Exercise/Lesson #5

Scientific Data Analysis Lab course

Alexis Pompili - UniBA

The mass resolution of the $\psi' \to \mu^+\mu^-$ signal is a function of the rapidity y

(as mentioned in Exercise 4).

(See appendix: $y = \frac{1}{2}\ln\frac{E+p_z}{E-p_z} = \frac{1}{2}\ln\frac{1+\beta\cos\theta}{1-\beta\cos\theta}$)

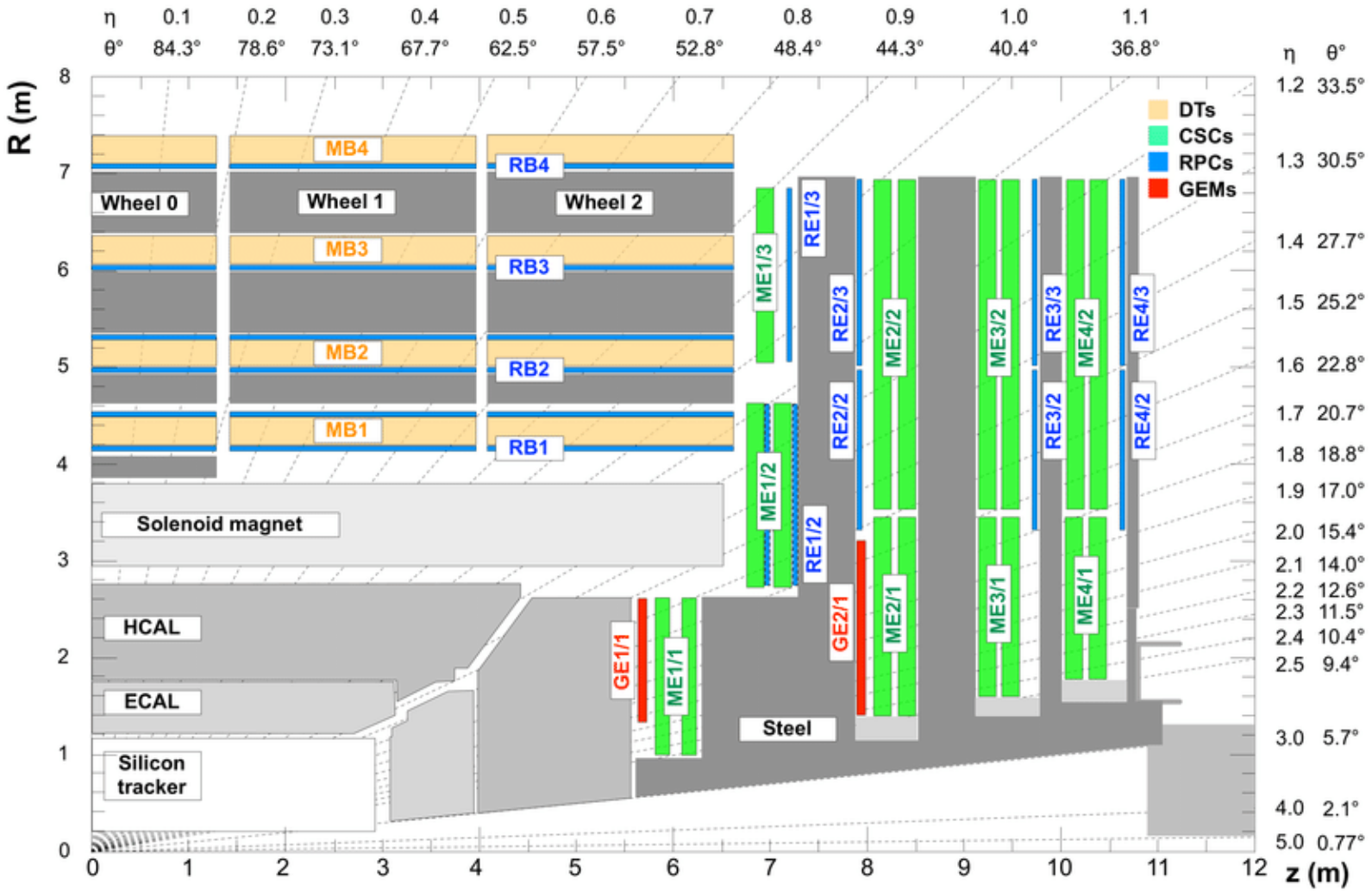
We consider the following **rapidity bins**:

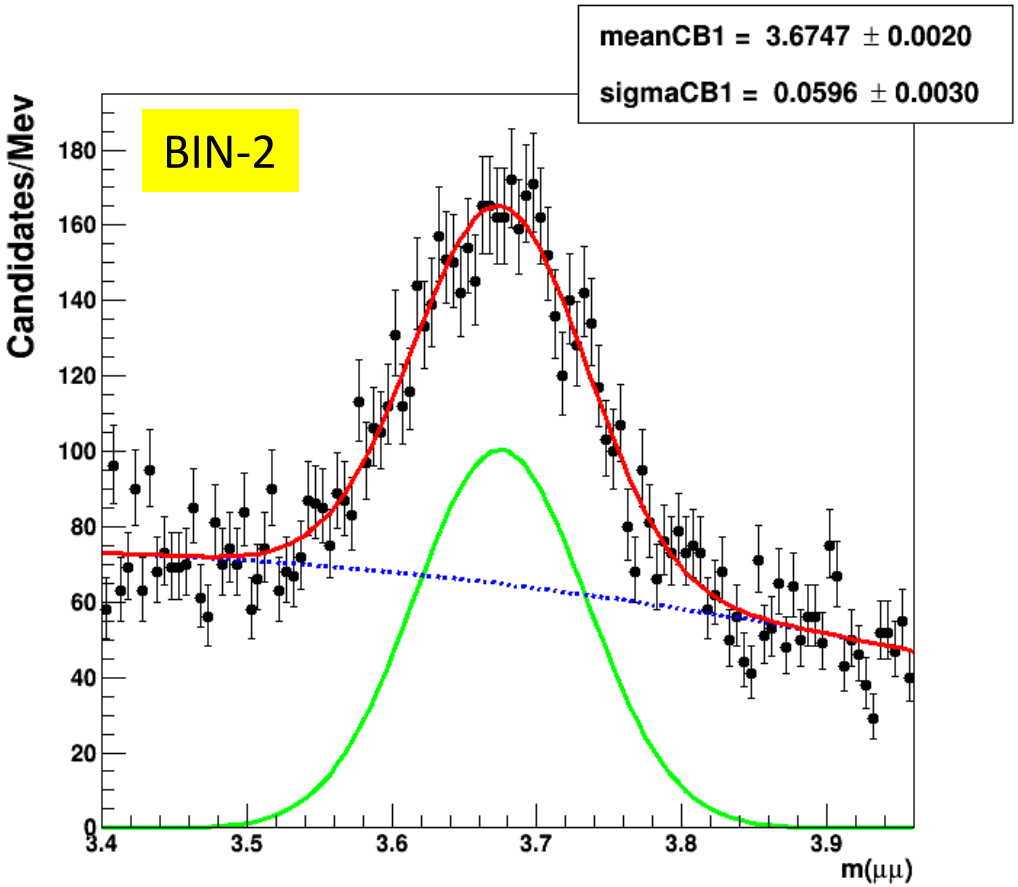
Bin	Rapidity	Bin	Rapidity
1	-2.4 / -2.2	13	0.0 / +0.2
2	-2.2 / -2.0	14	+0.2 / +0.4
3	-2.0 / -1.8	15	+0.4 / +0.6
4	-1.8 / -1.6	16	+0.6 / +0.8
5	-1.6 / -1.4	17	+0.8 / +1.0
6	-1.4 / -1.2	18	+1.0 / +1.2
7	-1.2 / -1.0	19	+1.2 / +1.4
8	-1.0 / -0.8	20	+1.4 / +1.6
9	-0.8 / -0.6	21	+1.6 / +1.8
10	-0.6 / -0.4	22	+1.8 / +2.0
11	-0.4 / -0.2	23	+2.0 / +2.2
12	-0.2 / 0.0	24	+2.2 / +2.4

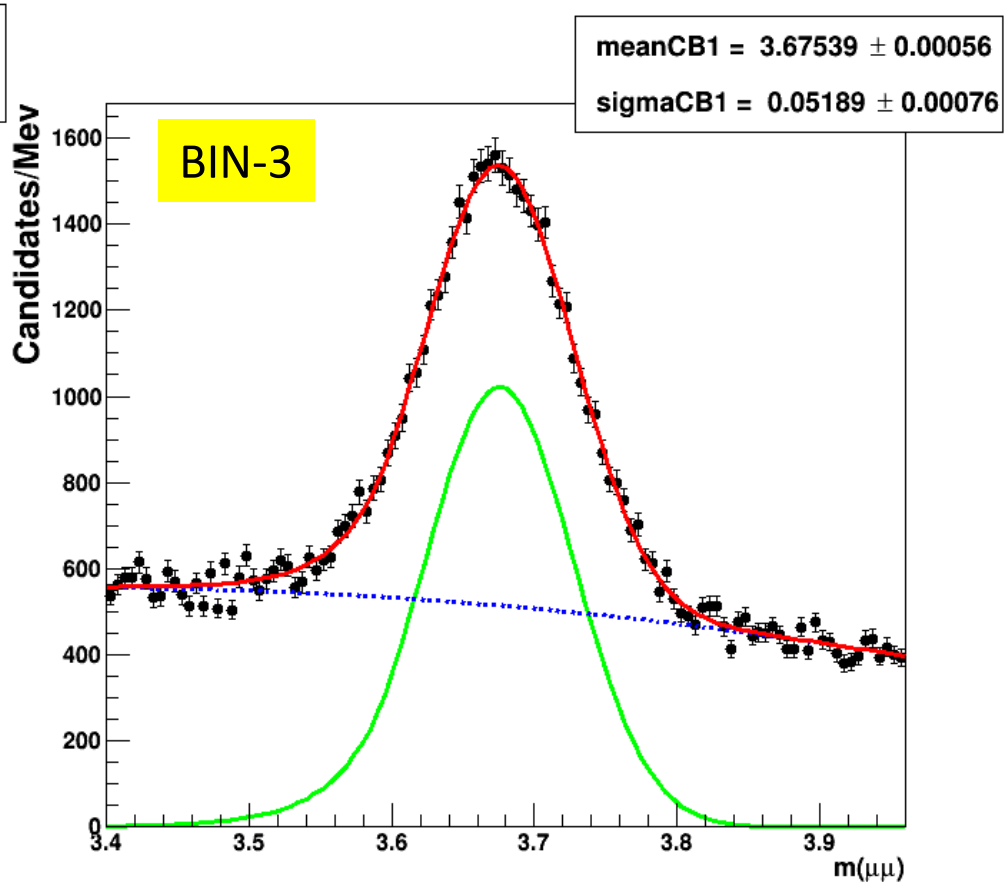
I will skip the two extreme bins due to low statistics (*overlined in red*)

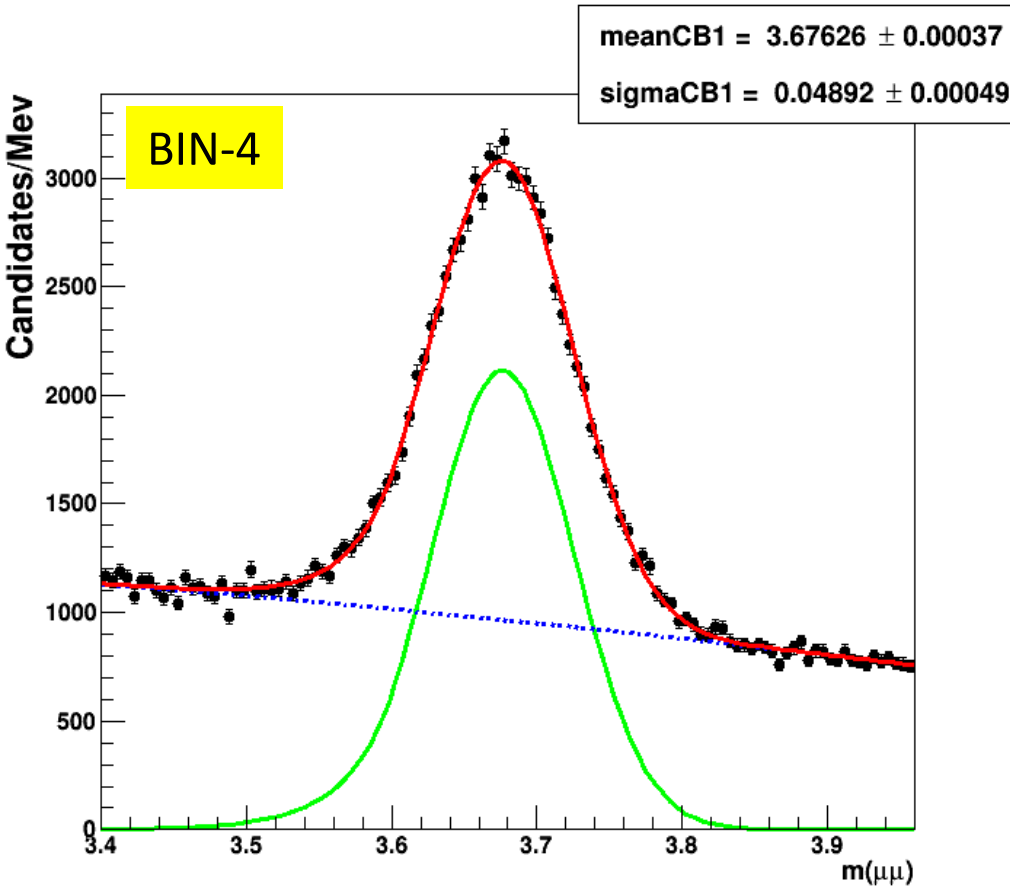
This is a quadrant of the CMS detector showing the subdetectors of the muon system (including the proposed GEM detectors):

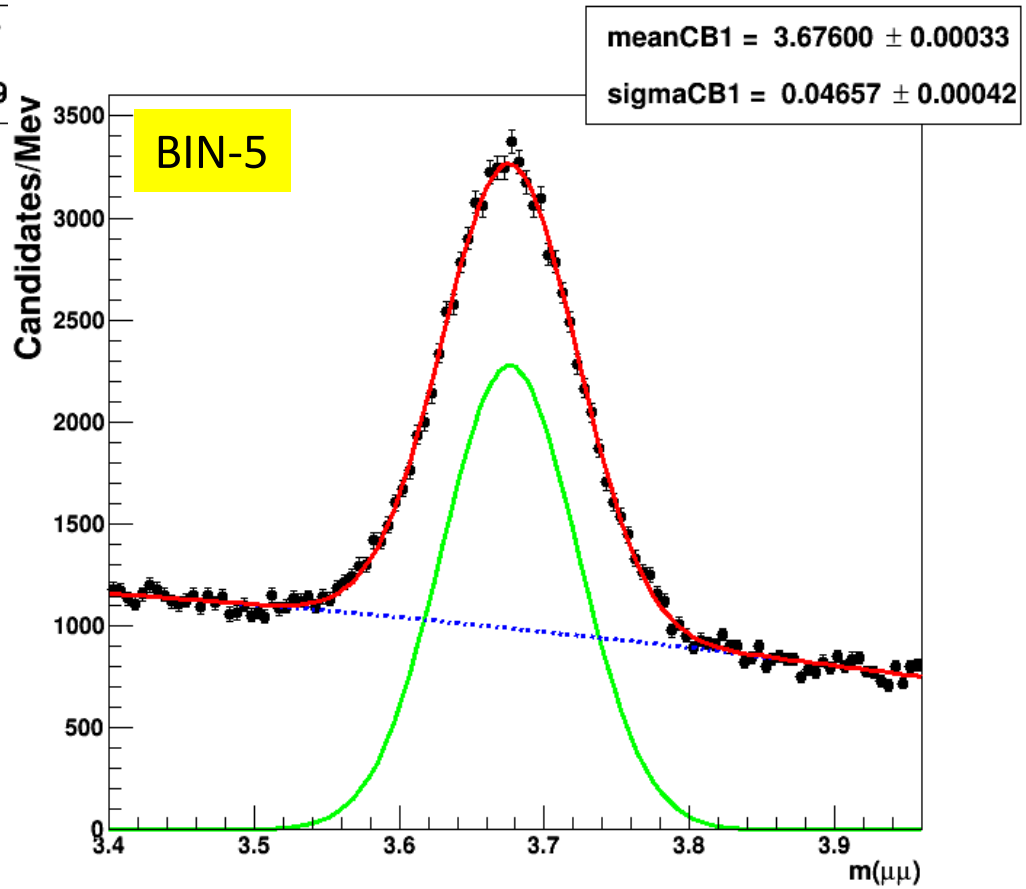
[note: pseudorapidity is the rapidity for *massless* particles: $y = \frac{1}{2}\ln\frac{1+\beta\cos\theta}{1-\beta\cos\theta} \rightarrow \frac{1}{2}\ln\frac{1+\cos\theta}{1-\cos\theta} = -\ln\tan\frac{\theta}{2} = \eta$]

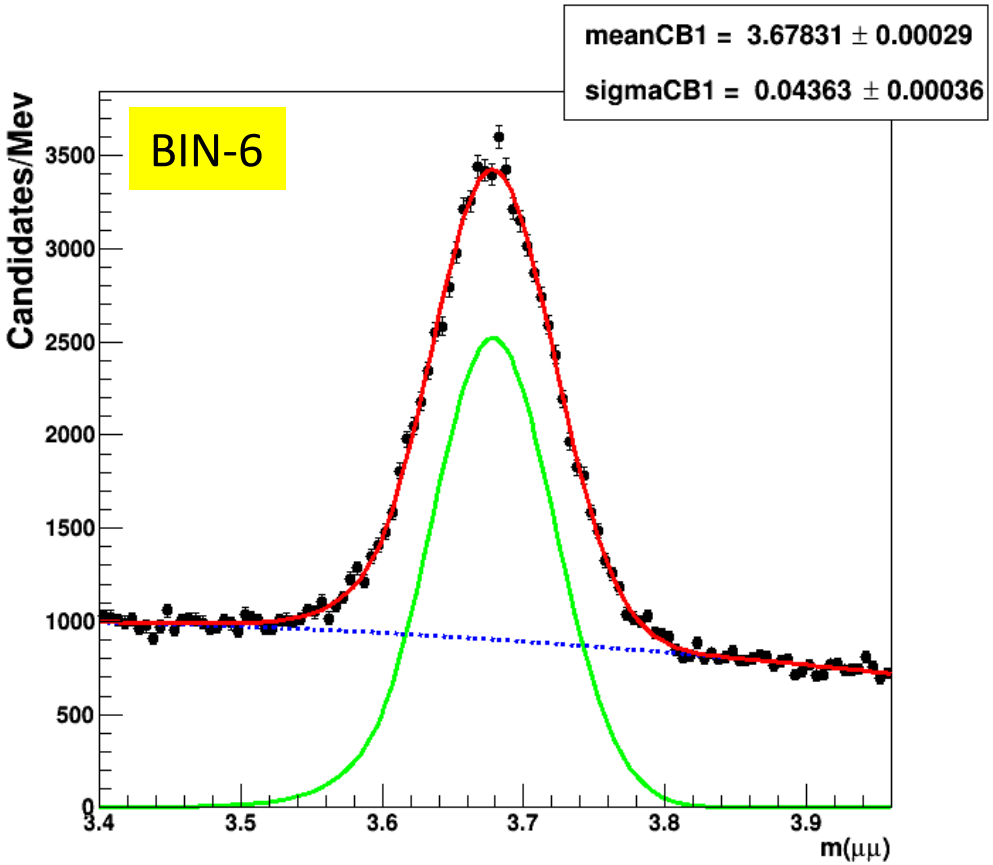


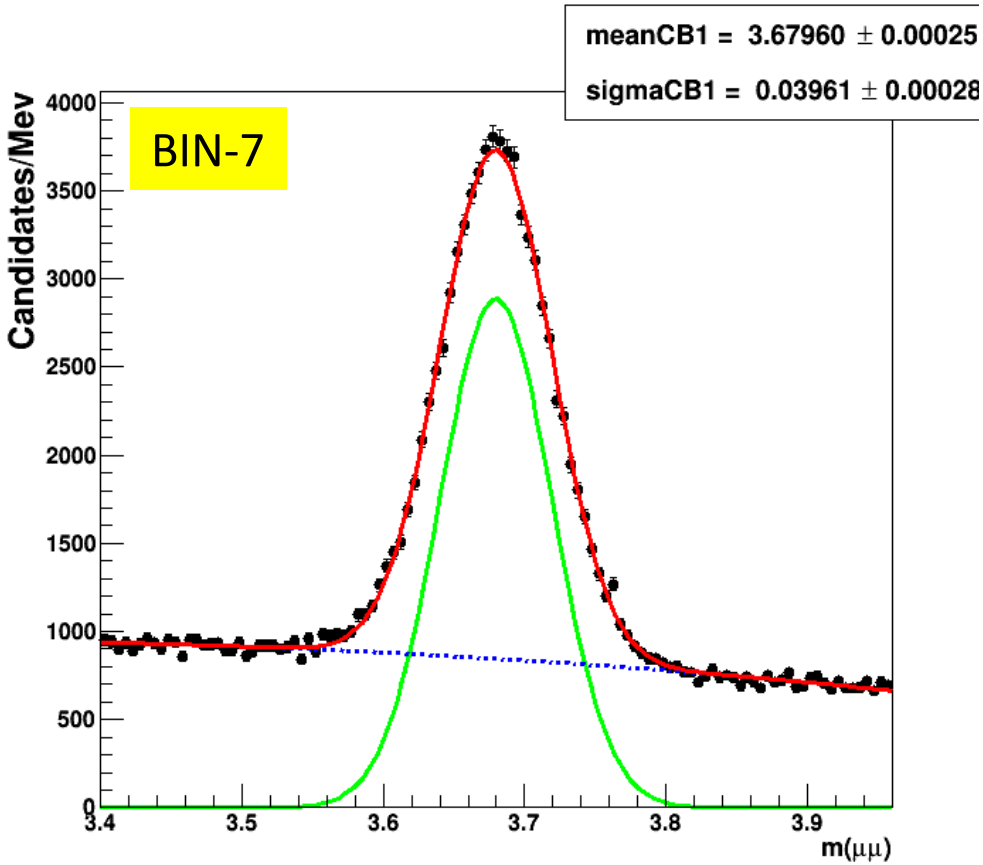

meanCB1 = 3.6747 ± 0.0020
sigmaCB1 = 0.0596 ± 0.0030
BIN-2
Candidates/Mev
m(μμ)

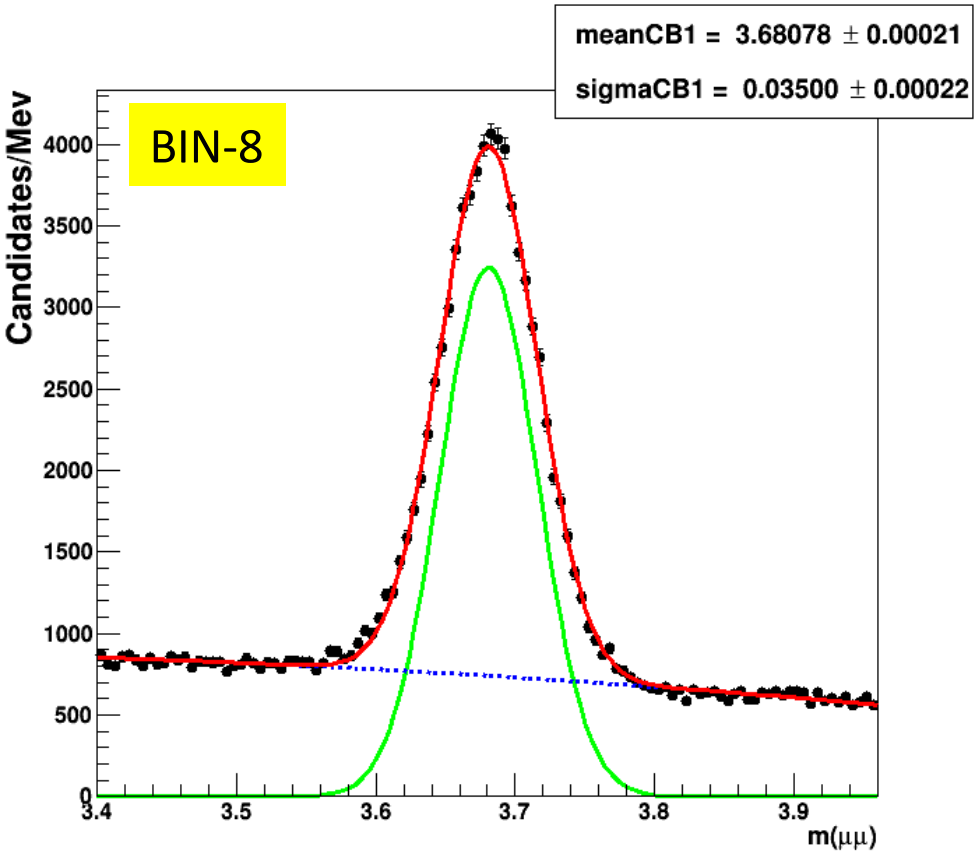

meanCB1 = 3.67539 ± 0.00056
sigmaCB1 = 0.05189 ± 0.00076
BIN-3
Candidates/Mev
m(μμ)

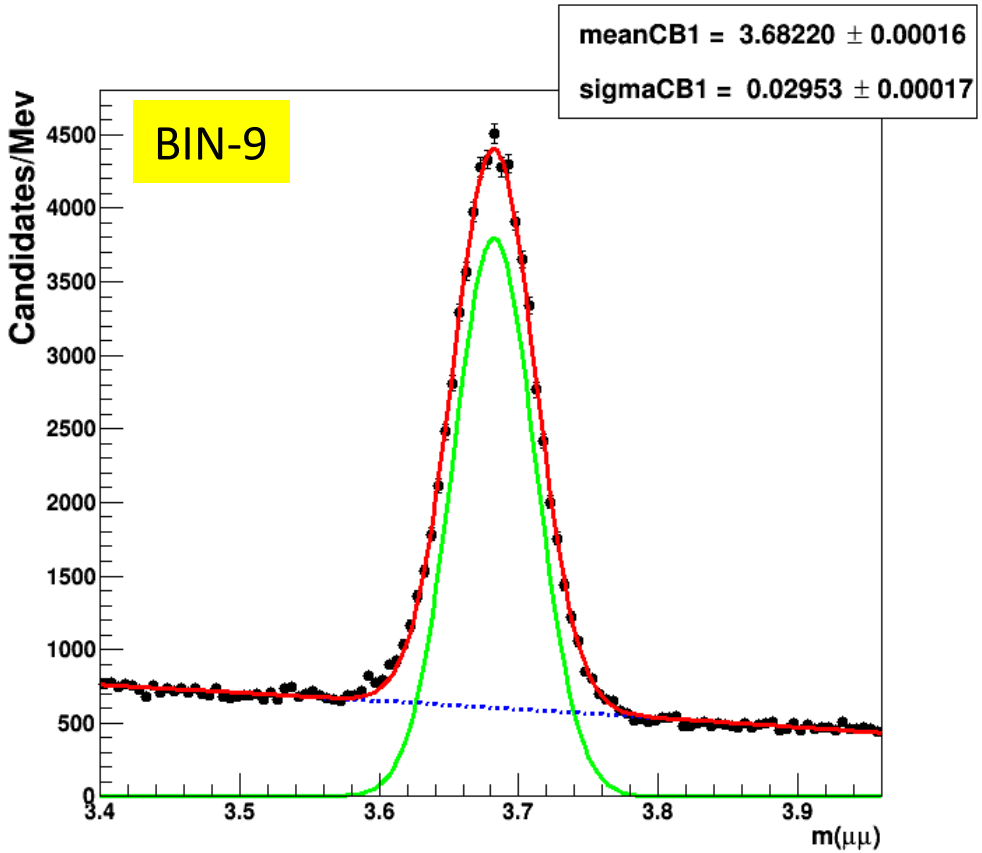

meanCB1 = 3.67626 ± 0.00037
sigmaCB1 = 0.04892 ± 0.00049
BIN-4
Candidates/Mev
m(μμ)

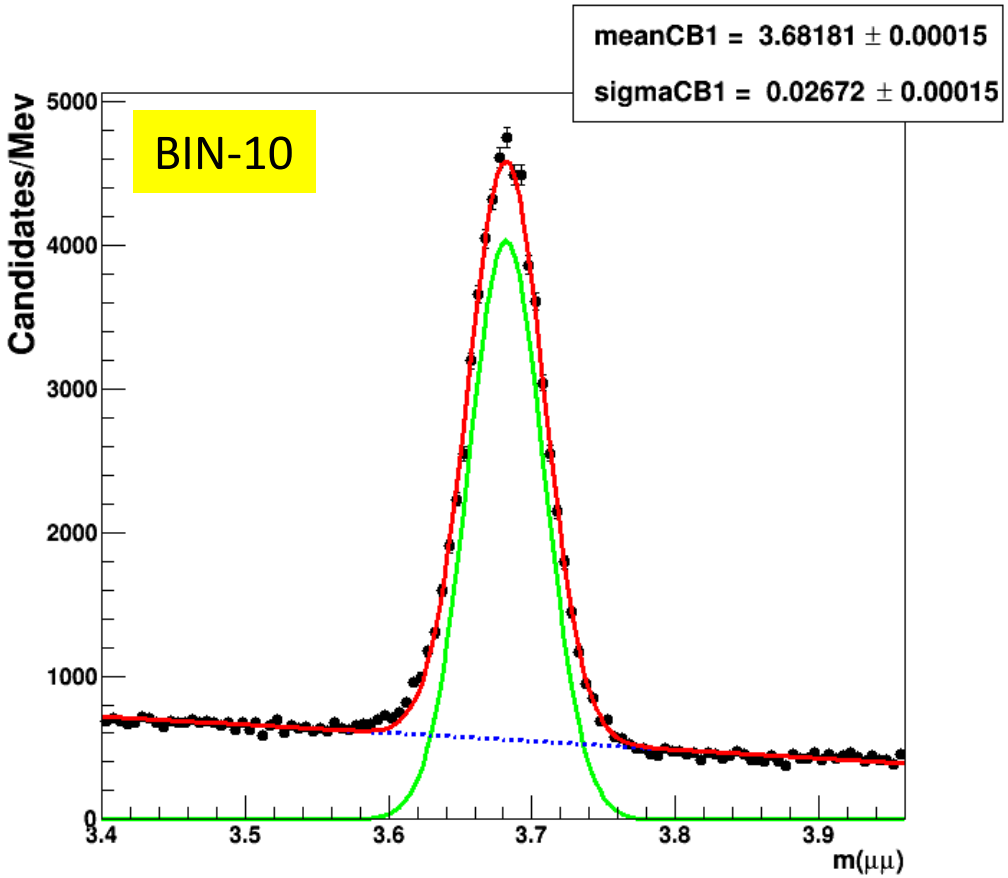

meanCB1 = 3.67600 ± 0.00033
sigmaCB1 = 0.04657 ± 0.00042
BIN-5
Candidates/Mev
m(μμ)

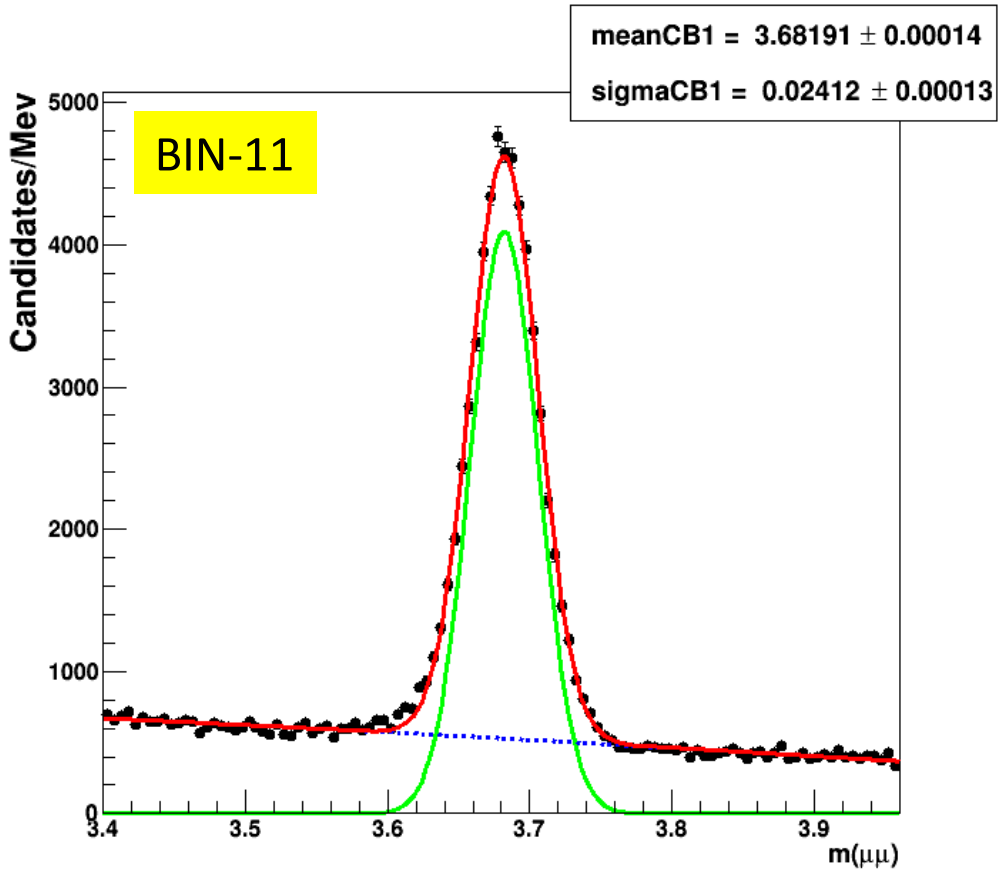

meanCB1 = 3.67831 ± 0.00029
sigmaCB1 = 0.04363 ± 0.00036
BIN-6
Candidates/Mev
m(μμ)

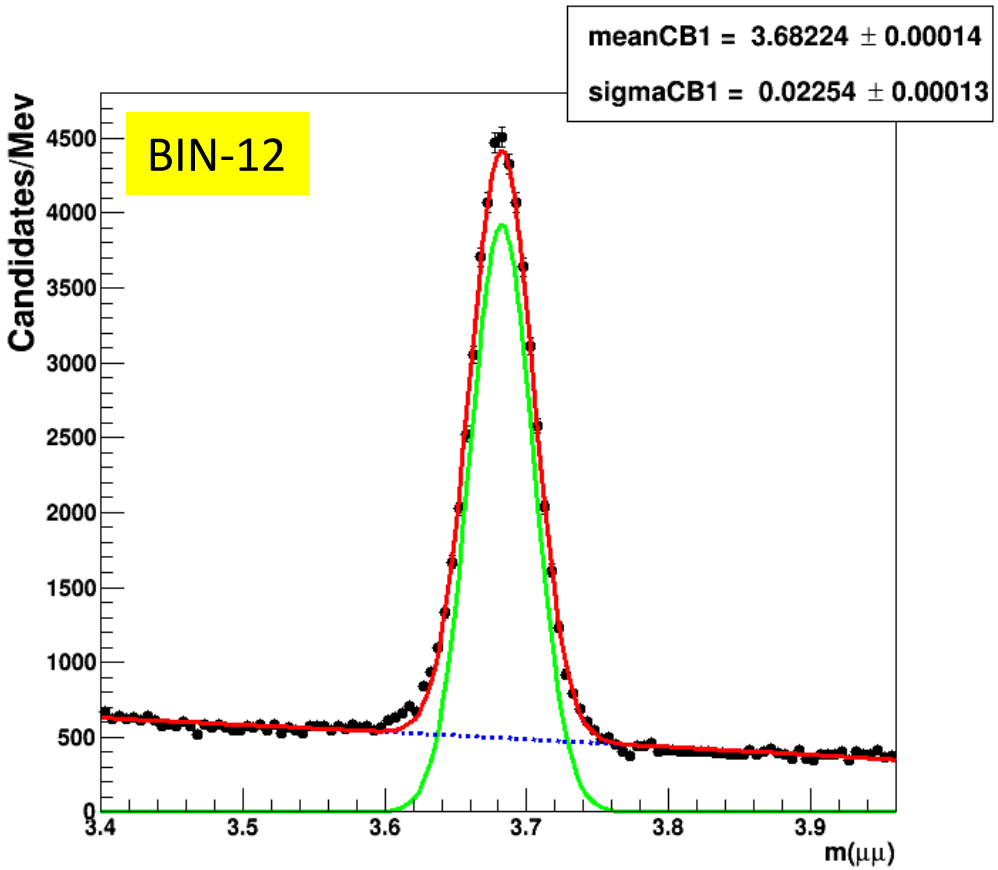

meanCB1 = 3.67960 ± 0.00025
sigmaCB1 = 0.03961 ± 0.00028
BIN-7
Candidates/Mev
m(μμ)

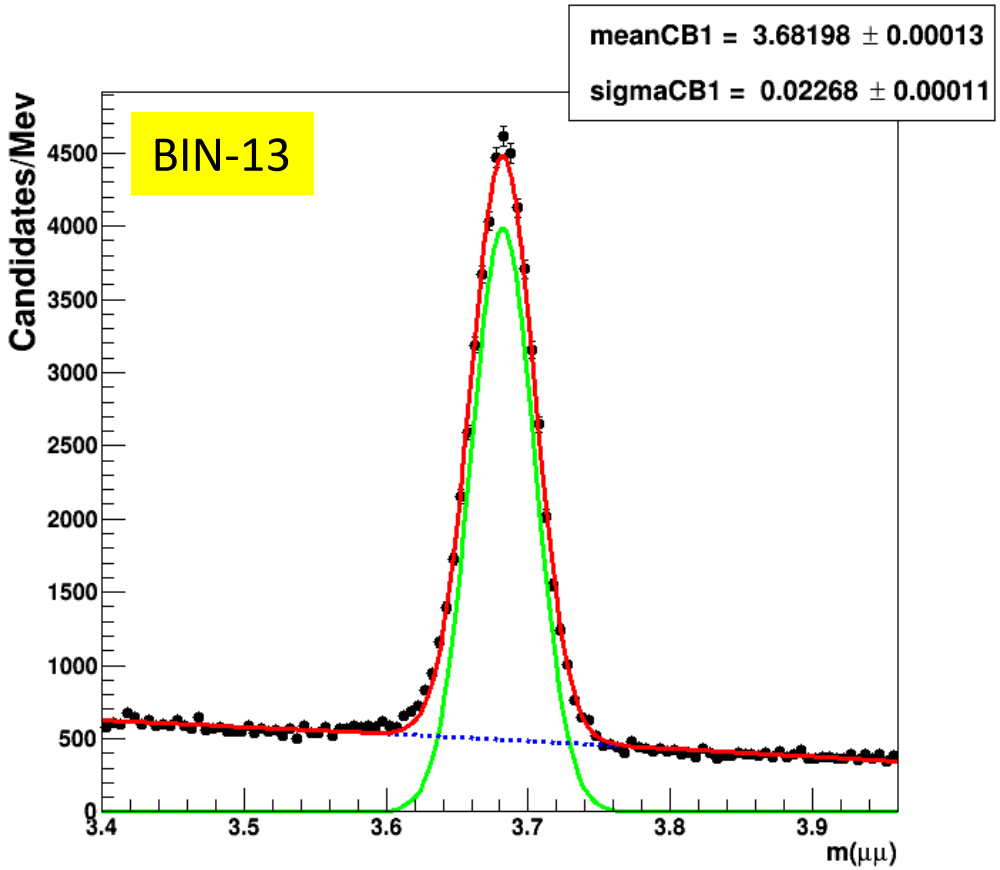

meanCB1 = 3.68078 ± 0.00021
sigmaCB1 = 0.03500 ± 0.00022
BIN-8
Candidates/Mev
m(μμ)

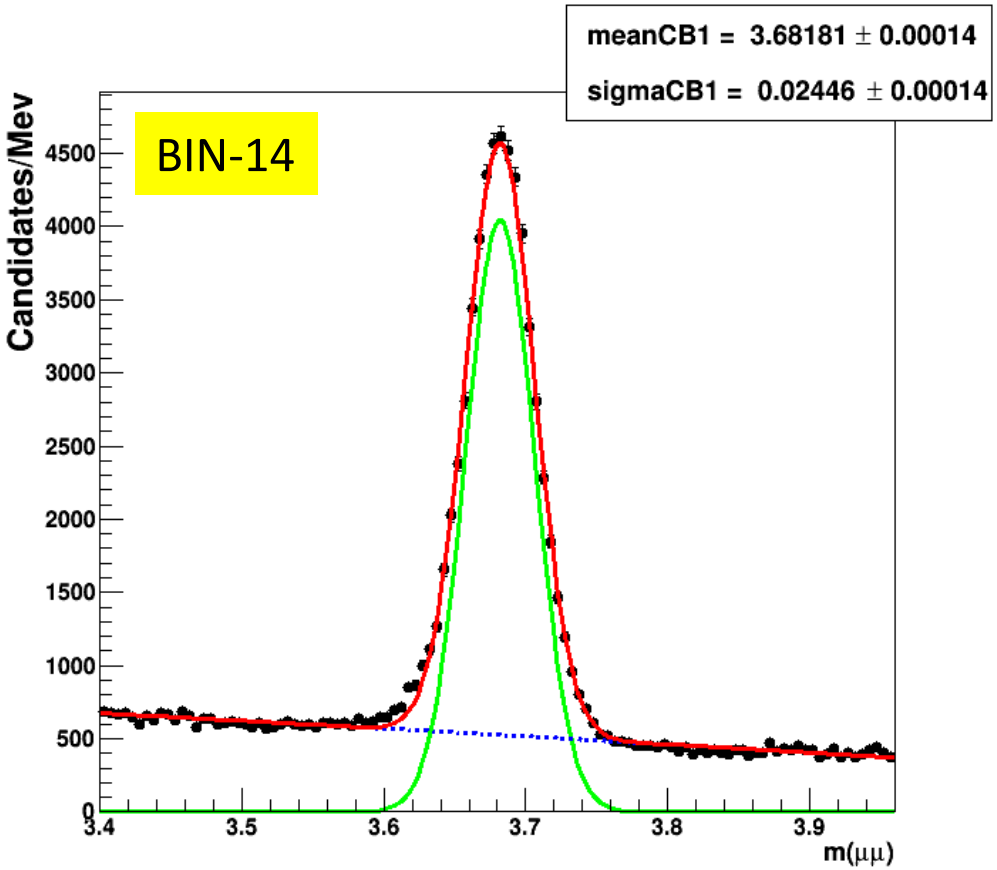

meanCB1 = 3.68220 ± 0.00016
sigmaCB1 = 0.02953 ± 0.00017
BIN-9
Candidates/Mev
m(μμ)

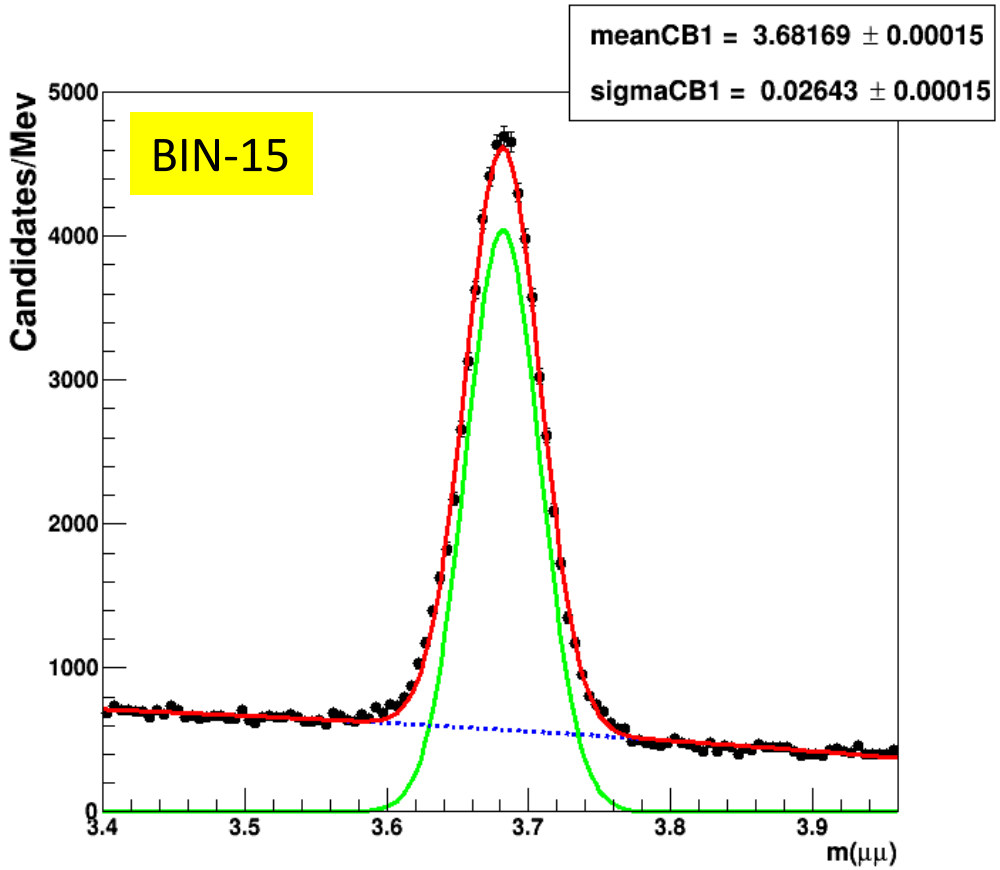

meanCB1 = 3.68181 ± 0.00015
sigmaCB1 = 0.02672 ± 0.00015
BIN-10
Candidates/Mev
5000
4000
3000
2000
1000
0
3.4
3.5
3.6
3.7
3.8
3.9
m(μμ)

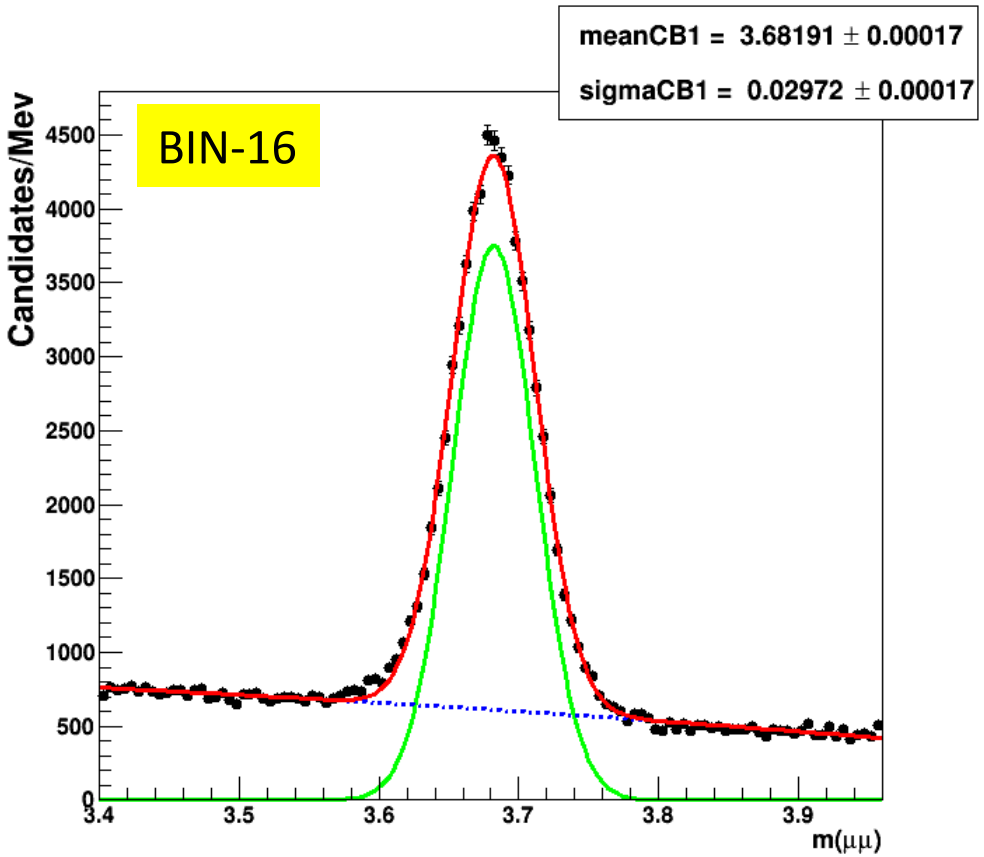

meanCB1 = 3.68191 ± 0.00014
sigmaCB1 = 0.02412 ± 0.00013
BIN-11
Candidates/Mev
5000
4000
3000
2000
1000
0
3.4
3.5
3.6
3.7
3.8
3.9
m(μμ)

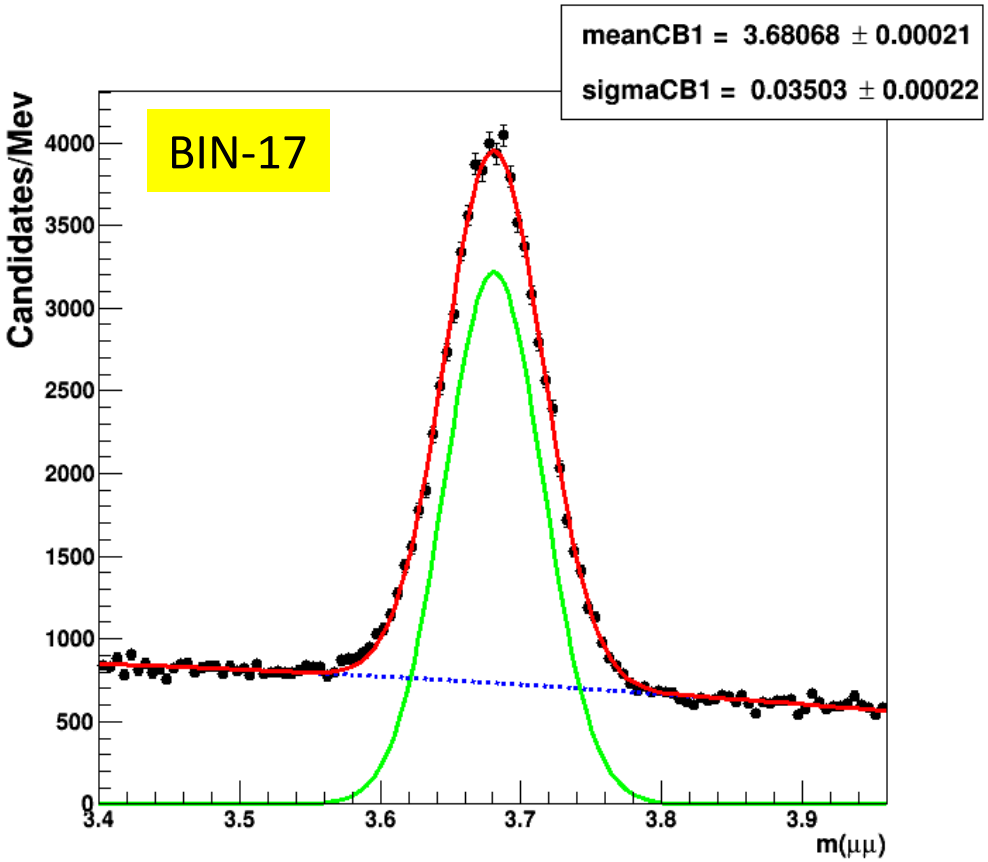

meanCB1 = 3.68224 ± 0.00014
sigmaCB1 = 0.02254 ± 0.00013
BIN-12
Candidates/Mev
m(μμ)

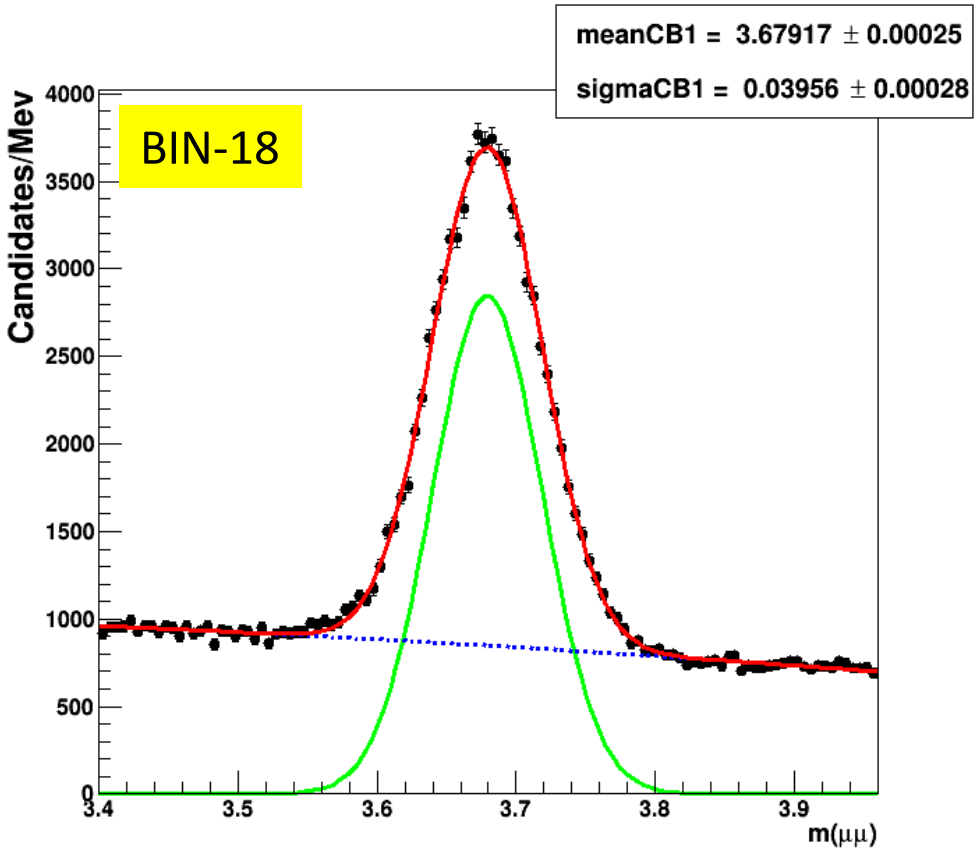

meanCB1 = 3.68198 ± 0.00013
sigmaCB1 = 0.02268 ± 0.00011
BIN-13
Candidates/Mev
m(μμ)

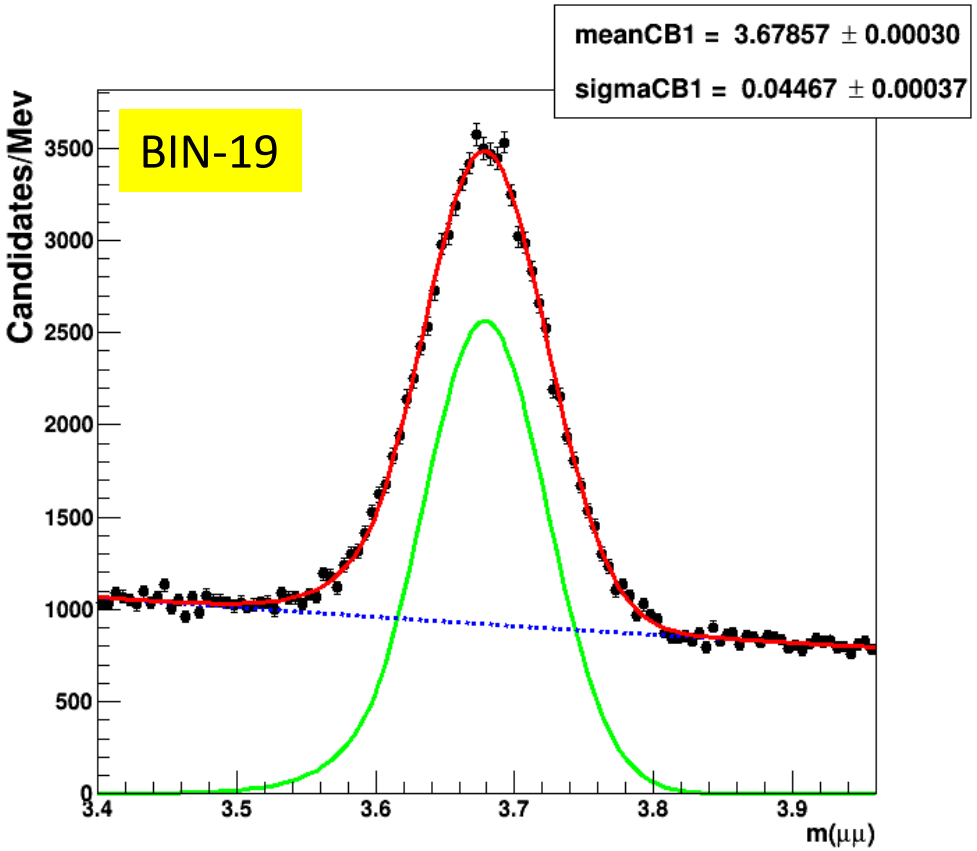

meanCB1 = 3.68181 ± 0.00014
sigmaCB1 = 0.02446 ± 0.00014
BIN-14
Candidates/Mev
m(μμ)

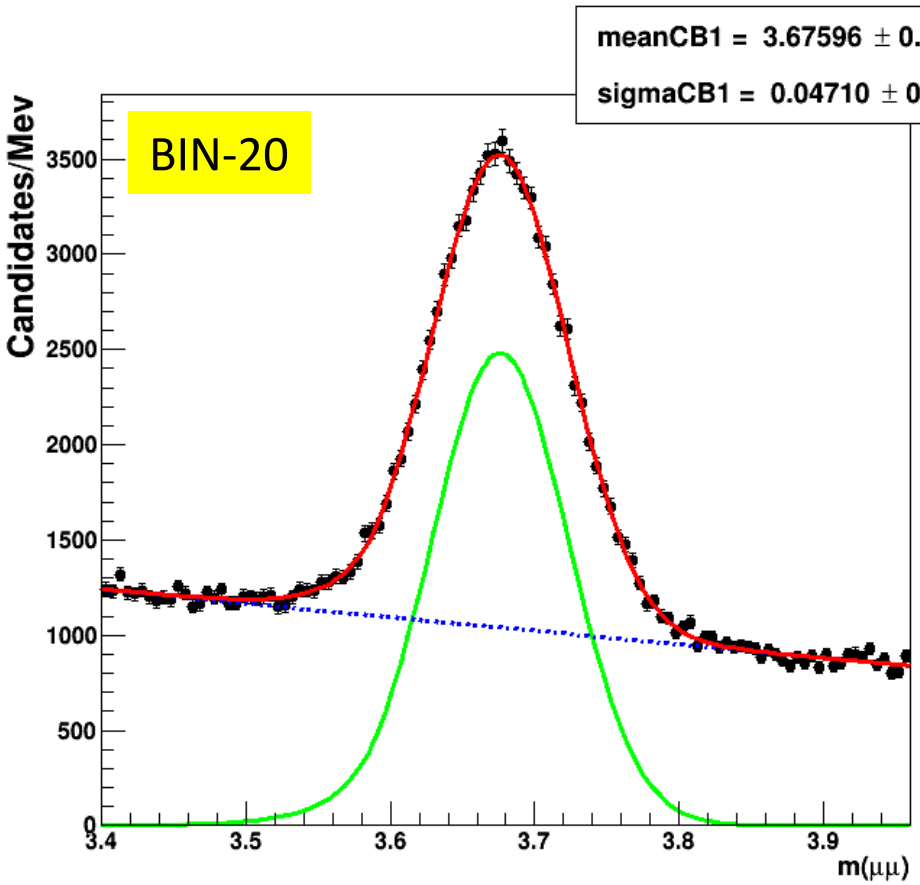

meanCB1 = 3.68169 ± 0.00015
sigmaCB1 = 0.02643 ± 0.00015
BIN-15
Candidates/Mev
m(μμ)

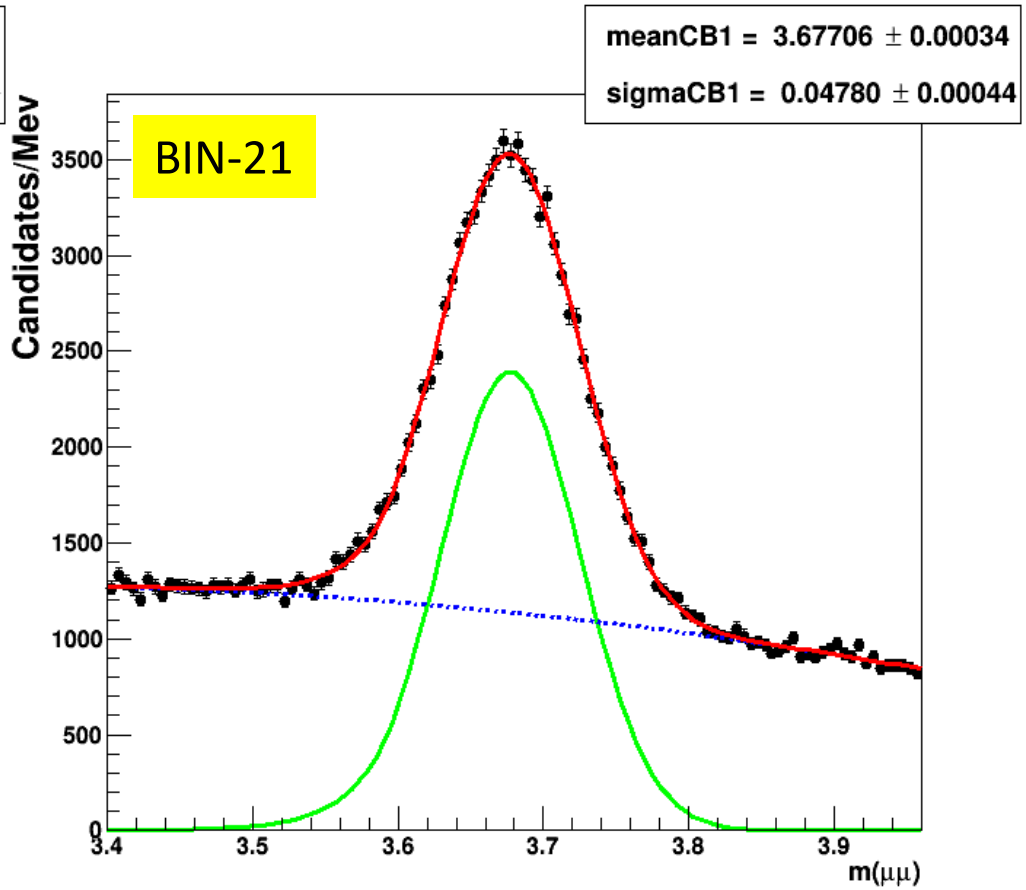

meanCB1 = 3.68191 ± 0.00017
sigmaCB1 = 0.02972 ± 0.00017
BIN-16
Candidates/Mev
m(μμ)

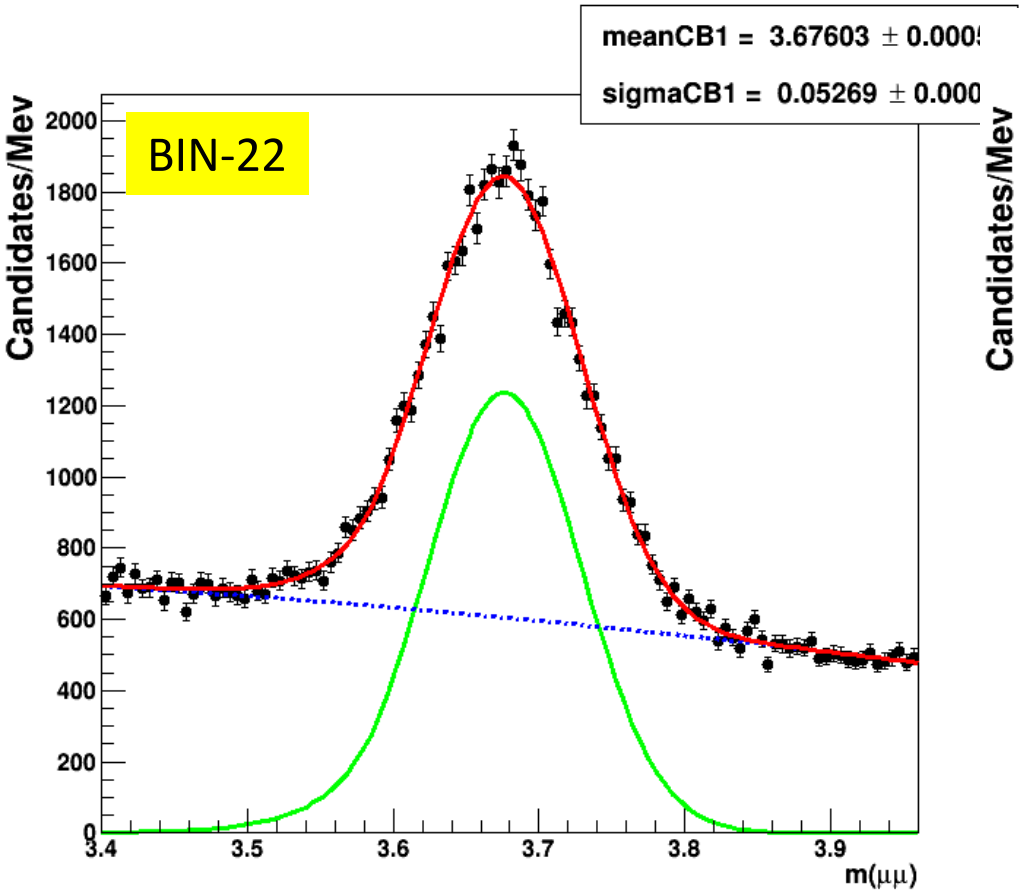

meanCB1 = 3.68068 ± 0.00021
sigmaCB1 = 0.03503 ± 0.00022
BIN-17
Candidates/Mev
m(μμ)

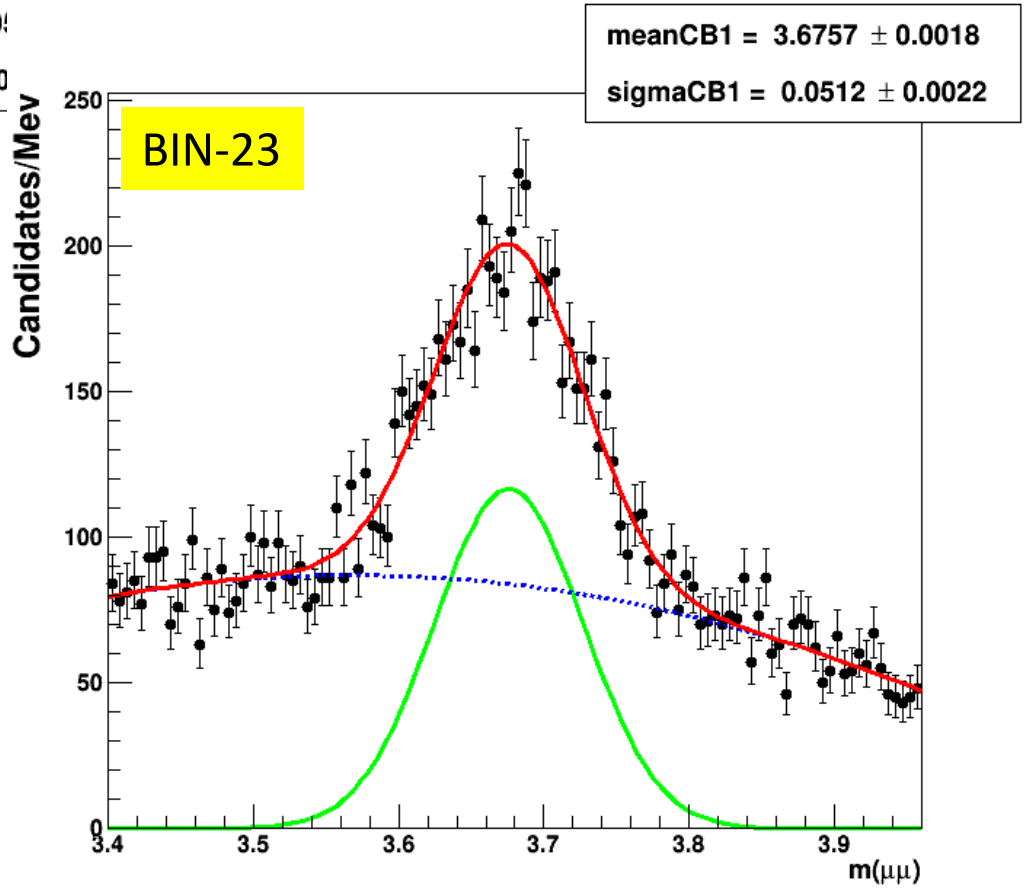

meanCB1 = 3.67917 ± 0.00025
sigmaCB1 = 0.03956 ± 0.00028
BIN-18
Candidates/Mev
m(μμ)


meanCB1 = 3.67857 ± 0.00030
sigmaCB1 = 0.04467 ± 0.00037
BIN-19
Candidates/Mev
m(μμ)

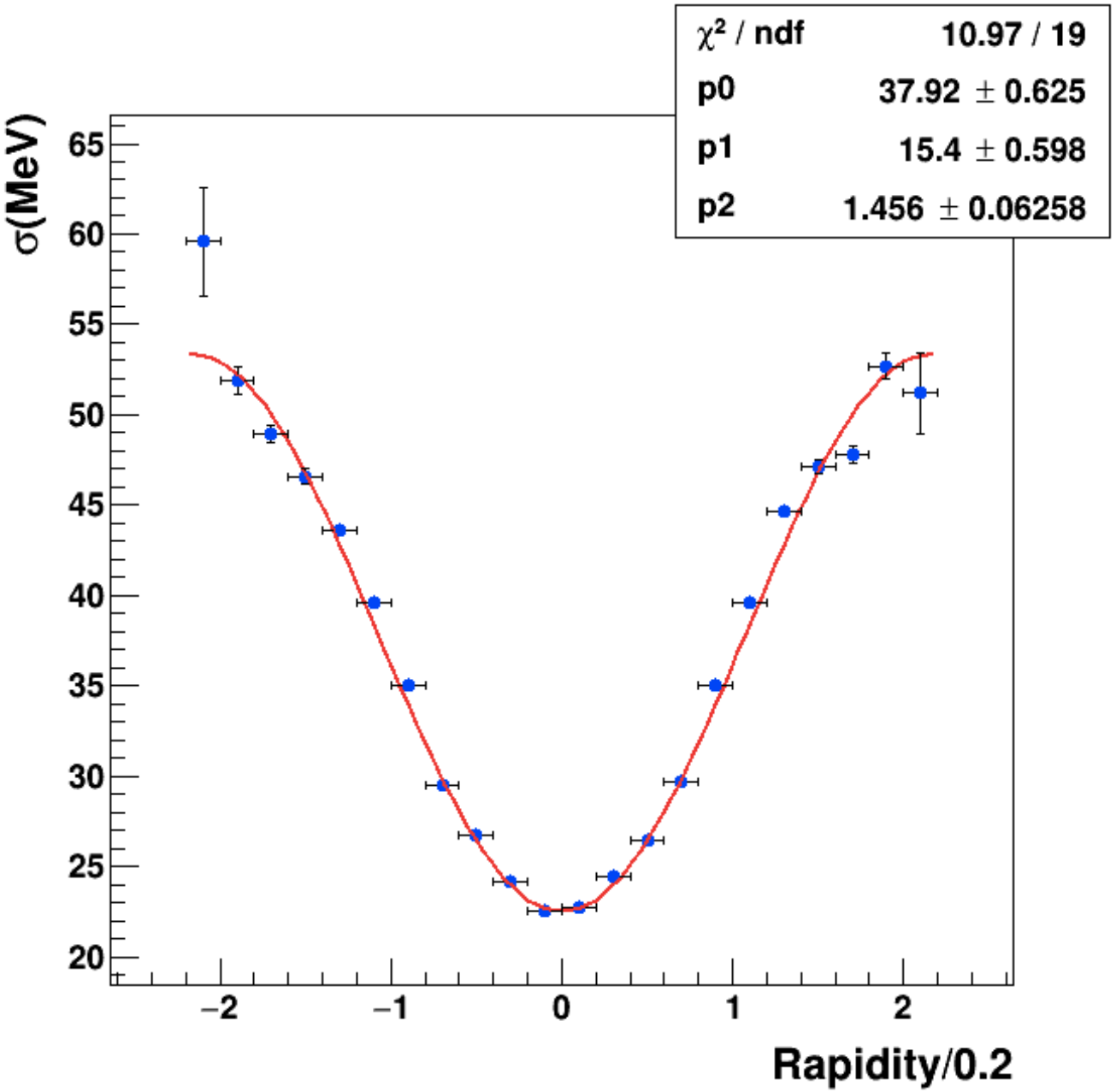

meanCB1 = 3.67596 ± 0.00032
sigmaCB1 = 0.04710 ± 0.00040
BIN-20
Candidates/Mev
m(μμ)


meanCB1 = 3.67706 ± 0.00034
sigmaCB1 = 0.04780 ± 0.00044
BIN-21
Candidates/Mev
m(μμ)


meanCB1 = 3.67603 ± 0.000
sigmaCB1 = 0.05269 ± 0.000
BIN-22
Candidates/Mev
m(μμ)


meanCB1 = 3.6757 ± 0.0018
sigmaCB1 = 0.0512 ± 0.0022
BIN-23
Candidates/Mev
m(μμ)

The final plot is fitted with the function: $f(x) = p0 - p1 * \cos(p2 * x)$



Let’s see the macro code used to produce the previous result:

```
#include <TROOT.h>
#include <TFile.h>
#include <TF1.h>
#include <TH1.h>
#include <TF2.h>
#include <TFormula.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TProfile.h>
#include <TString.h>
#include <TLine.h>
#include <TPad.h>
#include <TMath.h>
#include <TLatex.h>
#include <TLegend.h>
#include <iostream>
#include <TColor.h>
#include <TAxis.h>
#include <RooGlobalFunc.h> // needed to resize the text in the statistics box of a frame

using namespace RooFit;

void myFinalRapidity(){

  // Operazioni solite di clear e reset
  gROOT->Reset();
  gROOT->Clear();

  // Opzioni di stile
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(10);

  // Dichiarazione file
  TFile f1("./hlt_5_newSoftMuon_alsoInPsiPrimeWind.root","READ");
  //
  TString numeri[22] =  {"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23"};
  //
  //-logic : bin-2=-2.2/-2.0 ; bin-3=-2.0/-1.8 ; bin-4=-1.8/-1.6 ; bin-5=-1.6/-1.4 ; bin-6=-1.4/-1.2 ; bin-7=-1.2/-1.0 ; bin-8=-1.0/-0.8 ; bin-9=-0.8/-0.6
  //         bin-10=-0.6/-0.4 ; bin-11=-0.4/-0.2 ; bin-12=-0.2,0.0 ; bin-13=0.0/0.2 ; bin-14=0.2/0.4 ; bin-15=0.4,0.6 ; bin-16=0.6/0.8
  //         bin-17=0.8/1.0 ; bin-18=1.0/1.2 ; bin-19:1.2/1.4 ; bin-20=1.4/1.6; bin-21=1.6/1.8 ; bin-22=1.8/2.0 ; bin-23=2.0/2.2
  //
  //         (skipping two extreme bins: bin-1=-2.4/-2.2 & bin-24=2.2/2.4 because if the uncertainty due to the low statistics)
  //
  //Canvas for my plots:
  //
  TCanvas *myCanvas = new TCanvas("myCanvas", "myCanvas", 700, 700);
  gPad->SetBottomMargin(0.15);
  gPad->SetLeftMargin(0.1);
  gPad->SetRightMargin(0.1);
  gPad->SetTopMargin(0.1);

  //Vettore per grafico
  double vec[22];
  double vec_err[22];
  // double vec_err_h[22]; // Se eventualmente usassi Minos
  // double vec_err_l[22]; // Se eventualmente usassi Minos
```

```
//
for (int i = 0; i<22; i++)
   {
    //Apertura file e lettura istogramma
    TH1D *histo = (TH1D*)f1.Get("PsiPrimeMass_bin"+numeri[i]);
    //
    // Crezione oggetto RooDataHist
    RooRealVar x("x", "x", 3.4, 3.96);
    RooDataHist *psiprime = new RooDataHist(histo->GetName(), histo->GetTitle(), RooArgSet(x), RooFit::Import(*histo, kFALSE));
    //
    //Frame for plot e opzioni di stile
    RooPlot *xframe = x.frame(Title(""));
    //xframe->SetTitle("#mu#mu invariant mass spectrum");
    xframe->SetTitle("");
    xframe->SetTitleOffset(1.32,"y");
    xframe->SetLabelSize(0.025, "y");
    xframe->SetTitleSize(0.038, "y");
    xframe->SetYTitle("Candidates/Mev"); //ATTENZIONE A VERO BINNAGGIO: VEDI DA TBrowser
    xframe->SetLabelSize(0.025, "x");
    xframe->SetTitleSize(0.029, "x");
    xframe->SetTitleOffset(0.93,"x");
    xframe->SetXTitle("m(#mu#mu)");
    //
    // Metto il RooDataHisto sul frame
    psiprime->plotOn(xframe);
    //
    // Tolgo titolo istogramma
    //char title[128]="";
    psiprime->SetTitle("");
    //
    //
    //---------------START FIT
    //
    // Signal peak
    RooRealVar meanCB1("meanCB1", "meanCB1", 3.675, 3.64, 3.72);
    RooRealVar sigmaCB1("sigmaCB1", "sigmaCB1", 0.040, 0.001, 0.700);
    RooRealVar alpha1("alpha1", "alpha1", 5.0, 0.01, 50.);
    RooRealVar nCB1("nCB1", "nCB1", 3.0, 0.001, 20.);
    RooCBShape myCB1("myCB1", "myCB1", x, meanCB1, sigmaCB1, alpha1, nCB1);
    //
    // Background shape
    //
    // Try Chebychev Poly of order2
    //
    RooRealVar c1("c1", "1st coeff", -0.30, -1000, 100); //originale -0.3
    RooRealVar c2("c2", "2nd coeff", 0.1, -1000, 100); // originale 0.01
    RooChebychev cheby("cheby", "Chebichev 2", x, RooArgList(c1,c2));
    //
    // Alternatively Background Exponential
    //RooRealVar aExp("aExp", "aExp", 0., -10000., 10000.);
    //RooExponential myExp("myExp", "myExp", x, aExp);
    //
    // Signal & Background yields
    RooRealVar nSig1("nSig1", "Number of first signal candidates", 4e+5, 1e+2, 1e+8);
    //RooRealVar nSig2("nSig2", "Number of second signal candidates", 4e+4, 1e+3, 1e+7);
    RooRealVar nBkg("nBkg", "Number of background candidates", 60e+4, 1e+2, 1e+8);
    //
    RooAddPdf *totalPdf = new RooAddPdf("totalPdf", "totalPdf", RooArgList(myCB1, cheby), RooArgList(nSig1, nBkg));
    //
    // RooAddPdf *totalPdf = new RooAddPdf("totalPdf", "totalPdf", RooArgList(myCB1, myCB2, myExp), RooArgList(nSig1, nSig2, nBkg));
    //
    // Fit command
    totalPdf->fitTo(*psiprime, Extended(kTRUE));
    //
```

```
    //
    // Filling vectors
    vec[i] = 1000.0*sigmaCB1.getValV(); //Moltiplico per 1000 per avere in MeV
    //vec_err_h[i]= sigmaCB1.getAsymErrorHi(); // Se uso MINOS per il fit
    //vec_err_l[i]= sigmaCB1.getAsymErrorLo(); // Se uso MINOS per il fit
    vec_err[i] = 1000*sigmaCB1.getError();
    //
    totalPdf->plotOn(xframe, RooFit::LineColor(kRed));
    totalPdf->plotOn(xframe, RooFit::Components(RooArgSet(myCB1)),LineColor(kGreen));
    totalPdf->plotOn(xframe, RooFit::Components(RooArgSet(cheby)), RooFit::LineStyle(kDashed));
    //
    // Plot con fondo esponenziale: commenta giu o su a seconda di cosa vuoi
    //totalPdf->plotOn(xframe, RooFit::Components(RooArgSet(myExp)), RooFit::LineStyle(kDashed));
    //
    // Ridisegno il full fit per fare correttamente le pull dopo
    totalPdf->plotOn(xframe, RooFit::LineColor(kRed));
    // Box parametri
    totalPdf->paramOn(xframe, Parameters(RooArgList(meanCB1, sigmaCB1)),Layout(0.57, 0.998, 0.99));
    //
    //
    // Faccio il draw del frame
    myCanvas->cd();
    //
    xframe->getAttText()->SetTextSize(0.03) ;
    xframe->Draw();
    //
    // Salvo la canvas
    myCanvas->SaveAs("./Plots/bin"+numeri[i]+".png");
    //
  }  // for-loop closed
//
```

I finished the set of 22 fits in rapidity bins and with the filled vectors **`vec[i]`** & **`vec_err[i]`**;
Now I can start the 2nd part where I plot and fit the mass resolution as a function of the rapidity.

```
//////////////////////////////////////////- 2nd PART - ////////////////////////////
// Come valori centrali assegno il centro del bin di rapidita'
double rapidity[22] = {-2.1, -1.9, -1.7, -1.5, -1.3, -1.1, -0.9, -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.1};
// Come errore assegno la semiampiezza dell'intervallo (0.1 in questo caso)
double rapidity_err[22];
for (int i=0; i<22; i++)
   {
    rapidity_err[i]=0.1;
   }
//
// Se il fit restituisse errori asimmetrici sui parametri
//TGraphAsymmErrors *grafico_errori = new TGraphAsymmErrors(22, rapidity, vec, rapidity_err, rapidity_err, vec_err_l, vec_err_h);
//
TCanvas *final = new TCanvas("final", "final", 700, 700);
TGraphErrors *grafico_errori = new TGraphErrors(22, rapidity, vec, rapidity_err, vec_err);
gPad->SetBottomMargin(0.15);
gPad->SetLeftMargin(0.15);
gPad->SetRightMargin(0.1);
gPad->SetTopMargin(0.1);
grafico_errori->SetMarkerStyle(20);
grafico_errori->SetMarkerColor(kBlue);
grafico_errori->SetTitle("");
grafico_errori->GetXaxis()->SetTitle("Rapidity/0.2");
grafico_errori->GetXaxis()->SetLabelSize(0.03);
grafico_errori->GetXaxis()->SetTitleOffset(1.1);
grafico_errori->GetYaxis()->SetTitleOffset(1.1);
grafico_errori->GetYaxis()->SetLabelSize(0.03);
grafico_errori->GetYaxis()->SetTitle("#sigma(MeV)");
grafico_errori->GetYaxis()->SetDecimals(1);
grafico_errori->Draw("AP");
//
// Try a Parabola
//TF1 *myFunc = new TF1("myFunc", "[0]+ [1]*x + [2]*x*x", -2.2, 2.2);
//myFunc->SetParameters(0.02, 1, 1);

// Try an hyperbolic Cosine
//
//TF1 *myFunc = new TF1("myFunc", "[0]+ 0.5*[1]*TMath::Exp([2]*x)+0.5*[1]*TMath::Exp(-[2]*x)", -2.2, 2.2);
// myFunc->SetNpx(100);
//myFunc->SetParameters(30., 1, 0.01);
//myFunc->SetParLimits(0,-10, 22);
//myFunc->SetParLimits(1, 0., 30.);
//myFunc->SetParLimits(2,-100, 100);
//
// Try a simple Cosine:
TF1 *myFunc = new TF1("myFunc", "[0]-[1]*TMath::Cos([2]*x)", -2.2, 2.2);
myFunc->SetParameters(35., 15., 1.);
myFunc->SetParLimits(0,10, 60);
myFunc->SetParLimits(1, 5., 30.);
myFunc->SetParLimits(2,-10, 10);
//
myFunc->SetLineColor(kRed);
myFunc->SetLineWidth(2);
//
gStyle->SetOptFit(111);
grafico_errori->Fit(myFunc,"R");
//
final->SaveAs("./Plots/grafico_finale.png");
//
//delete final;
}
```

- Rapidity y can be defined for any particle emerging from the collision. Let's consider a particle of mass m, energy-momentum E, p and define the rapidity

$$y = \frac{1}{2}\ln\frac{E+p_z}{E-p_z} = \frac{1}{2}\ln\frac{1+\beta\cos\theta}{1-\beta\cos\theta}$$

- Pseudorapidity η: it is the rapidity of a particle of 0 mass:

$$\eta = \frac{1}{2}\ln\frac{1+\beta\cos\theta}{1-\beta\cos\theta} \rightarrow \frac{1}{2}\ln\frac{1+\cos\theta}{1-\cos\theta} = -\ln\tan\frac{\theta}{2}$$

- Transverse energy and momentum:

$$E_T^2 = p_x^2 + p_y^2 + m^2 = E^2 - p_z^2 = \frac{E^2}{\cosh^2 y}; p_T^2 = p_x^2 + p_y^2 = p^2\sin^2\theta$$

- General consideration: Energy and momentum conservation are expected to hold "roughly" in the transverse plane. This gives rise to the concept of missing E_T
- We do not expect momentum conservation on the longitudinal direction.