Introdução à Análise de dados em FAE **(DATA)**

Lista 3

Professores: Sandro, Mauricio e Eliza *Name:* Dalmo da Silva Dalto

https://github.com/Dalmomr/web-project/tree/new_branch

EXERCICIO 0:

```
{

TFile *input = new TFile("DYJetsToLL.root","read");
TTree *t = (TTree *)input->Get("Events");

c1->cd(1);

t->MakeClass("meu");

fiz essa modifica  o no .h{}
virtual void     Loop(TTree *tree1);
}


#define meu_cxx
#include "meu.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

void meu::Loop(TTree* tree1)
{
    // Verifica se a  rvore     um ponteiro  v lido
    if (!tree1) {
        throw std::runtime_error("Erro: Ponteiro de  rvore  inv lido");
    }

    // Define as vari veis que ser o preenchidas na TTree
    Float_t muon_mass;
    Float_t tau_mass;

    // Associa as vari veis aos ramos da TTree
    tree1->Branch("muon_mass", &muon_mass, "muon_mass/F");
    tree1->Branch("tau_mass", &tau_mass, "tau_mass/F");

    if (!fChain) {
        throw std::runtime_error("Erro: fChain n o est  inicializado");
    }

    Long64_t nentries = fChain->GetEntriesFast();
    Long64_t nbytes = 0, nb = 0;

    for (Long64_t jentry = 0; jentry < nentries; jentry++) {
        Long64_t ientry = LoadTree(jentry);
        if (ientry < 0) break;

        nb = fChain->GetEntry(jentry);
        nbytes += nb;

```

```
50          // Verifica se Muon_mass e Tau_mass t m pelo menos um elemento
51          if (Muon_mass[0] && Tau_mass[0]) {
52              // Preenche as vari veis com os valores dos ramos
53              muon_mass = Muon_mass[0];
54              tau_mass = Tau_mass[0];
55
56              // Preenche os ramos da TTree com os valores das vari veis
57              tree1->Fill();
58          } else {
59              std::cerr << "Acesso inv lido aos dados de Muon_mass ou Tau_mass na
                    entrada " << jentry << std::endl;
60          }
61      }
62  }
63
64
65  .L meu.C;
66
67  meu l;
68  TTree *tree2 = new TTree("myTree", "Tree Title");
69  meu l;
70  l.Loop(tree2);
71
72  TCanvas *c1= new TCanvas("c1","c1",1000,500);
73  c1->Divide(1,2);
74  c1->cd(1);
75  tree1->Draw("muon_mass");
76  c1->cd(2);
77  tree1->Draw("tau_mass");
78  }
```

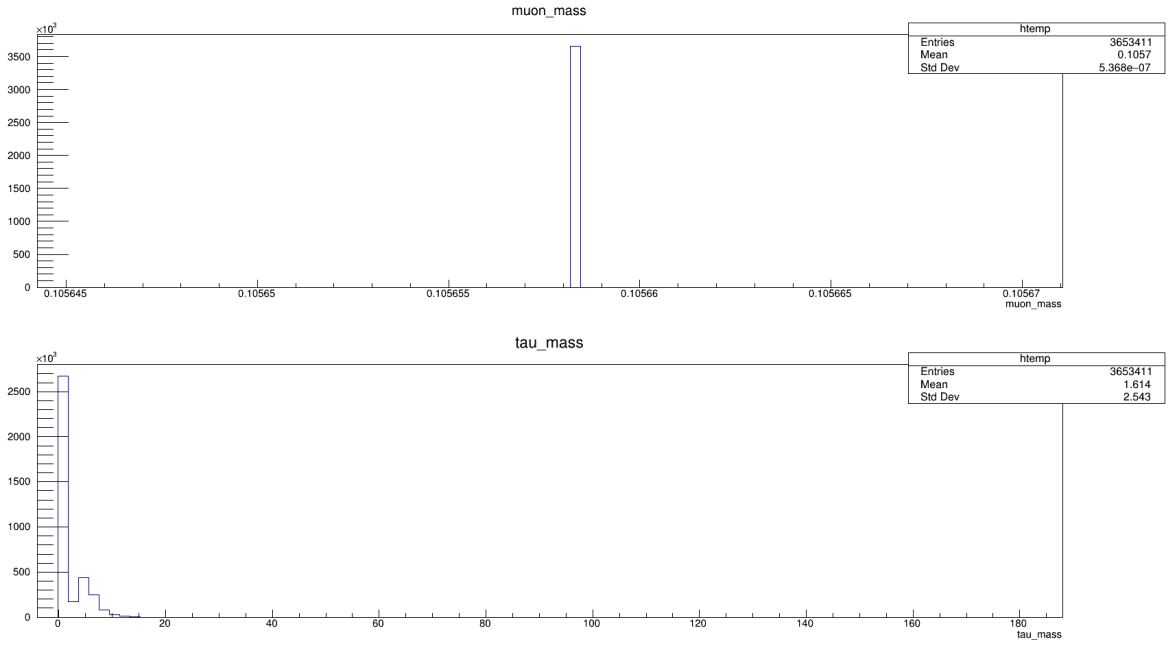

Figura 1: EXERCICIO 0

EXERCICIO 1:

```
1  {
```

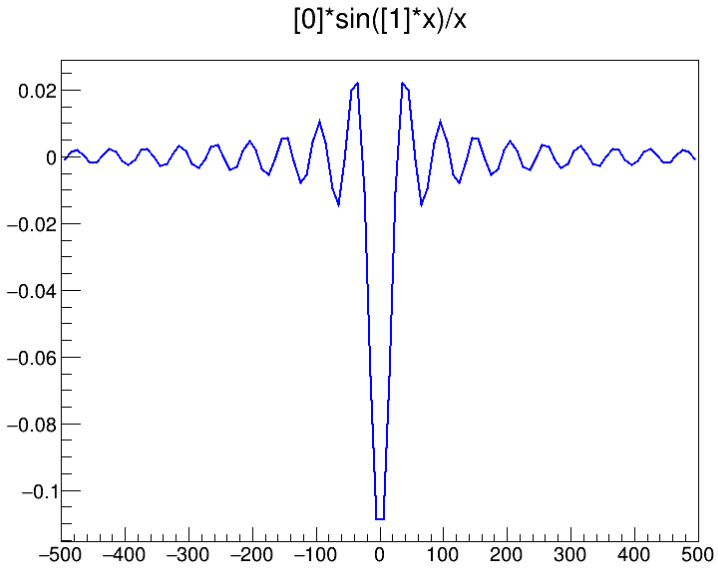

Figura 2: EXERCICIO 1

```
2
3  TF1 * f1 = new TF1("f1","[0]*sin([1]*x)/x", -500, +500);
4
5  f1->SetParameter(0, 1); // p0 = 1
6  f1->SetParameter(1, 2); // p1 = 2
7  f1->SetLineColor(kBlue);
8
9
10 TCanvas *c1 = new TCanvas("c1", "Function Plot", 800, 600);
11
12 f1->Draw();
13
14 double functionValue = f1->Eval(1);
15 printf("a. Function value for x = 1: %.4f\n", functionValue);
16
17 double functionDerivative = f1->Derivative(1);
18 printf("b. Function derivative for x = 1: %.4f\n", functionDerivative);
19
20 double integral = f1->Integral(0, 3);
21 printf("c. Integral of the function between 0 and 3: %.4f\n", integral);
22 }
```

A saída dessa macro resulta em um plot mostrado acima e os seguintes resultados:
a. Function value for x = 1: 0.9093
b. Function derivative for x = 1: -nan
c. Integral of the function between 0 and 3: 1.4247
O resultado da derivada é nan porque não tem derivada naquele ponto.

EXERCICIO 2:

```
1      {
2
3  ifstream arq1;
4  ifstream arq2;
5
6  arq1.open("graphdata.txt");
7  arq2.open("graphdata_error.txt");
8
```

```
9
10 float x[10],y[10];
11 float x_1[10], ex[10],y_1[10], ey[10];
12
13 int i=0;
14
15 while(!arq1.eof() and !arq2.eof()){
16
17 arq1>>x[i]>>y[i];
18 arq2>>x_1[i]>>y_1[i]>>ex[i]>>ey[i];
19
20 cout<<x[i]<<" "<<y[i]<<endl;
21
22 cout<<x_1[i]<<" "<<y_1[i]<<" "<<ex[i]<<" "<<ey[i]<<endl;
23
24
25 i++;
26
27
28 }
29
30 TCanvas *c1= new TCanvas("c1","c1",1000,500);
31
32 c1->Divide(2,1);
33 c1->cd(1);
34 TGraph *t= new TGraph(10,x,y);
35 t->Draw();
36 t->SetTitle("Plot sem barra de erros");
37
38 c1->cd(2);
39 TGraphErrors *t1= new TGraphErrors(10,x_1,y_1,ex,ey);
40 t1->Draw();
41 t1->SetTitle("Plot com barra de erros");
42
43 }
```

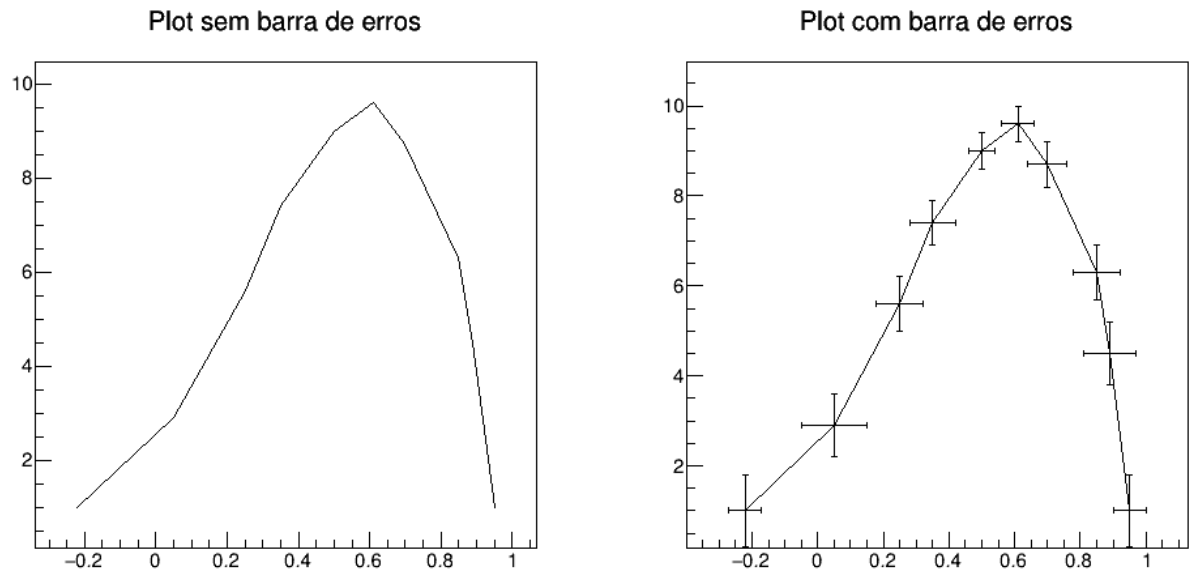

Figura 3: EXERCICIO 2

EXERCICIO 3:

```
    {

TRandom *t= new TRandom;
TH1F *h1= new TH1F("Fit gauss","Fit gauss",50,0,10);

for(int i=0; i<10000;i++){

        float random_gauss= t->Gaus(5,2);

        h1->Fill(random_gauss);
}

h1->Draw();
h1->Fit("gaus");
gStyle->SetOptFit(1111);
}
```

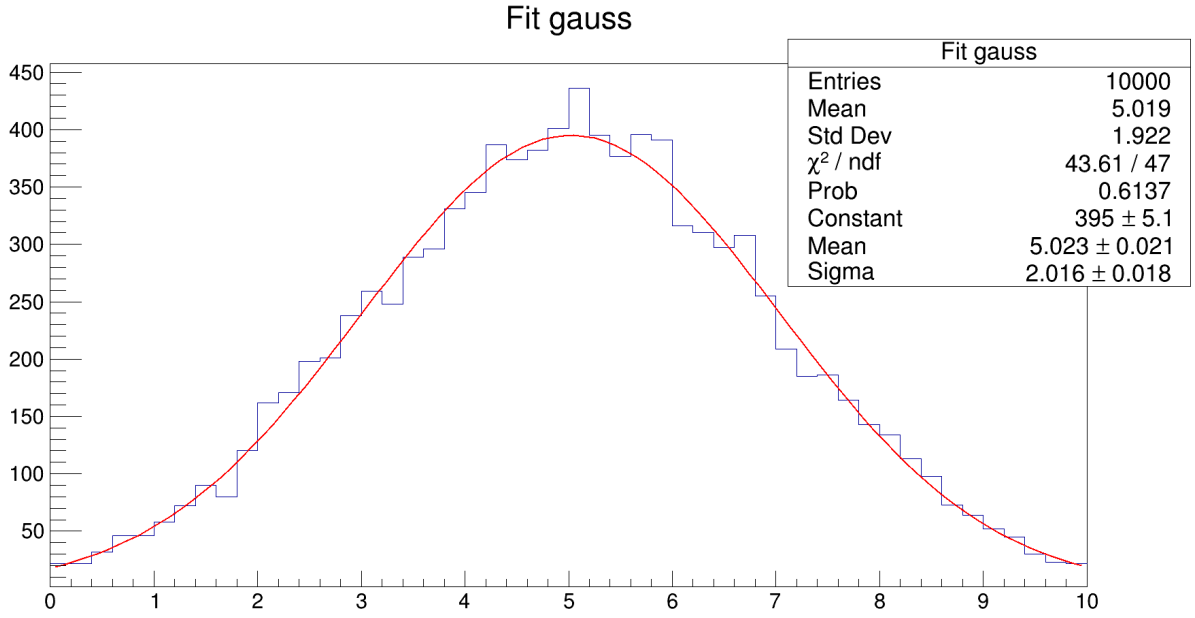

Figura 4: EXERCICIO 3

EXERCICIO 4:

```
    {
TCanvas *c1= new TCanvas("c1","c1",1500,800);

TFile *input = new TFile("tree.root","read");
TTree *t = (TTree *)input->Get("tree1");

gStyle->SetOptFit(1111);

TF1 *gauss = new TF1("g1","gaus",0,10);


c1->Divide(3,1);
c1->cd(1);
```

```
14 t->Draw("ebeam>>h1");
15 h1->Fit(gauss);
16
17 float mean=gauss->GetParameter(1);
18 float sigma=gauss->GetParameter(2);
19
20 c1->cd(2);
21
22 char cut[20];
23
24 sprintf(cut,"ebeam>%f",0.2 + mean);
25
26 t->Draw("px+py+pz>>h2",cut,"");
27
28
29 h2->Fit(gauss);
30 h2->SetTitle("px+py+pz (ebeam>0.2 + Mean_{ebeam})");
31
32 c1->cd(3);
33
34
35 t->Draw("px+py+pz>>h3","","");
36
37 h3->Fit(gauss);
38 h3->SetTitle("px+py+pz");
39
40 c1->Print("exercicio4.png","png");
41 }
```

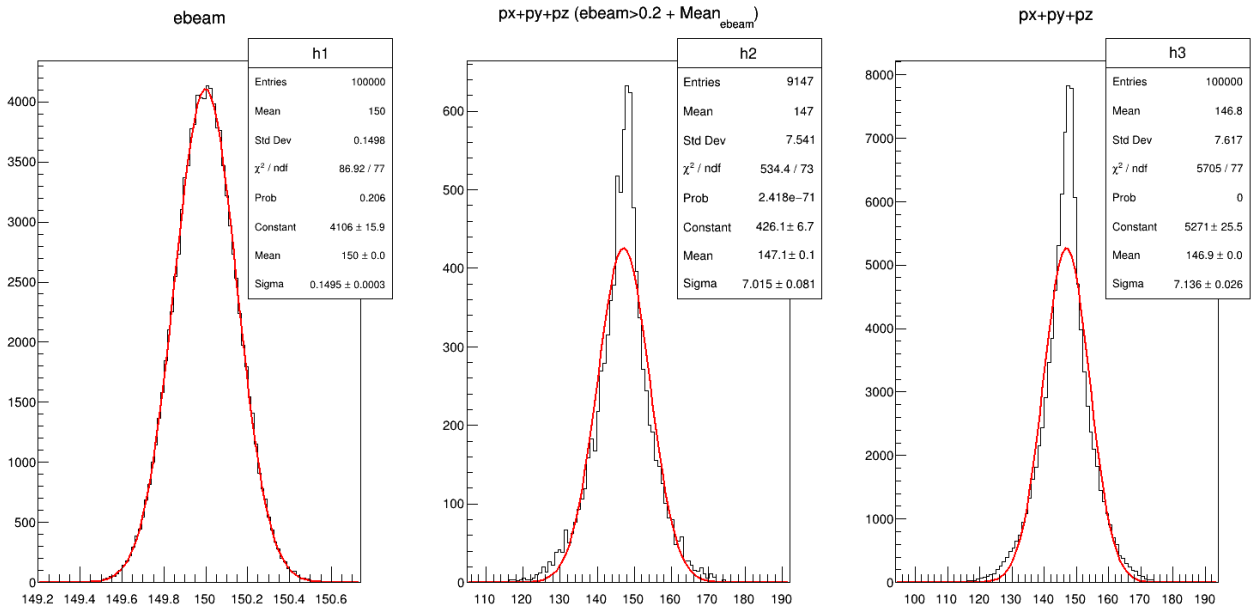

Figura 5: EXERCICIO 4