Introdução à Análise de dados em FAE **(19/04/2024)**

Exercícios ROOT

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

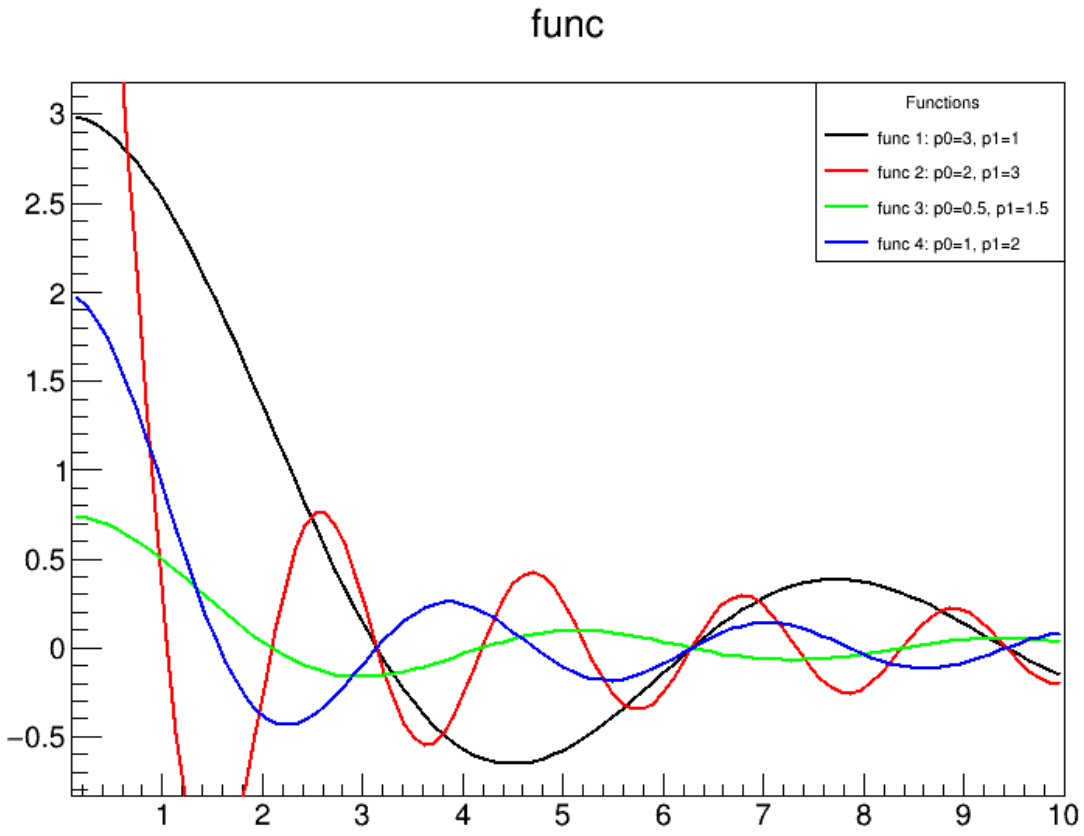
TEXTO

EXERCICIO 1

Create a function with parameters, p0 * sin (p1 * x) / x, and also draw it for different parameter values. Set the colour of the parametric function to blue. After having drawn the function, compute for the parameter values (p0 = 1, p1 = 2):
a.Function value for x=1
b.Function derivative for x=1
c.Integral of the function between 0 and 3

```
#include <TCanvas.h>
#include <TMath.h>
#include <TF1.h>
#include <TLegend.h>

void parametric_function() {
    // Desenha a funcao
    TCanvas *canvas = new TCanvas("canvas", "Parametric Function", 800, 600);

    // Array com diferentes conjuntos de parametros
    double params[][2] = {{3, 1}, {2, 3}, {0.5, 1.5}, {1, 2}};

    // Numero de parametros para testar
    int numberOfParams = sizeof(params) / sizeof(params[0]);

    // Cria uma legenda
    TLegend *legend = new TLegend(0.7, 0.7, 0.9, 0.9); // Posicao: canto superior
        direito
    legend->SetHeader("Functions", "C"); // Cabecalho da legenda

    // Cria e desenhar multiplas funcoes
    for (int i = 0; i < numberOfParams; i++) {
        double p0 = params[i][0];
        double p1 = params[i][1];

        // Define a funcao com os parametros p0 e p1
        TF1 *func = new TF1("func", [p0, p1](double *x, double *p) { return p0 * sin(
            p1 * x[0]) / x[0]; }, 0.1, 10, 0);
        func->SetParameters(p0, p1);  // Configura os parametros

        // Configura a cor e o estilo da linha para cada funcao
        func->SetLineColor(i+1);  // Cores diferentes para cada funcao
        func->SetLineWidth(2);

        // Desenha a funcao
        if (i == 0) func->Draw();
        else func->Draw("same");

        // Adicionar cada funcao a legenda com um rotulo correspondente
        legend->AddEntry(func, Form("func %d: p0=%g, p1=%g", i + 1, p0, p1), "l");
    }

    // Desenha a legenda
```

```
42      legend->Draw();
43
44      // Atualiza o canvas
45      canvas->Update();
46
47      // Salva a imagem
48      canvas->SaveAs("parametric_functions.png");
49  }
50
51  int main() {
52      parametric_function();
53      return 0;
54  }
```



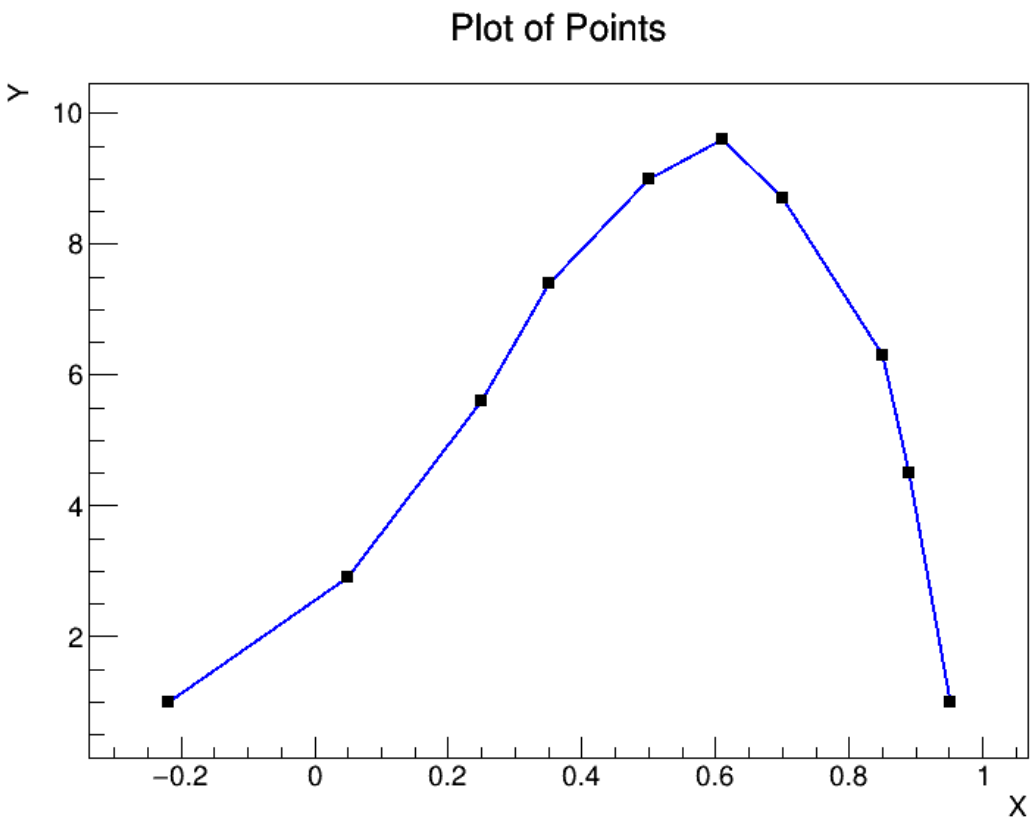
EXERCICIO 2

Using these points:

```
-0.22 1
0.05 2.9
0.25 5.6
0.35 7.4
0.5 9
0.61 9.6
0.7 8.7
0.85 6.3
0.89 4.5
0.95 1
```

Plot these points using the TGraph class. Use as marker point a black box. Looking at the possible options for drawing the TGraph in TGraphPainter, plot a line connecting the points.

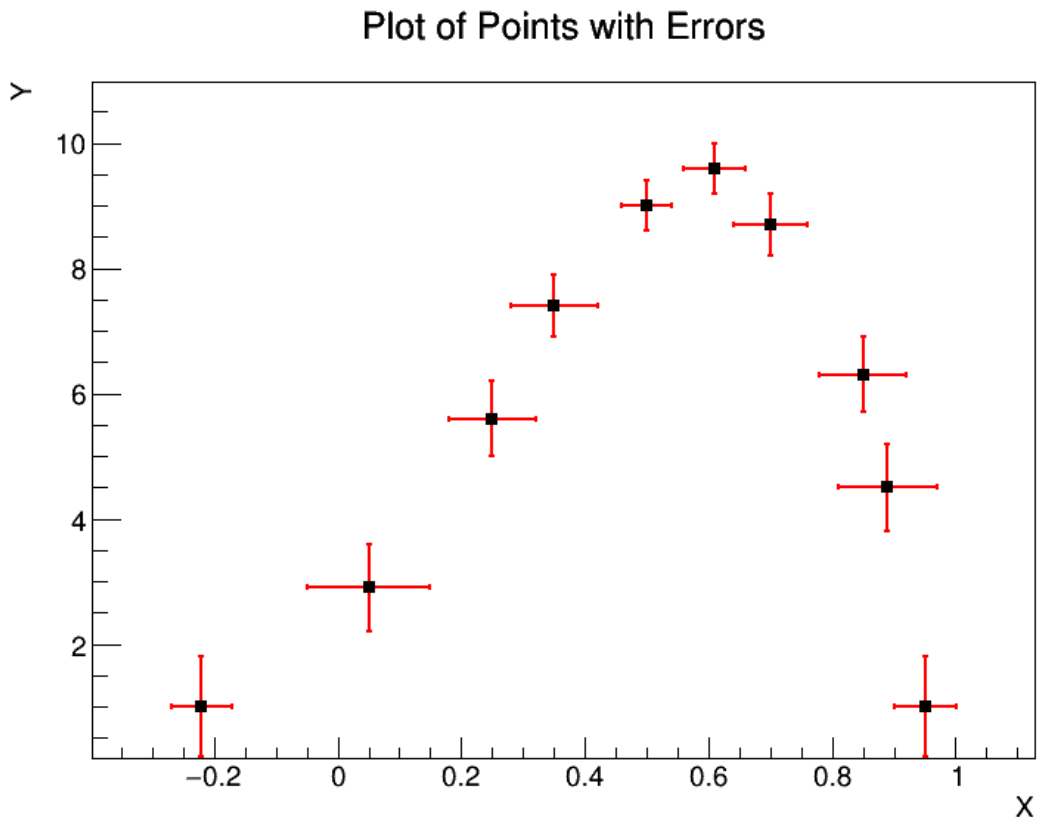
```
#include <TCanvas.h>
#include <TGraph.h>
#include <TAxis.h>

void plot_points() {
    // Dados dos pontos
    const int n = 10;
    double x[n] = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61, 0.7, 0.85, 0.89, 0.95};
    double y[n] = {1, 2.9, 5.6, 7.4, 9, 9.6, 8.7, 6.3, 4.5, 1};

    // Criando o TGraph
    TGraph *gr = new TGraph(n, x, y);
    gr->SetTitle("Plot of Points;X;Y");
    gr->SetMarkerStyle(21);  // Estilo de marcador de caixa (box)
    gr->SetMarkerColor(kBlack);
    gr->SetLineColor(kBlue);
    gr->SetLineWidth(2);

    // Criando o canvas
    TCanvas *c1 = new TCanvas("c1", "Canvas for Plotting Points", 800, 600);
    gr->Draw("APL");  // A: Axis, P: Points, L: Line

    // Salvando o canvas
    c1->SaveAs("points_graph.png");
}

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



Make a TGraphError and display it by using the points with error, containing error in x and y.

```
-0.22 1 0.05 0.8
0.05 2.9 0.1 0.7
0.25 5.6 0.07 0.6
0.35 7.4 0.07 0.5
0.5 9 0.04 0.4
0.61 9.6 0.05 0.4
0.7 8.7 0.06 0.5
0.85 6.3 0.07 0.6
0.89 4.5 0.08 0.7
0.95 1 0.05 0.8
```

```cpp
#include <TCanvas.h>
#include <TGraphErrors.h>
#include <TAxis.h>

void plot_points_with_errors() {
    // Dados dos pontos com erros
    const int n = 10;
    double x[n] = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61, 0.7, 0.85, 0.89, 0.95};
    double y[n] = {1, 2.9, 5.6, 7.4, 9, 9.6, 8.7, 6.3, 4.5, 1};
    double ex[n] = {0.05, 0.1, 0.07, 0.07, 0.04, 0.05, 0.06, 0.07, 0.08, 0.05};
    double ey[n] = {0.8, 0.7, 0.6, 0.5, 0.4, 0.4, 0.5, 0.6, 0.7, 0.8};

    // Criando o TGraphErrors
    TGraphErrors *grErrors = new TGraphErrors(n, x, y, ex, ey);
    grErrors->SetTitle("Plot of Points with Errors;X;Y");
    grErrors->SetMarkerStyle(21);  // Estilo de marcador de caixa (box)
    grErrors->SetMarkerColor(kBlack);
    grErrors->SetLineColor(kRed);
    grErrors->SetLineWidth(2);

    // Criando o canvas
    TCanvas *c1 = new TCanvas("c1", "Canvas for Plotting Points with Errors", 800,
        600);
    grErrors->Draw("AP");  // A: Axis, P: Points

    // Salvando o canvas
    c1->SaveAs("points_with_errors_graph.png");
}

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

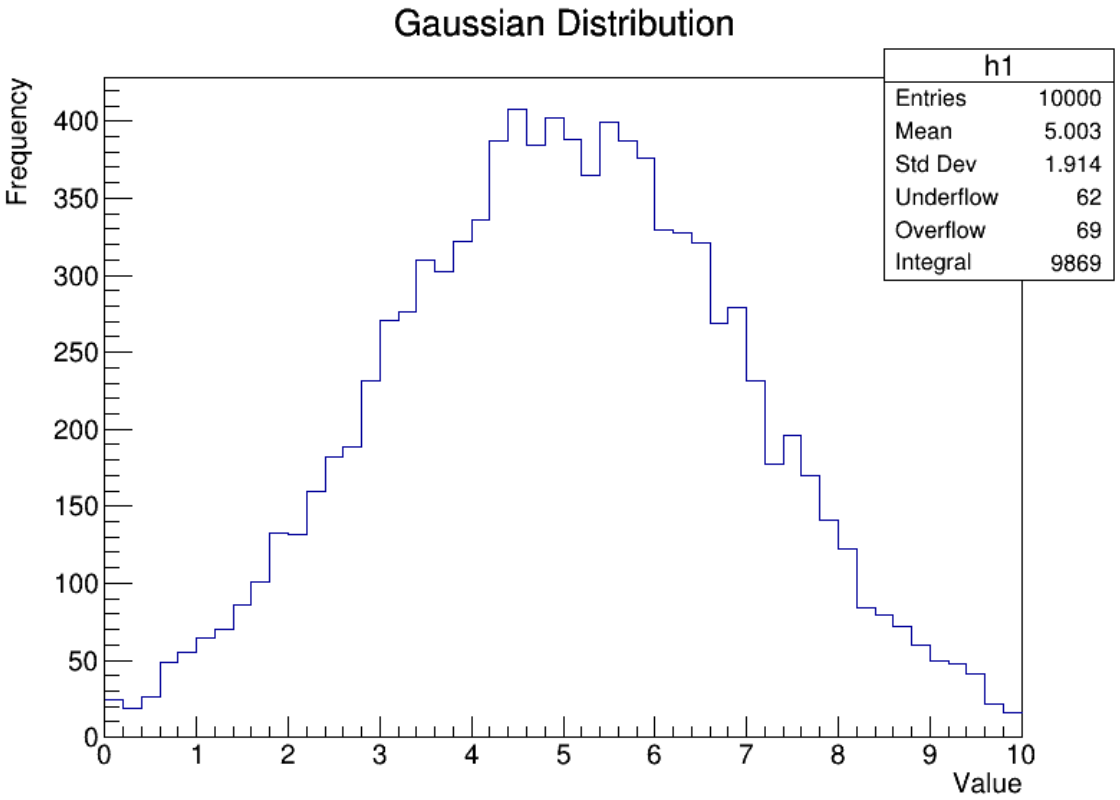


EXERCICIO 3

Create a one-dimensional histogram with 50 bins between 0 to 10, and fill it with 10000 gaussian distributed random numbers with mean 5 and sigma 2. Plot the histogram and, looking at the documentation in the THistPainter, show in the statistic box the number of entries, the mean, the RMS, the integral of the histogram, the number of underflows, the number of overflows, the skewness and the kurtosis.

```
#include <TH1F.h>
#include <TRandom3.h>
#include <TCanvas.h>

void create_histogram() {
    // Criando o histograma
    TH1F *h1 = new TH1F("h1", "Gaussian Distribution;Value;Frequency", 50, 0, 10);

    // Gerador de numeros aleatorios
    TRandom3 rand(0);  // seed = 0 usa o relogio para gerar a semente

    // Preenchendo o histograma com numeros gaussianos
    for (int i = 0; i < 10000; i++) {
        h1->Fill(rand.Gaus(5, 2));  // media = 5, sigma = 2
    }

    // Criando um canvas para desenhar o histograma
    TCanvas *c1 = new TCanvas("c1", "Histogram Example", 800, 600);
    h1->Draw();

    // Configurando a caixa de estatisticas
    // kTRUE = 1, kFALSE = 0
    h1->SetStats(kTRUE);
    gStyle->SetOptStat(1111111); // Mostrar todas as estatisticas solicitadas
    gStyle->SetOptFit(1111); // Mostrar informacoes de ajuste

```

```
    // Adicionando skewness e kurtosis a caixa de estatisticas
    gStyle->SetOptStat(1111111);
    gStyle->SetOptFit(112); // Mostrar skewness (s) e kurtosis (k)

    // Redesenhando para atualizar com as novas configuracoes de estatisticas
    h1->Draw();

    // Salvando o resultado
    c1->SaveAs("histogram.png");
}

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

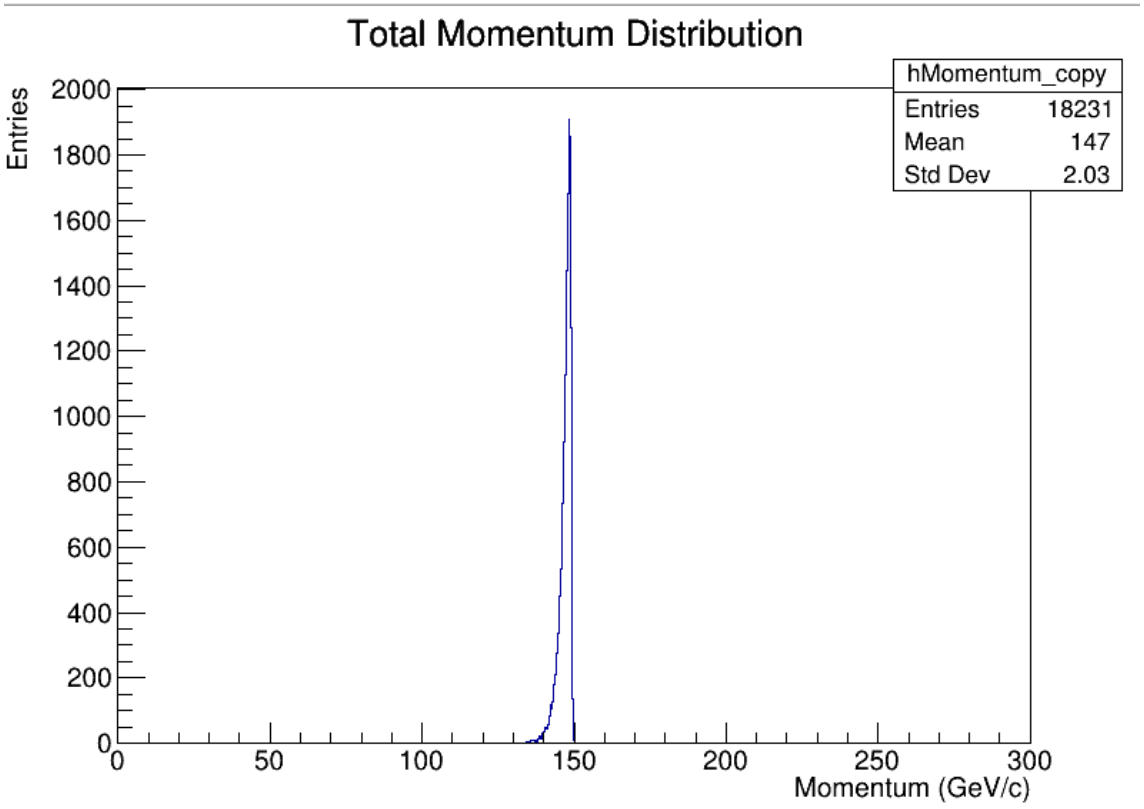


EXERCICIO 4

Using the tree contained in tree.root make a distribution of the total momentum of each whose beam energy was outside of the mean by more than 0.2. Use TCut objects to make your events selections. Project this distribution into a histogram, draw it and save it to a file.

```
#include <TFile.h>
#include <TTree.h>
#include <TH1F.h>
#include <TCanvas.h>
#include <TMath.h>
#include <iostream>

void analyzeTree() {
    TFile *file = TFile::Open("/mnt/c/Users/Isis/Downloads/tree.root");
    if (!file || file->IsZombie()) {
        std::cerr << "Error opening file!\n";
```

```
12          return;
13      }
14
15      TTree *tree = nullptr;
16      file->GetObject("tree1", tree);
17      if (!tree) {
18          std::cerr << "Tree not found!\n";
19          file->Close();
20          delete file;
21          return;
22      }
23
24      Float_t ebeam, px, py, pz;
25      tree->SetBranchAddress("ebeam", &ebeam);
26      tree->SetBranchAddress("px", &px);
27      tree->SetBranchAddress("py", &py);
28      tree->SetBranchAddress("pz", &pz);
29
30      TH1F *hMomentum = new TH1F("hMomentum", "Total Momentum Distribution;Momentum (
            GeV/c);Entries", 1000, 0, 300);
31
32      Long64_t nentries = tree->GetEntries();
33      Double_t sum_energy = 0;
34      for (Long64_t i = 0; i < nentries; ++i) {
35          tree->GetEntry(i);
36          sum_energy += ebeam;
37      }
38      Double_t mean_energy = sum_energy / nentries;
39      std::cout << "Mean energy: " << mean_energy << std::endl;
40
41      for (Long64_t i = 0; i < nentries; ++i) {
42          tree->GetEntry(i);
43          if (TMath::Abs(ebeam - mean_energy) > 0.2) {
44              Double_t p_total = sqrt(px*px + py*py + pz*pz);
45              hMomentum->Fill(p_total);
46              std::cout << "Filling p_total: " << p_total << std::endl;
47          }
48      }
49
50      std::cout << "Histogram entries: " << hMomentum->GetEntries() << std::endl;
51
52      TCanvas *c1 = new TCanvas("c1", "Canvas", 800, 600);
53      hMomentum->DrawCopy();
54      c1->Modified();
55      c1->Update();  // Atualiza o canvas
56
57      c1->Connect("TCanvas", "Closed()", "TApplication", gApplication, "Terminate()");
58
59      file->Close();
60      delete file; // Cleanup
61  }
62
63  int main(int argc, char **argv) {
64      TApplication theApp("App", &argc, argv);
65      analyzeTree();
66      theApp.Run();
67      return 0;
68  }
```


Total Momentum Distribution
hMomentum_copy
Entries 18231
Mean 147
Std Dev 2.03
Entries
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
Momentum (GeV/c)