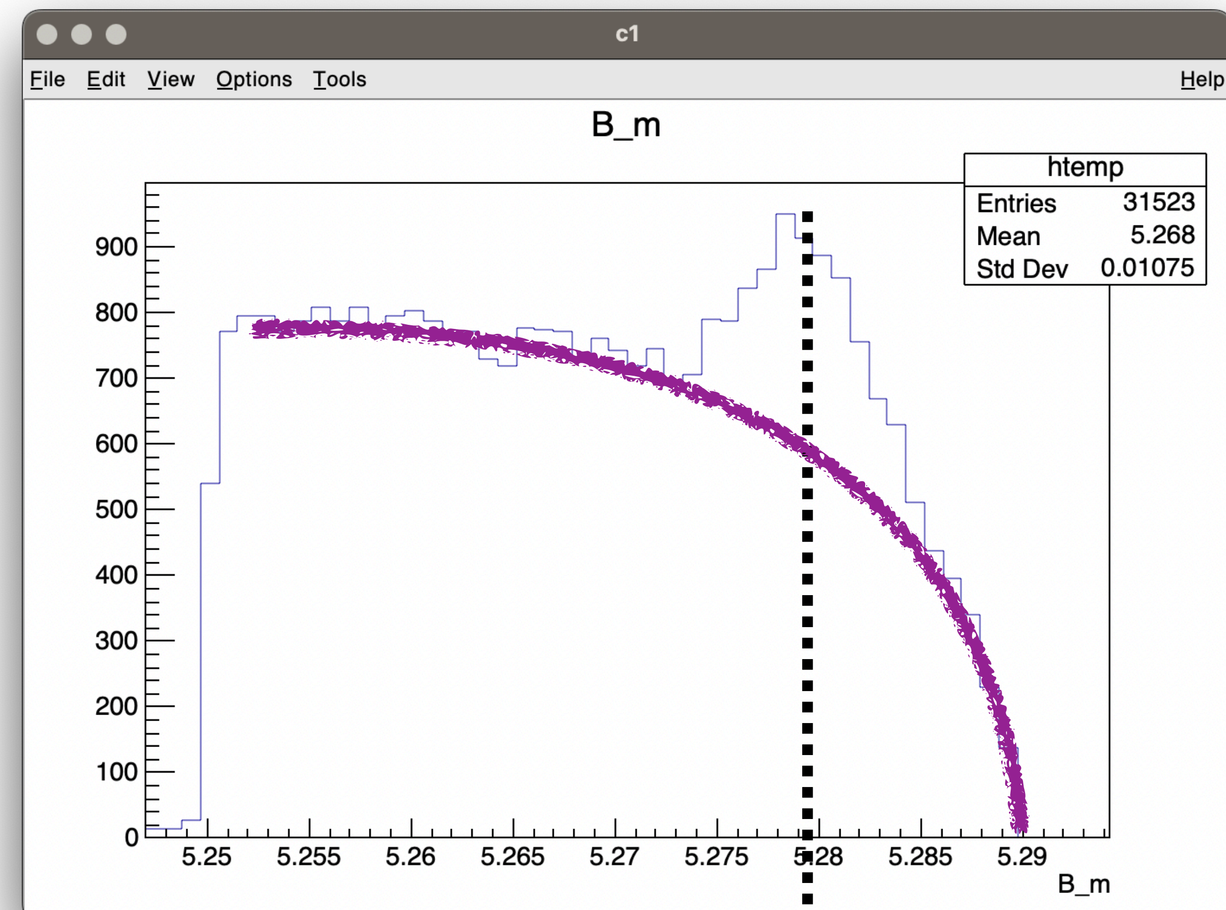
```
mb-md-01:thirdLesson dorigo$ root -l data_B0toKpi.root
root [0]
Attaching file data_B0toKpi.root as _file0...
(TFile *) 0x7fee07a9a360
root [1] .ls
TFile**         data_B0toKpi.root
 TFile*         data_B0toKpi.root
  KEY: TTree    dataTree;1      B0toKpi data
  KEY: TH1D     h_K_px;1
root [2] dataTree->Draw("B_m")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] dataTree->Draw("B_de")
root [4]
```

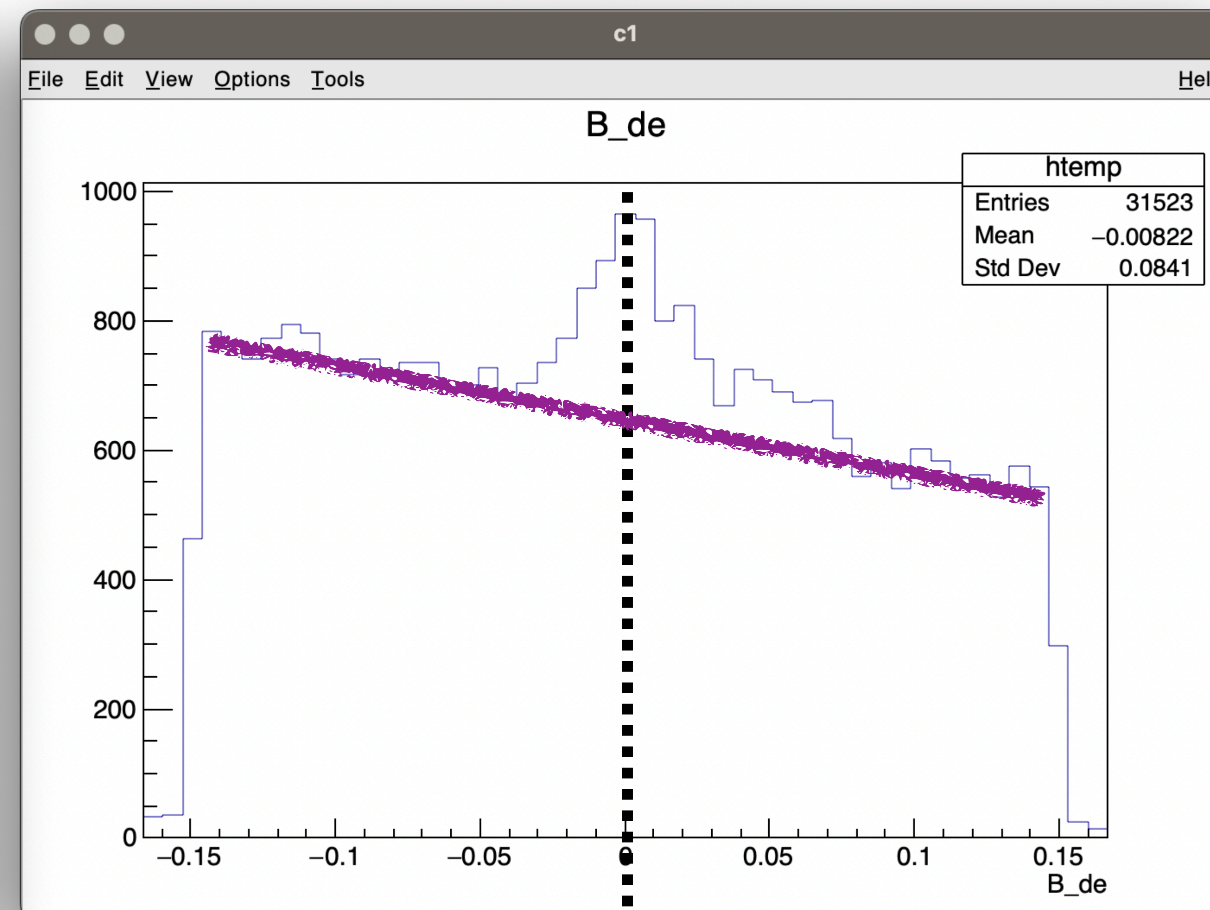




The peak

```
mb-md-01:thirdLesson dorigo$ root -l data_B0toKpi.root
root [0]
Attaching file data_B0toKpi.root as _file0...
(TFile *) 0x7fee07a9a360
root [1] .ls
TFile**         data_B0toKpi.root
 TFile*         data_B0toKpi.root
  KEY: TTree    dataTree;1      B0toKpi data
  KEY: TH1D     h_K_px;1
root [2] dataTree->Draw("B_m")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] dataTree->Draw("B_de")
root [4]
```

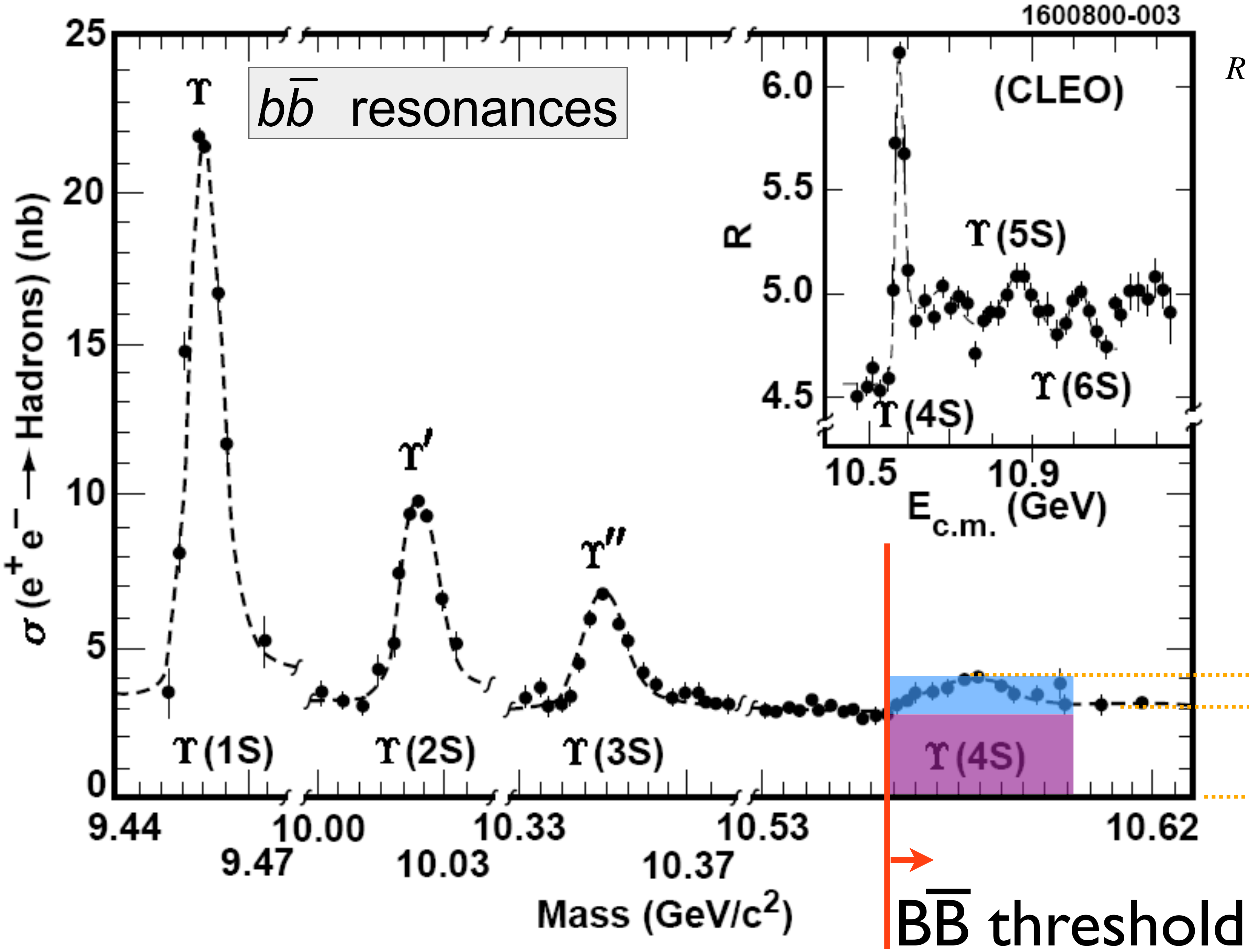




m(B^0) ~ 5.280 GeV/c^2

Expect ~0 for a B^0

A lot of background



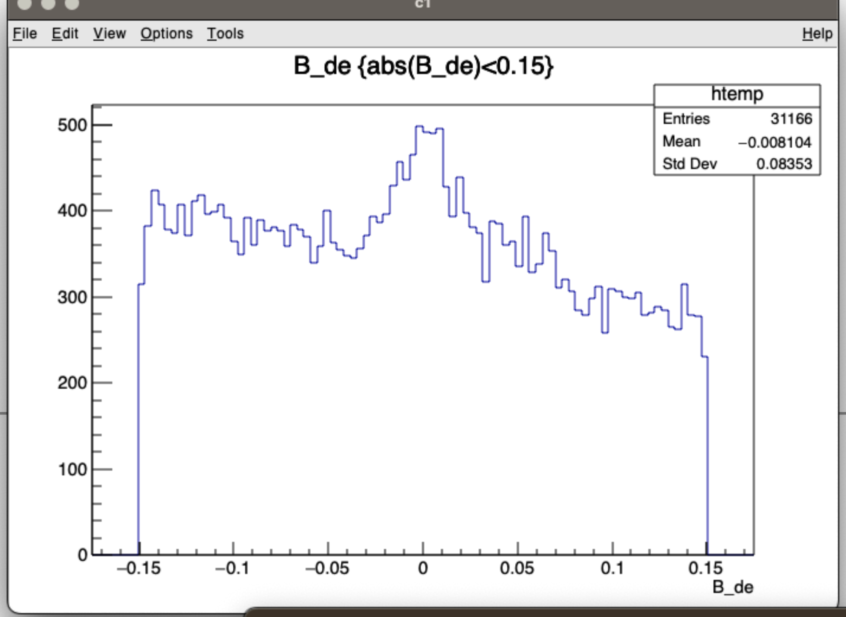
$$R = \frac{\sigma(e^+e^- \to q\bar{q})}{\sigma(e^+e^- \to \mu^+\mu^-)} = 3\sum_q e_q^2/e^2$$

Signal

$$\frac{\sigma(b\bar{b})}{\sigma(hadrons)} = 0.28$$

Background

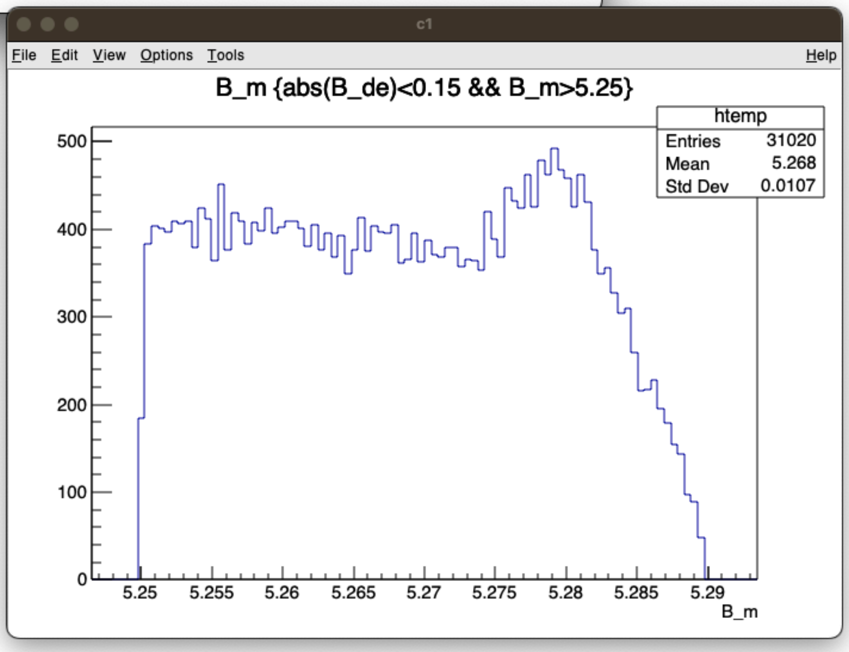
Let’s explore the data online



- You can draw the data from the prompt, making also selections

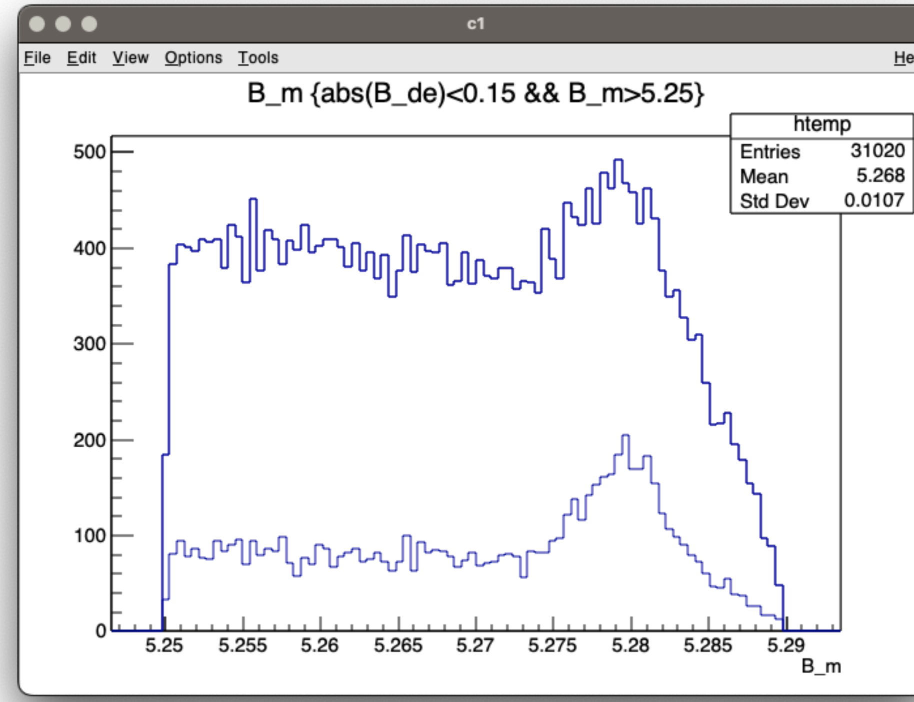
```
mb-md-01:thirdLesson dorigo$ root -l data_B0toKpi.root
root [0]
Attaching file data_B0toKpi.root as _file0...
(TFile *) 0x7fcbfd607300
root [1] dataTree->Draw("B_de", "abs(B_de)<0.15")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
(long long) 31166
```

```
root [3] dataTree->Draw("B_m", "abs(B_de)<0.15 && B_m>5.25")
(long long) 31020
root [4]
```



- And adding drawing options

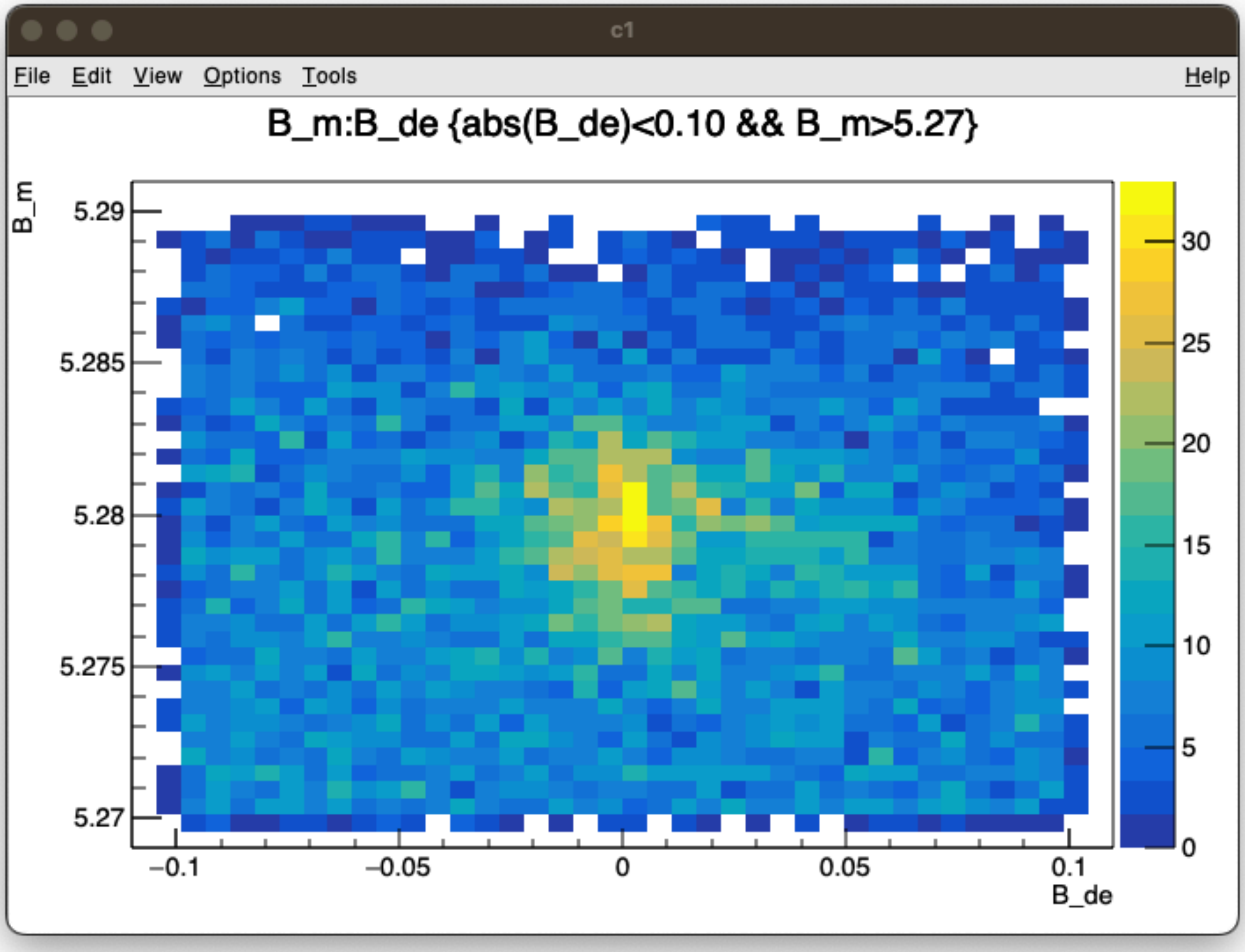
```
root [7] dataTree->Draw("B_m", "abs(B_de)<0.03 && B_m>5.25","same")
(long long) 7399
```



Let’s explore the data online

- Can also draw 2D distributions

```
root [10] dataTree->Draw("B_m:B_de", "abs(B_de)<0.10 && B_m>5.27","COLZ")
(long long) 10068
```



Some drawing options

- Added some histograms in `makeTree.C` to show some drawing options

```
    //histograms of the new variables
    TH1D* h_B_de = new TH1D("h_B_de", " ; #DeltaE [GeV]; counts", 20, -0.15, 0.15);
    TH1D* h_B_m = new TH1D("h_B_m", " ; M(B) [GeV/c^{2}]; counts", 20, 5.26, 5.29);

    //a 2D histogam of the new variables
    TH2D* h_B_de_m = new TH2D("h_B_de_m", " ; #DeltaE [GeV]; M(B) [GeV/c^{2}]", //titles...
                              30,-0.15, 0.15, //binning and range in x-axis
                              20, 5.26, 5.29); //binning and range in y-axis
```

```
        //fill the tree
        dataTree->Fill();

        //fill the new histograms
        h_B_de->Fill(B_de);
        h_B_m->Fill(B_m);
        h_B_de_m->Fill(B_de,B_m);

```

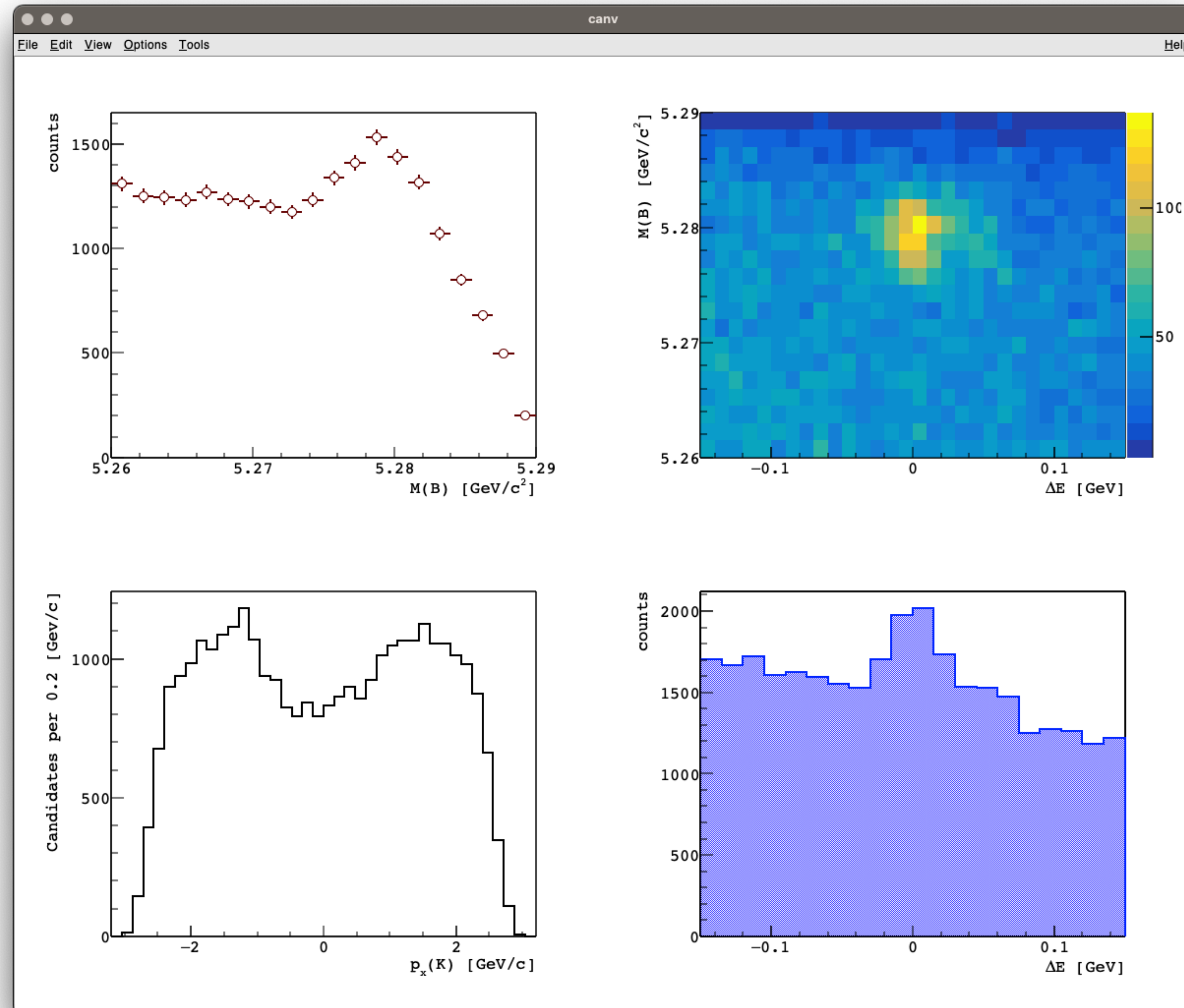
Some drawing options

- Create a canvas, split in 2x2 parts, and draw histograms in different forms

```
TCanvas* canv = new TCanvas("canv","canv",1200,1000);
canv->Divide(2,2); //split the canvas in 2x2 parts

canv->cd(1);//enter the first part
//a few drawing options
h_B_m->SetMarkerStyle(4);
h_B_m->SetMarkerSize(1);
h_B_m->SetMarkerColor(kRed+3);
h_B_m->SetLineColor(kRed+3);
h_B_m->SetMinimum(0);
h_B_m->Draw("err");//draw with error bars

canv->cd(2);//enter the second part
h_B_de_m->Draw("COLZ");

canv->cd(4);//enter the fourth part
h_B_de->SetFillStyle(3001);
h_B_de->SetFillColor(kBlue-4);
h_B_de->SetLineColor(kBlue);
h_B_de->SetLineWidth(2);
h_B_de->SetMinimum(0);
h_B_de->Draw("histo");

canv->cd(3); //enter the third part
h_px->Draw();
```

Some drawing options

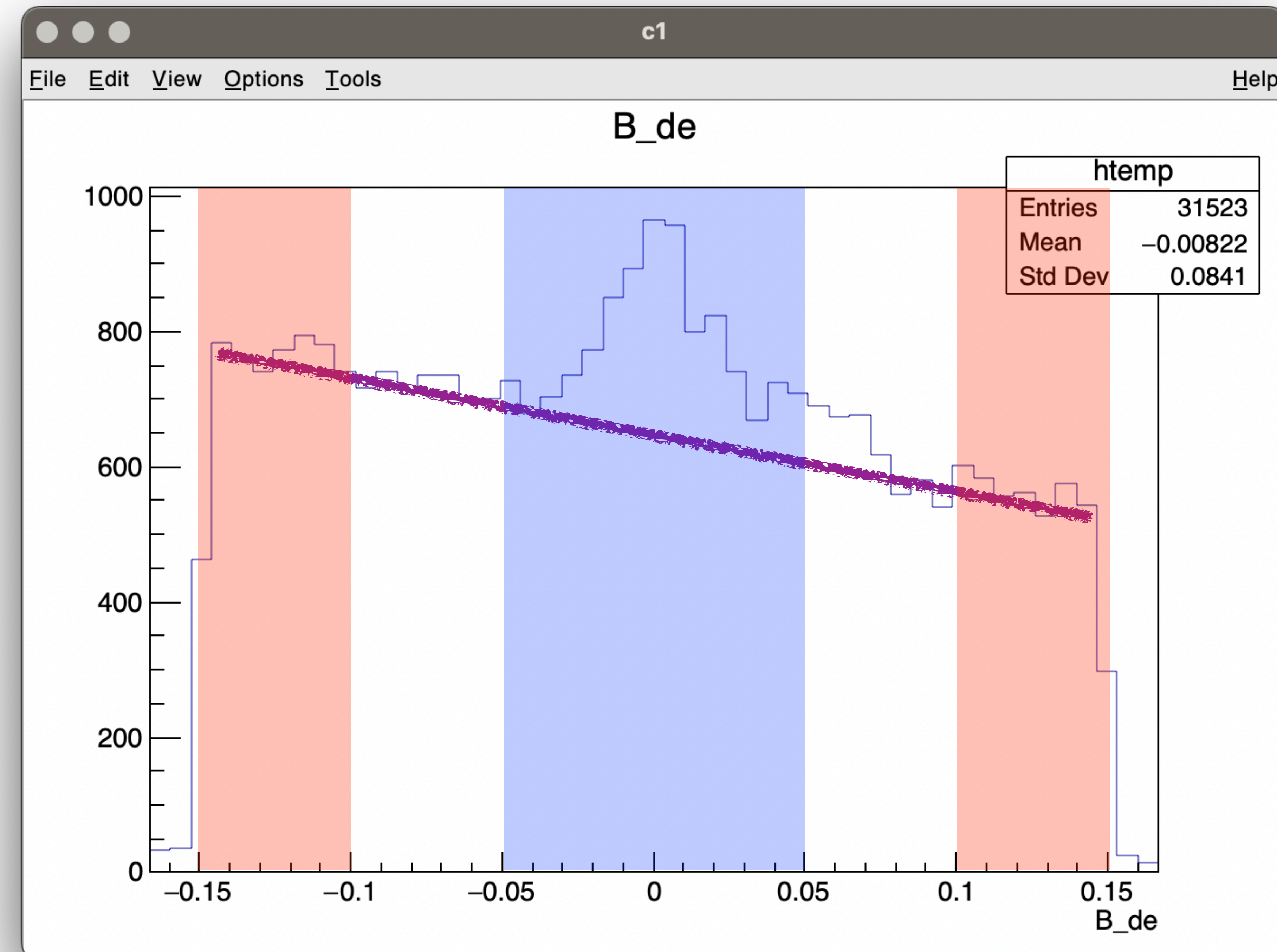
- The output



https://root.cern.ch/doc/v632/classTHistPainter.html
https://root.cern.ch/manual/histograms

Make histograms of signal and background

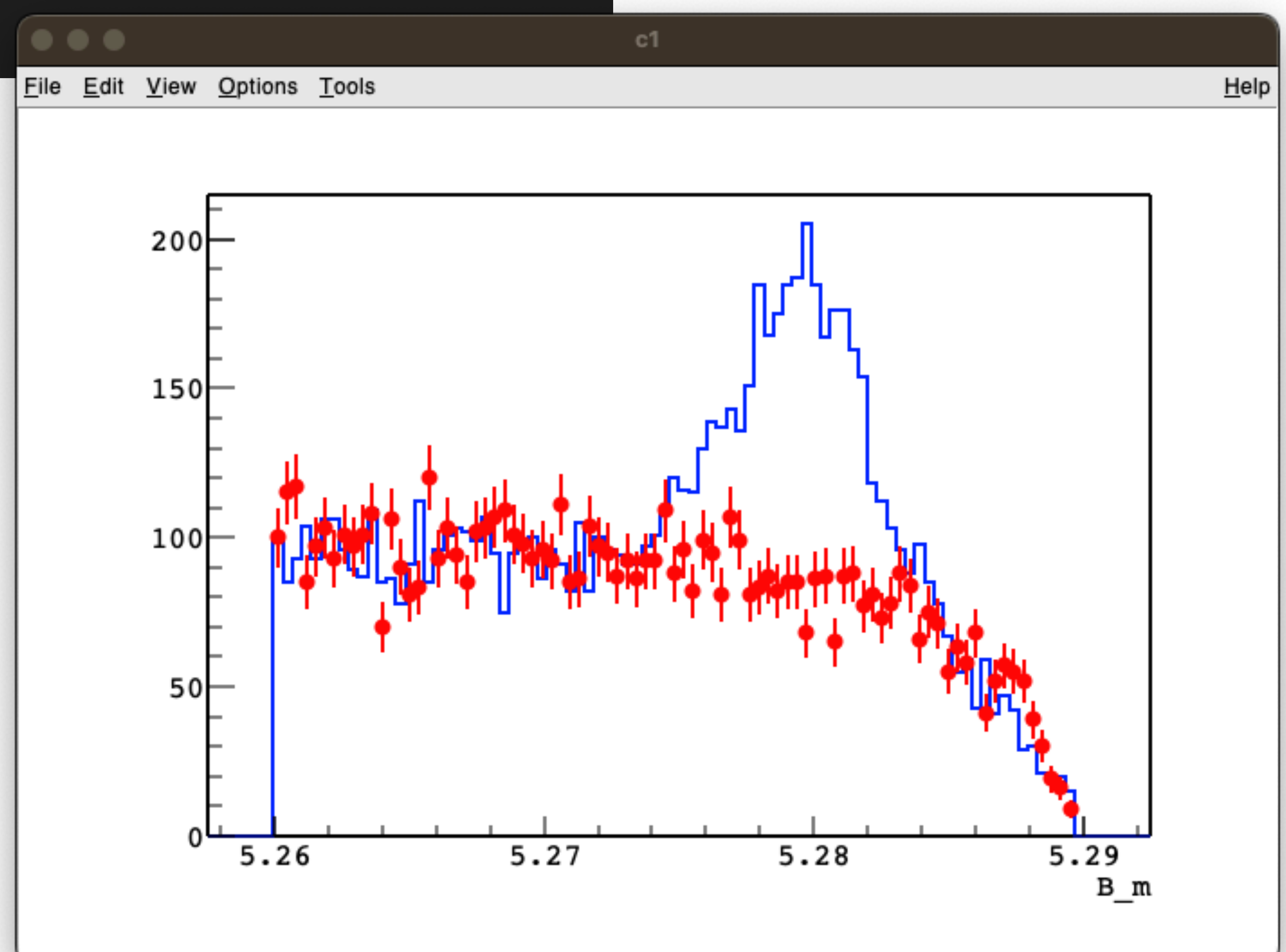
- Let's have a look at the B mass for these categories of events in ΔE:



Make histograms of signal and background

- Let's have a look at the B mass for these categories of events in ΔE:

```
mb-md-01:thirdLesson dorigo$ root -l data_B0toKpi.root
root [0]
Attaching file data_B0toKpi.root as _file0...
(TFile *) 0x7f7be62541d0
root [1] dataTree->Draw("B_m","B_m>5.26 && abs(B_de)<0.05")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
(long long) 8634
root [2] dataTree->Draw("B_m","B_m>5.26 && abs(B_de)>0.1","same")
(long long) 7117
root [3]
```



- We can use these histograms "to extract" the peak: let's do it in a script.

- We will **read the `TTree`**, create the histograms and **make their difference**.

Reading a tree (`histoPeak.C`)

```
#include "Riostream.h"
#include "TFile.h"
#include "TTree.h"
#include "TCanvas.h"
#include "TH1D.h"

using namespace std;

void histoPeak(){

    //open the root file to read
    TFile* file = TFile::Open("./data_B0toKpi.root");
    //and take the tree with the method Get()
    TTree* tree = (TTree*) file->Get("dataTree");

    //just a trivial check
    int tot_entries = tree->GetEntries();
    cout << "Total entries in the tree: " << tot_entries << endl;

    //define the variable we want to access to
    double B_m, B_de;

    //and link them to the branch address of the tree
    tree->SetBranchAddress("B_m",&B_m);
    tree->SetBranchAddress("B_de",&B_de);
```

Use directly the method while defining the (pointer to the) object

`Get()` is general from `TObject`, need to "cast" the type

Very similar to the definition of the branches

Reading a tree (`histoPeak.C`)

```
    //just two histograms to fill
    TH1D* h_m_sig = new TH1D("h_m_sig",
                             ";m(B) [GeV/c^{2}]; Entries",
                             30,5.26,5.29);
    h_m_sig->Sumw2();// very important!

    //let's clone the histogram for the background
    TH1D* h_m_bkg = (TH1D*) h_m_sig->Clone("h_m_bkg");

    //loop over the entries
    for(int iEntry; iEntry<tot_entries; ++iEntry){

        //take an entry
        tree->GetEntry(iEntry);

        //fill the histograms
        if(fabs(B_de)>0.1) h_m_bkg->Fill(B_m);
        else if(fabs(B_de)<0.05) h_m_sig->Fill(B_m);
    }


    //let's clone the histogram
    TH1D* h_m_peak = (TH1D*) h_m_sig->Clone("h_m_peak");
    //Let's subtract the histogram of the background!
    h_m_peak->Add(h_m_bkg,-1);
```

Take the i-th entry:
all variables linked to
the branch addresses
take the values for the
i-th candidate in the tree

Manipulating histograms (`histoPeak.C`)

```
//just two histograms to fill
TH1D* h_m_sig = new TH1D("h_m_sig",
                         ";m(B) [GeV/c^{2}]; Entries",
                         30,5.26,5.29);
h_m_sig->Sumw2();// very important!

//let's clone the histogram for the background
TH1D* h_m_bkg = (TH1D*) h_m_sig->Clone("h_m_bkg");

//loop over the entries
for(int iEntry; iEntry<tot_entries; ++iEntry){

    //take an entry
    tree->GetEntry(iEntry);

    //fill the histograms
    if(fabs(B_de)>0.1) h_m_bkg->Fill(B_m);
    else if(fabs(B_de)<0.05) h_m_sig->Fill(B_m);
}


//let's clone the histogram
TH1D* h_m_peak = (TH1D*) h_m_sig->Clone("h_m_peak");
//Let's subtract the histogram of the background!
h_m_peak->Add(h_m_bkg,-1);
```

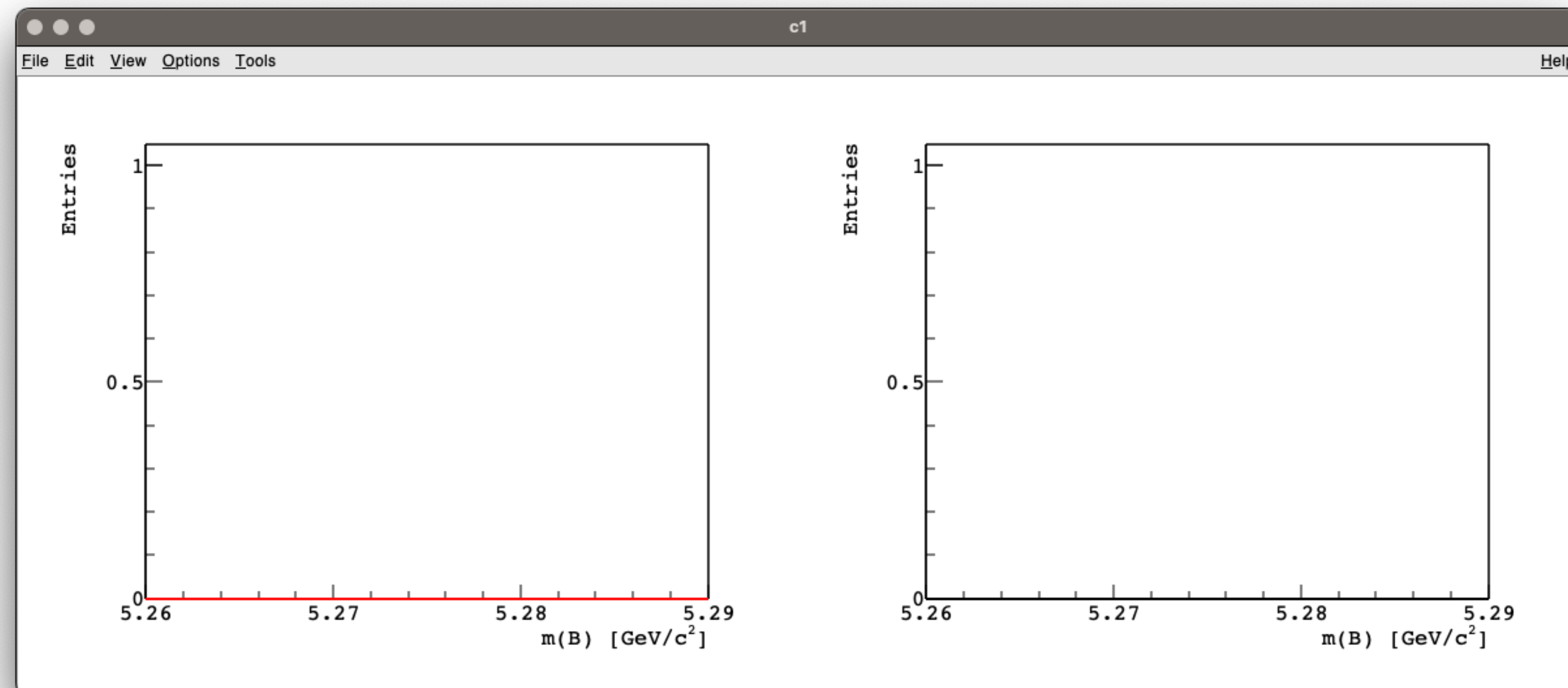
Define an signal histogram. Set `Sumw2()`: this is extremely important for the correct calculation of errors;

Clone for the background histogram.

Clone again: now the histogram is not empty. Make the subtraction with Add().

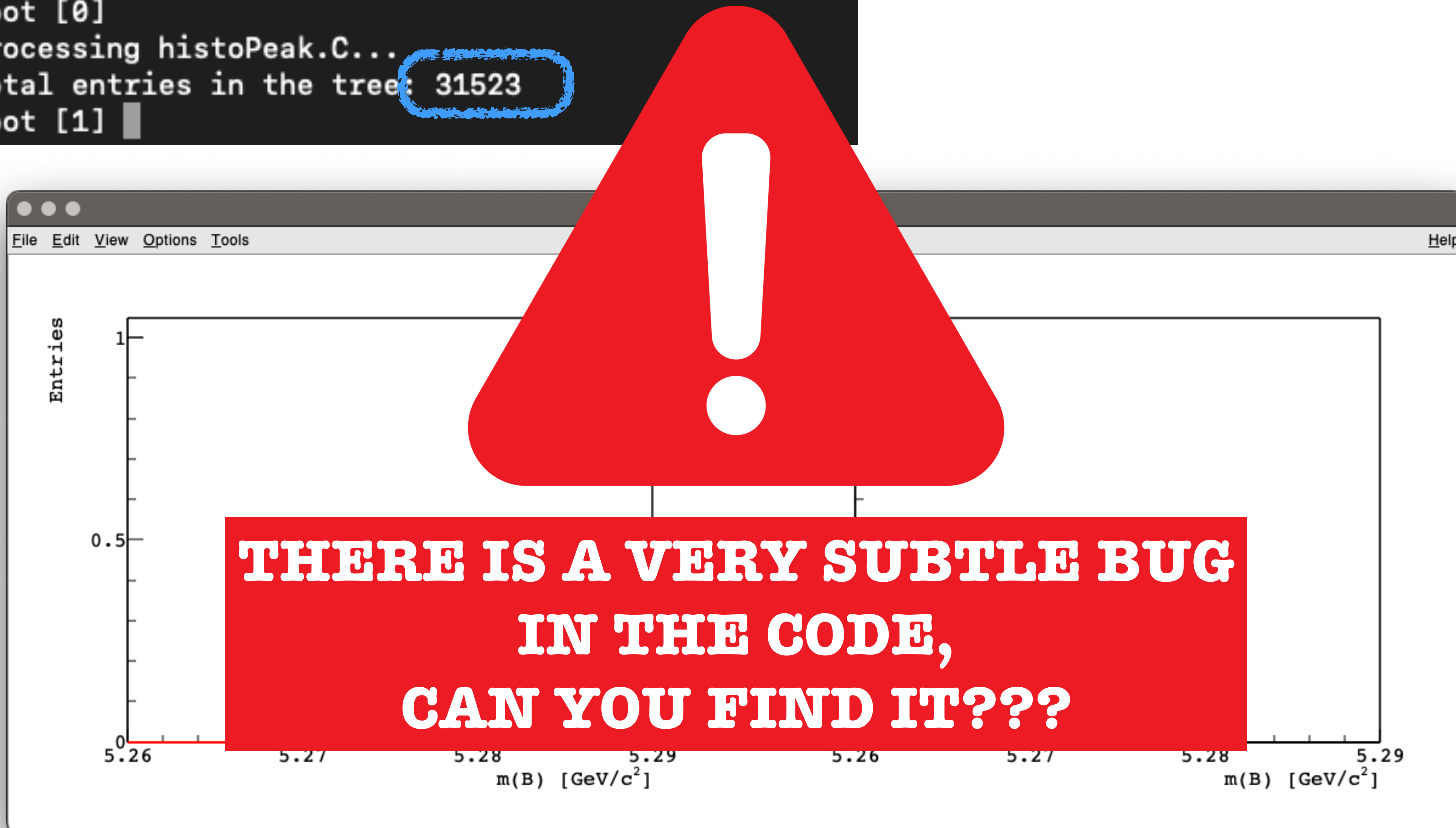
The output (`histoPeak.C`)

```
mb-md-01:thirdLesson dorigo$
mb-md-01:thirdLesson dorigo$ root -l histoPeak.C
root [0]
Processing histoPeak.C...
Total entries in the tree: 31523
root [1]
```



The output (`histoPeak.C`)

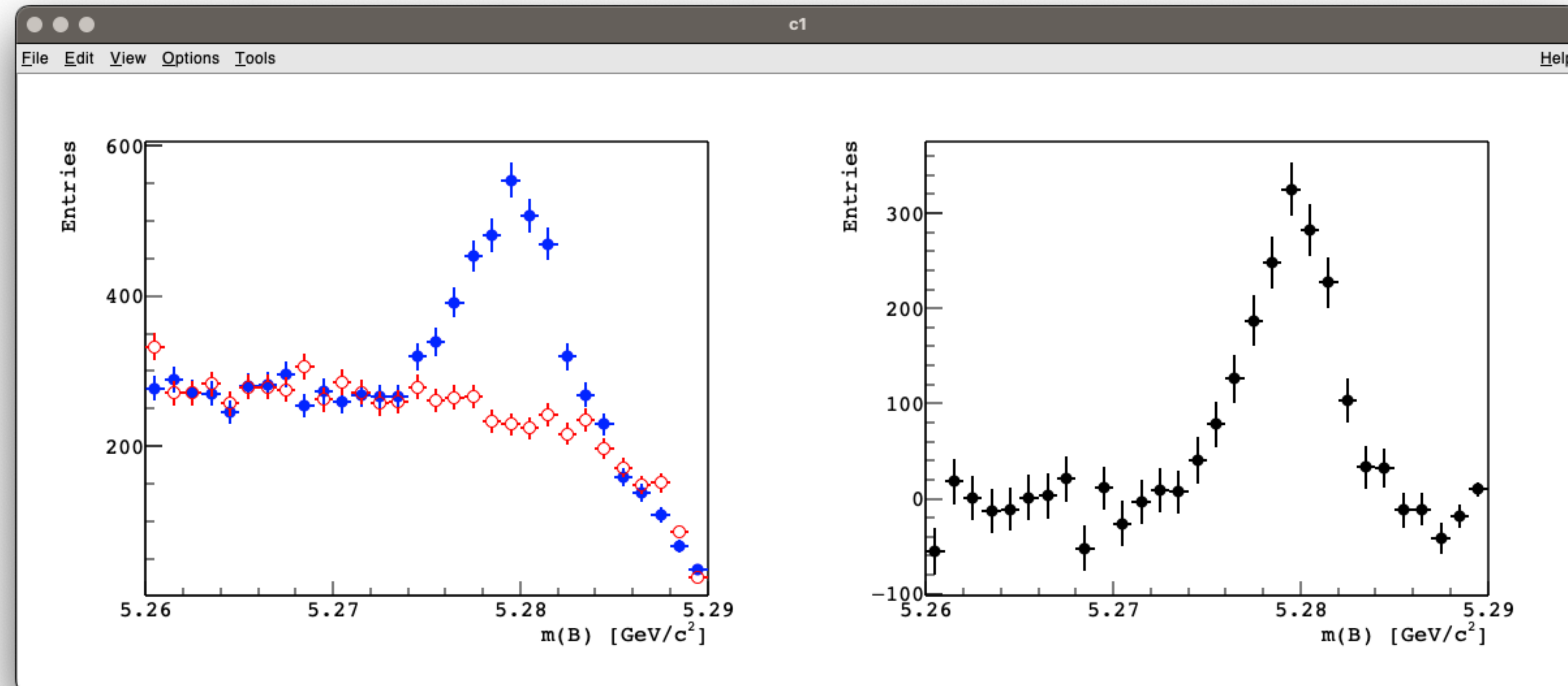
```
mb-md-01:thirdLesson dorigo$
mb-md-01:thirdLesson dorigo$ root -l histoPeak.C
root [0]
Processing histoPeak.C...
Total entries in the tree: 31523
root [1]
```



THERE IS A VERY SUBTLE BUG IN THE CODE, CAN YOU FIND IT???

Let’s analyse the signal

- After fixing the bug, let’s check that signal peaks at the expected B^0 mass. What’s the experimental resolution on M?



A note on the histogram errors

- By default, for a bin with N entries, root calculates the uncertainty as $\sqrt{N}$: for each bin, it does store only the information of N.

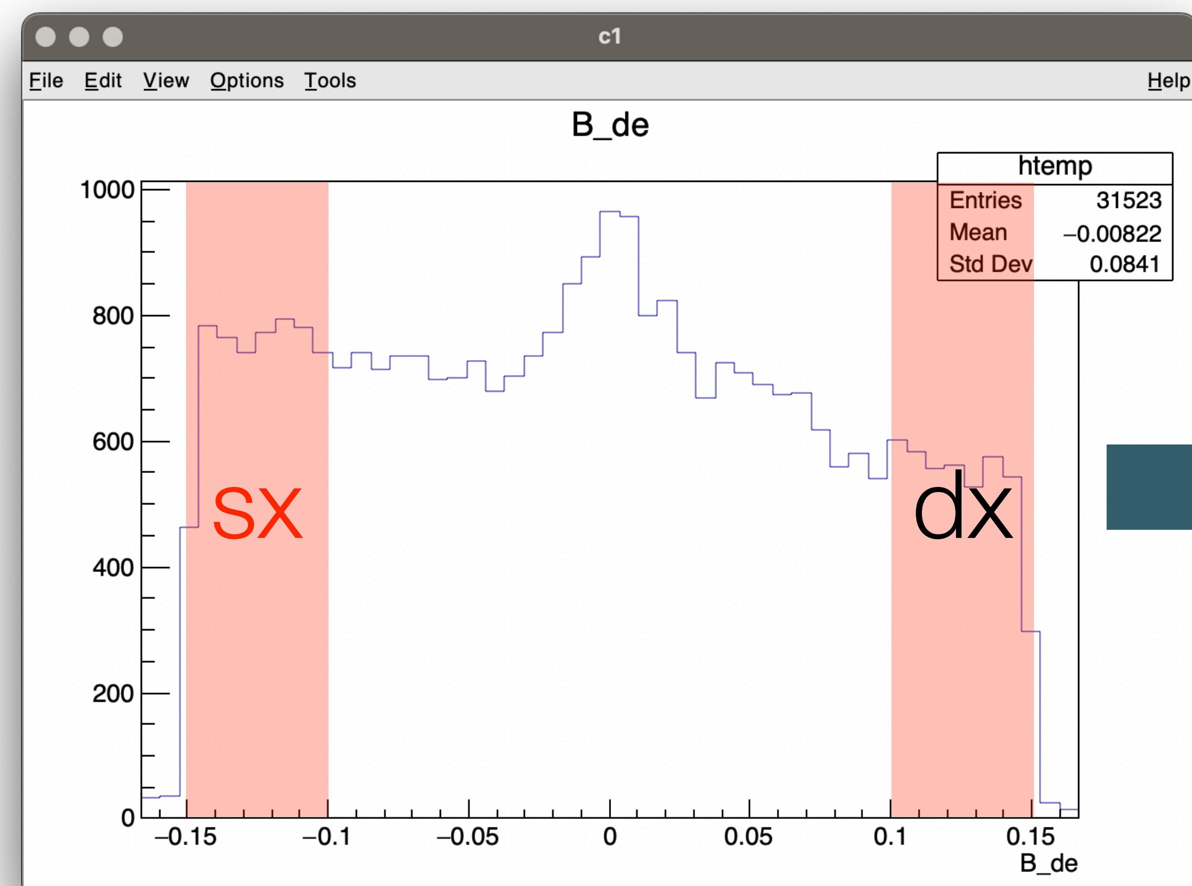
- `Sumw2()` enables to store also the *sum of squares of the weights*: it corresponds to save for any bin the information $\left(\sum w_i, \sum_i w_i^2 \right)$, which is (entry, error^2).
  For simple counts, $w_i = 1$: we save (N, N).

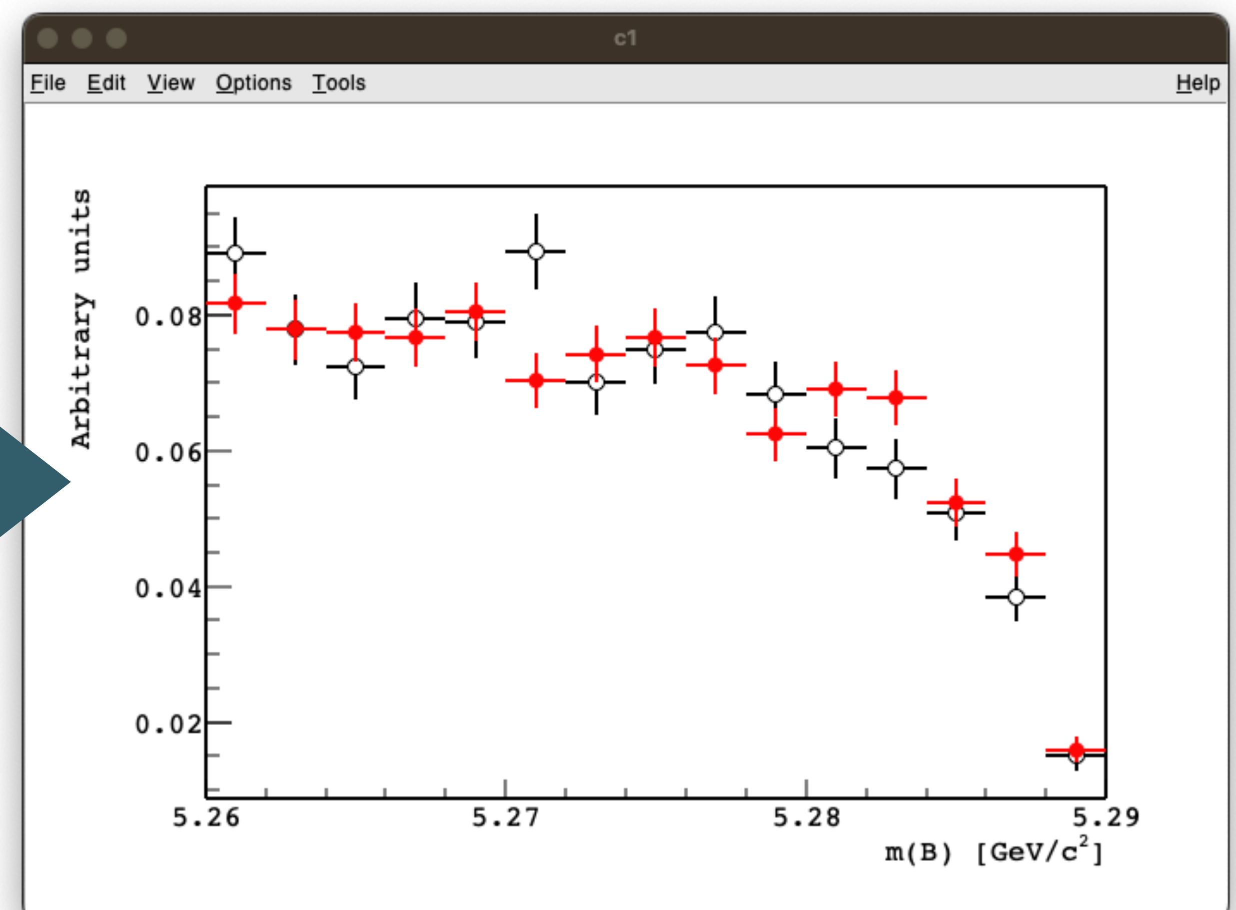
- When we do operations with histograms, root will do the correct propagation of uncertainty. For instance, for the subtraction $M - N$, the uncertainty is $\sqrt{M + N}$, instead of $\sqrt{M - N}$.

Other very useful operations for data analysis

- To compare the shape of two distributions, it's common to normalise them to the same (unit) integral and to plot them on the same canvas. For this, you can use Scale().

```
h_m_sx->Scale(1./h_m_sx->Integral());
h_m_dx->Scale(1./h_m_dx->Integral());

h_m_dx->Draw();
h_m_sx->Draw("same");
```



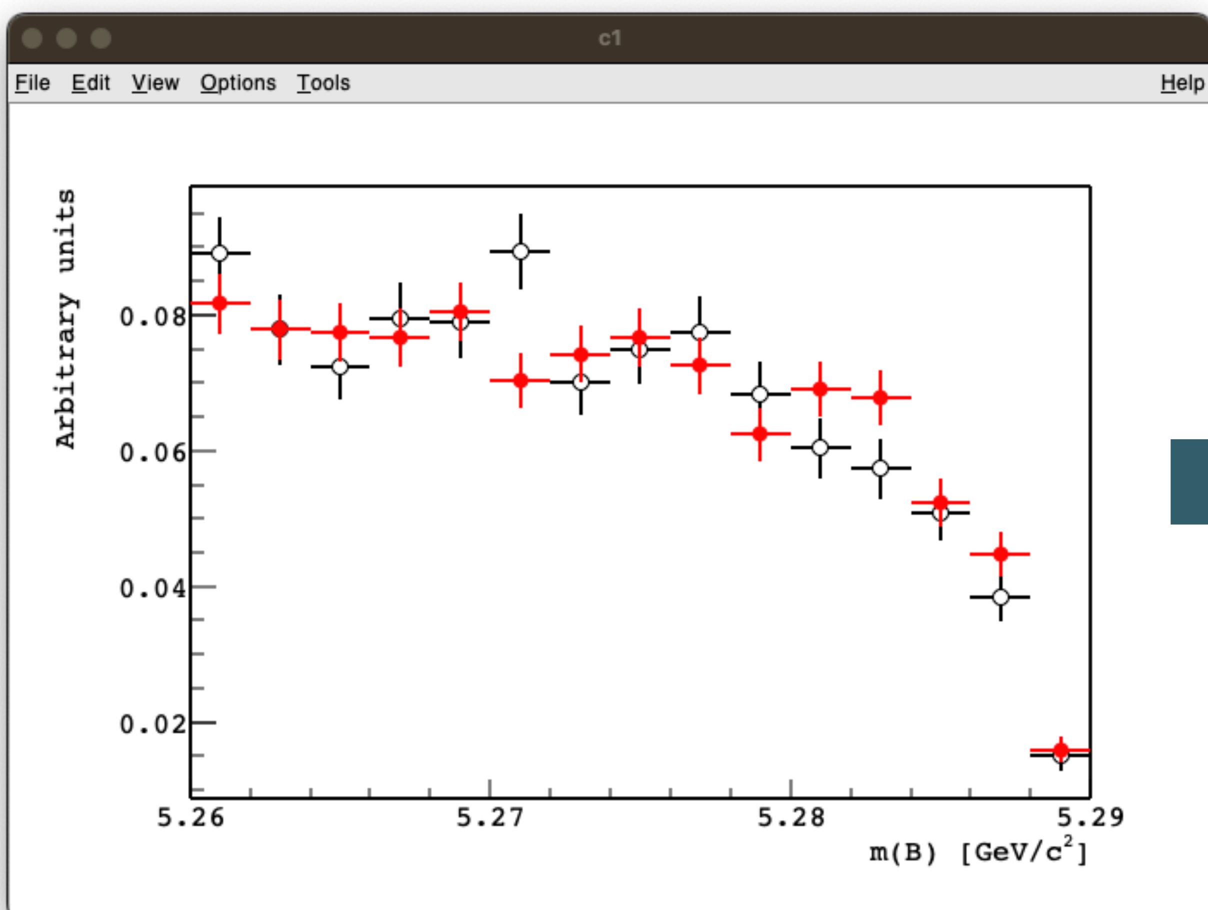


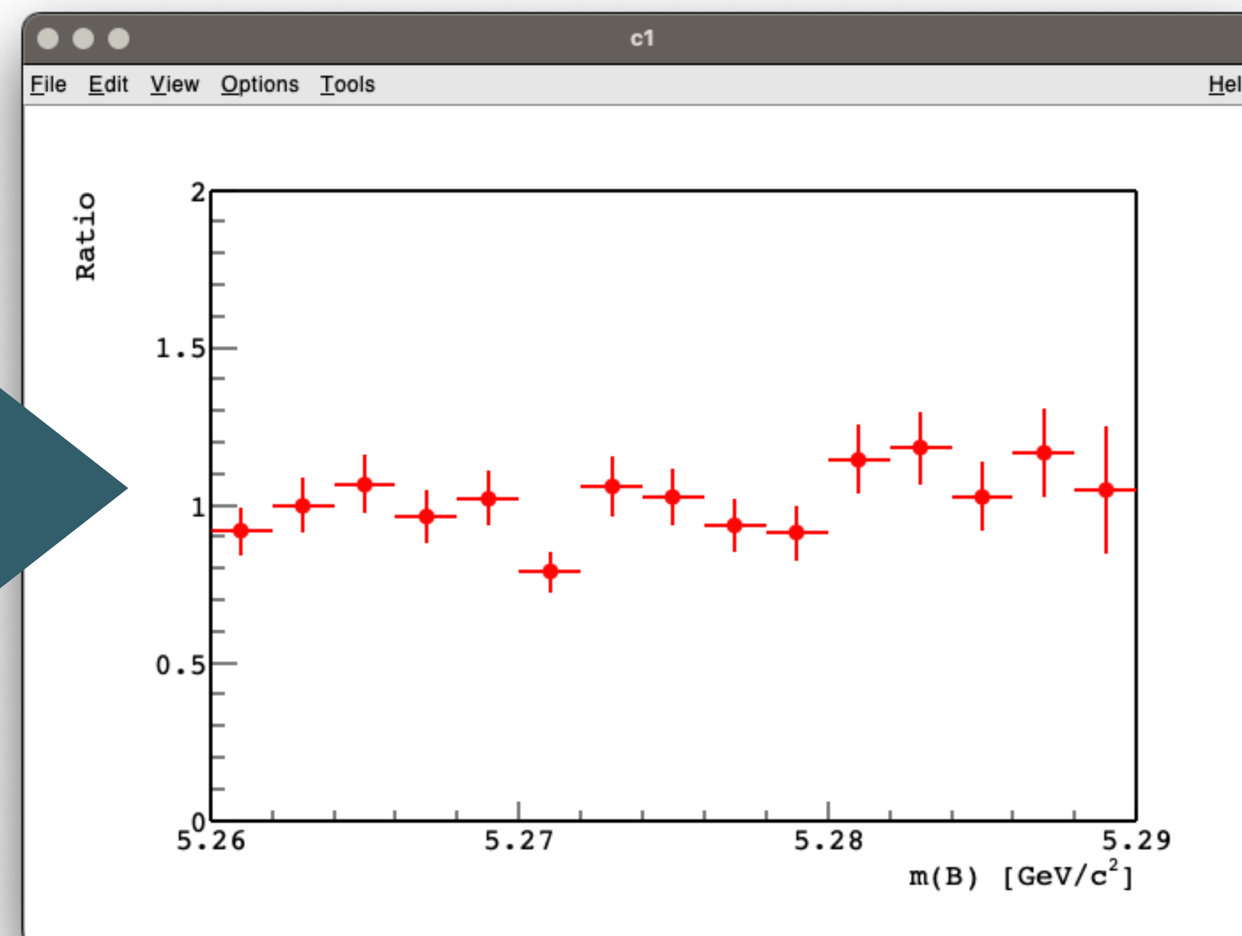
Other very useful operations for data analysis

- `Divide()` provides more quantitative information: if the data belongs to the same parent distribution, the ratio of the histograms is flat (you can even fit the ratio with a line to assess flatness)

```
h_m_sx->Scale(1./h_m_sx->Integral());
h_m_dx->Scale(1./h_m_dx->Integral());

h_m_dx->Draw();
h_m_sx->Draw("same");
```

```
h_m_sx->Divide(h_m_dx);
h_m_sx->Draw();
```





Take home messages

1. We learnt how to inspect data through distributions (histograms), 1D and 2D. Can do it “interactively” or in a script.

2. We know how to make fancy plots. **Make sure that your plot clearly shows the message you want to convey:** the content must be right and the format is important (visible data/titles/numbers/labels/legend…)

3. **Root by default sets bin errors as sqrt of the entries.** For proper error propagations, use `Sumw2()`.

4. To compare distributions, normalised them to the same (unit) area and make ratios.

Exercises

1. Modify histoPeak.C to plot the M distributions of the left and right ΔE sidebands. Compare the two distributions: plot them normalised in the same canvas and plot their ratios. Fit the ratio with a pol0 and a pol1 using the DrawPanel and comment the results

2. Obtain the ΔE signal distribution. To do that, proceed similarly to what we did in class: subtract the background from a signal-region histogram. To define the signal and background events, use:
signal for $M > 5.275\,\mathrm{GeV/c^2}$; background for $M < 5.275\,\mathrm{GeV/c^2}$.
When subtracting the background histogram, scale its integral by 0.4.