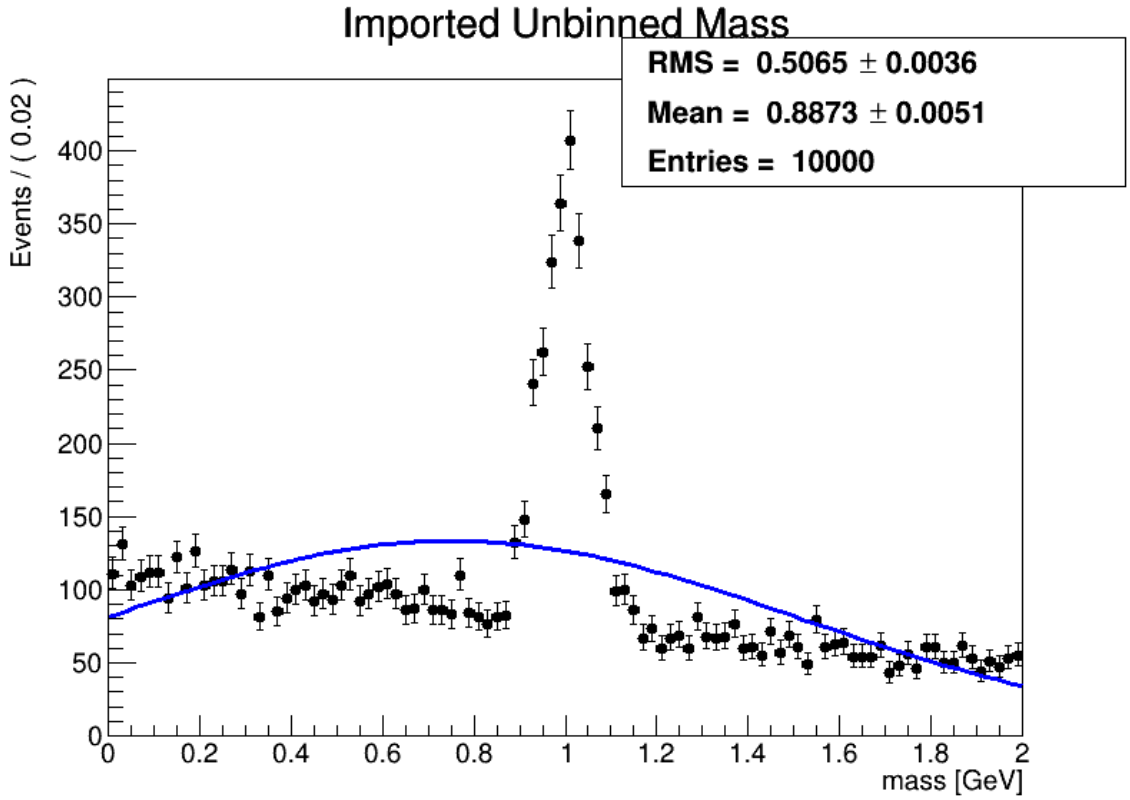
Introdução à Análise de dados em FAE **(26/04/24)**

EXERCÍCIOS ROOFIT

Professores: Sandro Fonseca, Maurício Thiel, Eliza Melo *Name:* Isis Prazeres Mota

EXERCICIO 1

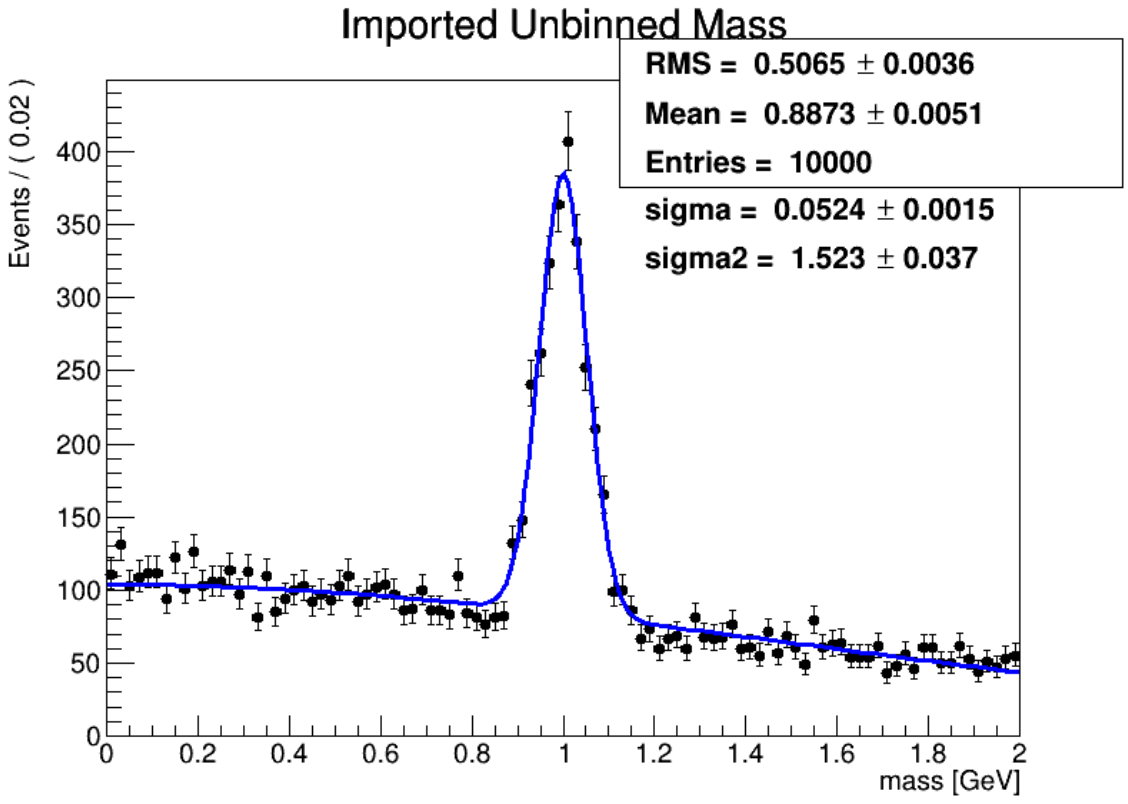
Exemplo para o exercício retorna o seguinte plot:



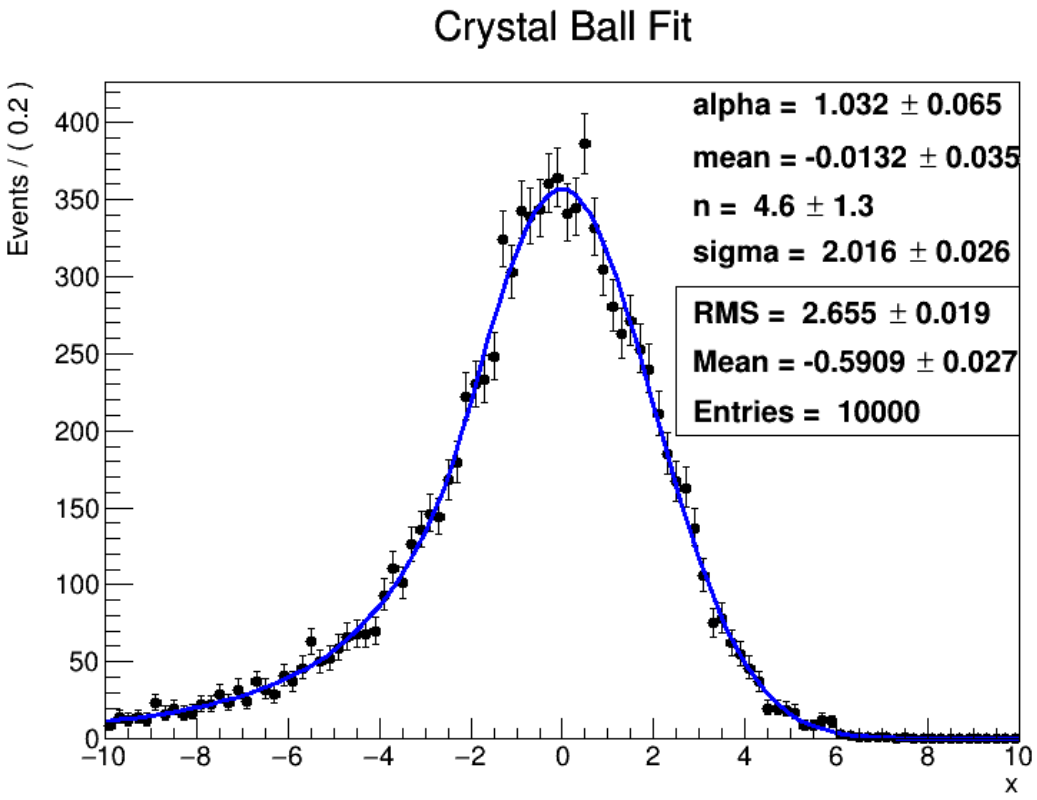
Após alguns ajustes, resolvi fazer duas gaussianas e somá-las para que pudesse ajustá-las aos dados. O código em c++ ficou assim:

```
#include <TFile.h>
#include <TNtuple.h>
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooGaussian.h>
#include <RooArgSet.h>
#include <RooPlot.h>
#include <TCanvas.h>
#include <RooAddPdf.h>
#include <RooPolynomial.h>

void fit_example() {
    // Codigo existente para abrir o arquivo e definir as variaveis...
    TFile *fin = TFile::Open("/mnt/c/Users/Isis/Downloads/example_data.root");
    if (!fin || fin->IsZombie()) {
        std::cerr << "Erro ao abrir o arquivo. Verifique o caminho e as permissoes."
           << std::endl;
```

```
17          return;
18      }
19      TNtuple *nt = (TNtuple*)fin->Get("nt");
20      RooRealVar mass("mass", "mass [GeV]", 0.0, 2.0);
21      RooDataSet data("data", "dataset with mass", nt, RooArgSet(mass));
22
23      // Definindo duas Gaussianas para o ajuste
24      RooRealVar mean("mean", "Mean of Gaussian 1", 1.0, 0.0, 2.0);
25      RooRealVar sigma("sigma", "Width of Gaussian 1", 0.18, 0.0, 10.0);
26      RooGaussian gauss1("gauss1", "First Gaussian PDF", mass, mean, sigma);
27
28      RooRealVar mean2("mean2", "Mean of Gaussian 2", 1.1, 0.0, 2.0);
29      RooRealVar sigma2("sigma2", "Width of Gaussian 2", 0.15, 0.0, 10.0);
30      RooGaussian gauss2("gauss2", "Second Gaussian PDF", mass, mean2, sigma2);
31
32      // Coeficiente para a combinacao das duas Gaussianas
33      RooRealVar coef("coef", "Coefficient for Gaussian Mixture", 0.5, 0.0, 1.0);
34
35      // Modelo final como uma soma de duas Gaussianas
36      RooAddPdf sumGauss("sumGauss", "Sum of two Gaussians", RooArgList(gauss1, gauss2)
            , RooArgList(coef));
37      sumGauss.fitTo(data);
38
39      // Plotando os resultados
40      RooPlot* frame = mass.frame(RooFit::Title("Imported Unbinned Mass"));
41      data.plotOn(frame);
42      sumGauss.plotOn(frame);
43      sumGauss.paramOn(frame, RooFit::Layout(0.55));
44      data.statOn(frame, RooFit::Layout(0.55));
45
46      TCanvas *c = new TCanvas("c", "Fit Example", 800, 600);
47      frame->Draw();
48      c->Draw();
49  }
```

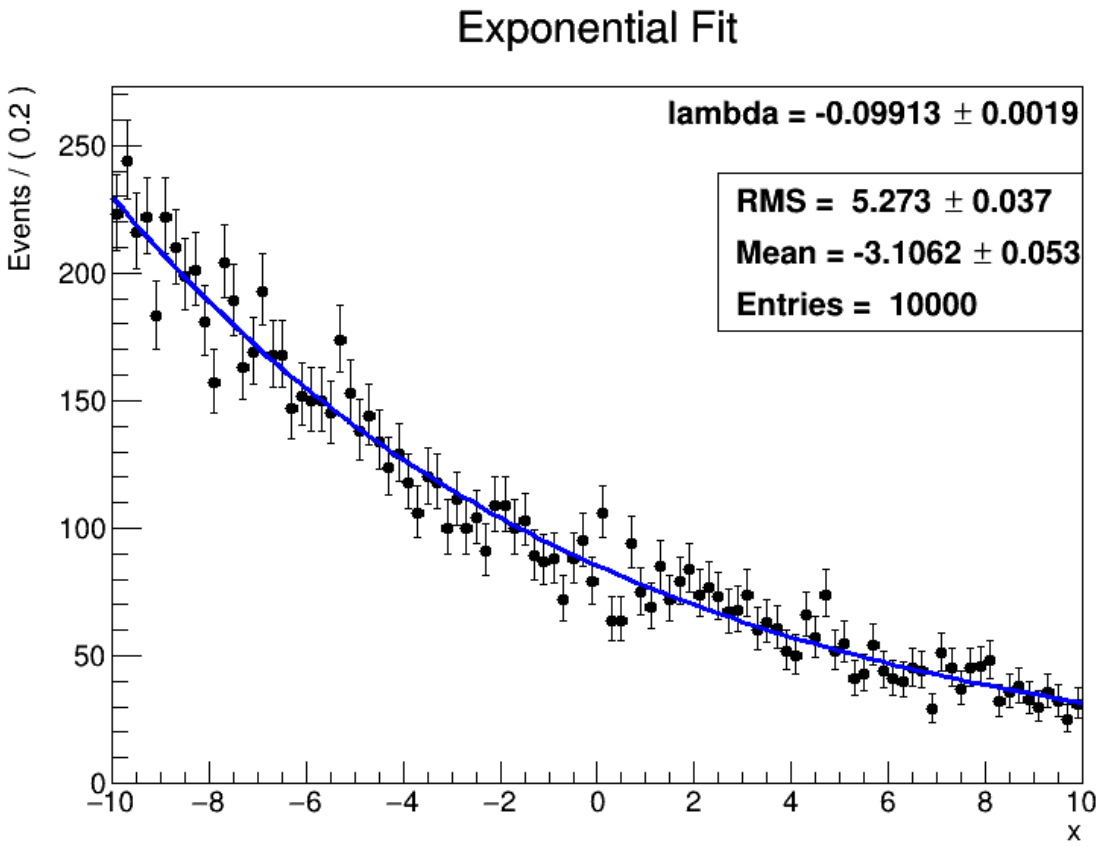


EXERCICIO 2




```
#include <RooRealVar.h>
#include <RooCBShape.h>
#include <RooDataSet.h>
#include <RooPlot.h>
#include <TCanvas.h>
#include <TFile.h>

void crystalBallFit() {
    // 1. Definir a variavel observavel
    RooRealVar x("x", "x", -10, 10);

    // 2. Parametros para a PDF Crystal Ball
    RooRealVar mean("mean", "Mean of Gaussian", 0, -10, 10);
    RooRealVar sigma("sigma", "Width of Gaussian", 2, 0.1, 10);
    RooRealVar alpha("alpha", "alpha", 1, 0, 10);
    RooRealVar n("n", "n", 5, 0, 10);

    // 3. Criar a PDF Crystal Ball
    RooCBShape cb("cb", "Crystal Ball Function", x, mean, sigma, alpha, n);

    // 4. Gerar toy data
    RooDataSet* data = cb.generate(RooArgSet(x), 10000);  // Gerar 10000 eventos

    // 5. Ajustar a PDF aos dados
    cb.fitTo(*data);

    // 6. Criar um frame para desenhar os dados e o fit
    RooPlot* frame = x.frame(RooFit::Title("Crystal Ball Fit"));
    data->plotOn(frame);
    cb.plotOn(frame);
```

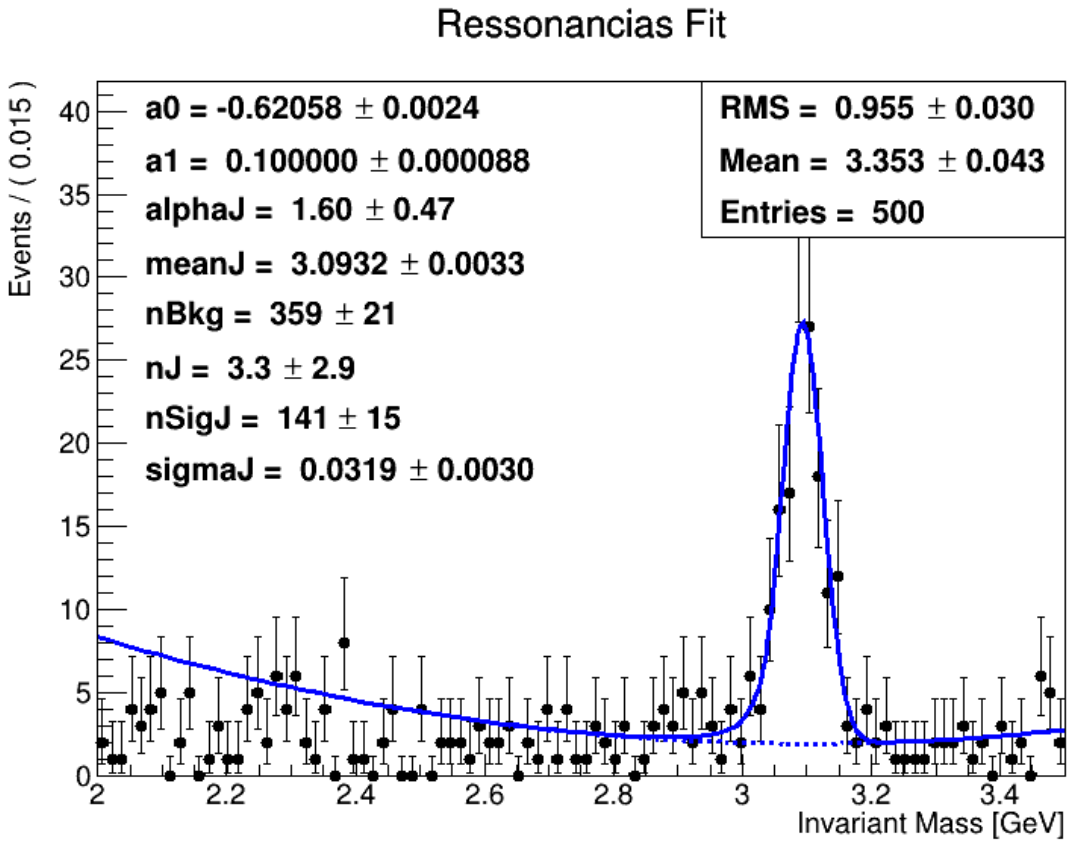
```
31
32      // 7. Adicionar informacoes estatisticas
33      cb.paramOn(frame, RooFit::Layout(0.6, 0.9, 0.9));
34      data->statOn(frame, RooFit::Layout(0.6, 0.9, 0.65));
35
36      // 8. Desenhar o frame
37      TCanvas* c = new TCanvas("c", "Crystal Ball Fit", 800, 600);
38      frame->Draw();
39
40      // 9. Salvar o canvas
41      c->SaveAs("crystal_ball_fit_cpp.png");
42  }
43
44  int main() {
45      crystalBallFit();
46      return 0;
47  }
```




```
1   #include <RooRealVar.h>
2   #include <RooExponential.h>
3   #include <RooDataSet.h>
4   #include <RooPlot.h>
5   #include <TCanvas.h>
6   #include <TFile.h>
7
8   void exponentialFit() {
9       // 1. Definir a variavel observavel
10      RooRealVar x("x", "x", -10, 10);
11
12      // 2. Parametro para a PDF Exponencial
13      RooRealVar lambda("lambda", "Decay rate", -0.1, -10.0, 10.0);
14
```

```
    // 3. Criar a PDF Exponencial
    RooExponential expo("expo", "Exponential PDF", x, lambda);

    // 4. Gerar toy data
    RooDataSet* data = expo.generate(RooArgSet(x), 10000);  // Gerar 10000 eventos

    // 5. Ajustar a PDF aos dados
    expo.fitTo(*data);

    // 6. Criar um frame para desenhar os dados e o fit
    RooPlot* frame = x.frame(RooFit::Title("Exponential Fit"));
    data->plotOn(frame);
    expo.plotOn(frame);

    // 7. Adicionar informacoes estatisticas e parametros
    expo.paramOn(frame, RooFit::Layout(0.54, 0.9, 0.9));  // Ajuste na posicao x1, x2, y2
    data->statOn(frame, RooFit::Layout(0.6, 0.9, 0.8)); // Ajuste para evitar sobreposicao

    // 8. Desenhar o frame
    TCanvas* c = new TCanvas("c", "Exponential Fit", 800, 600);
    frame->Draw();

    // 9. Salvar o canvas
    c->SaveAs("exponential_fit_cpp.png");
}

int main() {
    exponentialFit();
    return 0;
}
```

EXERCICIO 3




```cpp
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooGaussian.h>
#include <RooCBShape.h>
#include <RooPolynomial.h>
#include <RooAddPdf.h>
#include <RooFitResult.h>
#include <RooPlot.h>
#include <TFile.h>
#include <TCanvas.h>
#include <iostream>

void fitResonances() {
    // Abrir arquivo
    TFile *file = TFile::Open("/mnt/c/Users/Isis/Downloads/DataSet_lowstat.root");

    // Carregar RooDataSet do arquivo
    RooDataSet *data = (RooDataSet*)file->Get("data");
    data->Print();

    // Definir variavel do RooFit
    RooRealVar mass("mass", "Invariant Mass [GeV]", 2.0, 3.5);

    // Parametros da Crystal Ball para J/psi
    RooRealVar meanJ("meanJ", "Mean of J/psi", 3.1, 2.9, 3.3);
    RooRealVar sigmaJ("sigmaJ", "Sigma of J/psi", 0.03, 0.01, 0.05);
    RooRealVar alphaJ("alphaJ", "Alpha of J/psi", 1, 0, 10);
    RooRealVar nJ("nJ", "n of J/psi", 5, 0, 10);
```

```
    RooCBShape cbJ("cbJ", "Crystal Ball for J/psi", mass, meanJ, sigmaJ, alphaJ, nJ);

    // Fundo polinomial
    RooRealVar a0("a0", "a0", 0.1, -1, 1);
    RooRealVar a1("a1", "a1", 0.01, -0.1, 0.1);
    RooPolynomial background("background", "Background", mass, RooArgList(a0, a1));

    // Modelo composto
    RooRealVar nSigJ("nSigJ", "Number of J/psi Events", 500, 0, 10000);
    RooRealVar nBkg("nBkg", "Number of Background Events", 5000, 0, 100000);
    RooAddPdf model("model", "Total Model", RooArgList(cbJ, background), RooArgList(
        nSigJ, nBkg));

    // Ajustar modelo
    RooFitResult *result = model.fitTo(*data, RooFit::Save());

    // Desenhar e salvar resultados
    RooPlot* frame = mass.frame(RooFit::Title("Ressonancias Fit"));
    data->plotOn(frame);
    model.plotOn(frame);
    model.plotOn(frame, RooFit::Components(background), RooFit::LineStyle(kDashed));
    model.paramOn(frame, RooFit::Layout(0.10, 0.9, 0.9));
    data->statOn(frame, RooFit::Layout(0.6, 0.9, 0.9));


    TCanvas* c = new TCanvas("c", "Fit to J/psi and background", 800, 600);
    frame->Draw();
    c->SaveAs("fit_result.png");
}
int main() {
    fitResonances();
    return 0;
}
```