Introdução à Análise de dados em FAE **(05/09/2024)**

Exercicios de estatística para análise de dados em HEP

Professores: Eliza Melo, Dilson Damião e Mauricio Thiel *Name:* Jorge Júlio Barreiros Venuto de Siqueira

EXERCÍCIO 1

Para realizar o que é pedido no exercício 1, foi realizado o seguinte código em C:

Código em C

```
#include <iostream>
#include <cmath>
#include "TCanvas.h"
#include "TF1.h"
#include "TGraph.h"
#include "TAxis.h"

double f(double *x, double *par) {
    double p0 = par[0];
    double p1 = par[1];
    return (x[0] == 0) ? p0 * p1 : p0 * sin(p1 * x[0]) / x[0];
}

void exercicio_1() {

    double p0 = 1.0;
    double p1 = 2.0;

    TF1 *func = new TF1("", f, 0, 5, 2);
    func->SetParameters(p0, p1);

    TCanvas *c1 = new TCanvas("c1", "Parametric Function", 800, 600);
    func->SetLineColor(kBlue);
    func->Draw();

    double x_val = 1.0;
    double func_value_at_1 = func->Eval(x_val);
    std::cout << "Function value at x = 1: " << func_value_at_1 << std::endl;

    double dx = 1e-6;
    double func_derivative_at_1 = (func->Eval(x_val + dx) - func->Eval(x_val - dx)) / (2 * dx);
    std::cout << "Function derivative at x = 1: " << func_derivative_at_1 << std::endl;

    double integral_value = func->Integral(0, 3);
    std::cout << "Integral from 0 to 3: " << integral_value << std::endl;

    c1->SaveAs("parametric_function.pdf");
}
```

Após isso, foi compilado utilizando o Root, com o comando

```
.x exercicio_1.C
```

resultando o que foi pedido

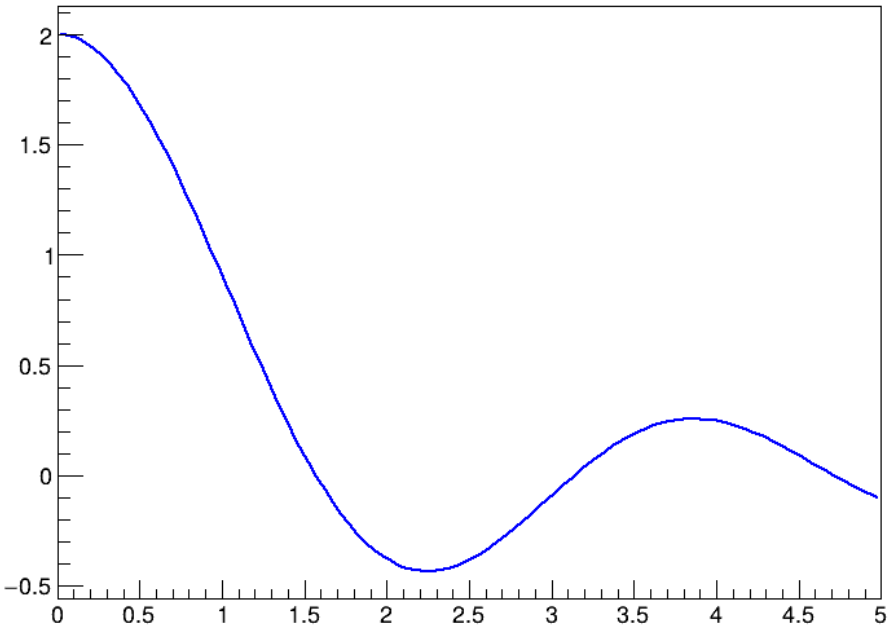


Figura 1: Resultado do exercício 1

Valor da função em $x = 1$:

$$f(1) = 0,909297$$

Valor da derivada da função em $x = 1$:

$$f'(1) = -1,74159$$

Valor da integral da função entre 0 e 3:

$$\int_0^3 f(x)\,dx = 1,42469$$

EXERCÍCIO 2

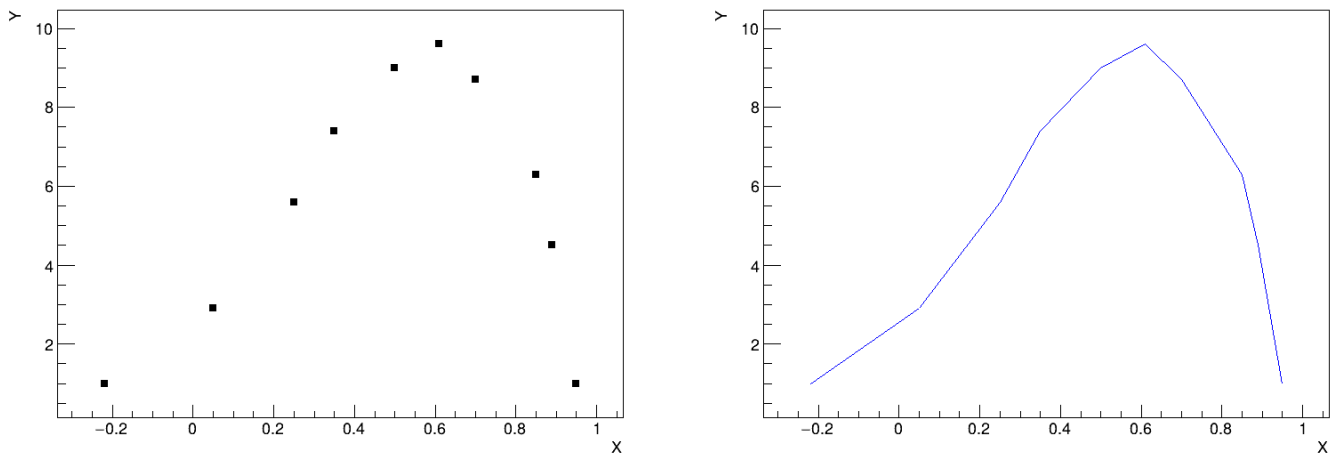
Código em C

```
#include <TGraph.h>
#include <TGraphErrors.h>
#include <TCanvas.h>
#include <iostream>
#include <fstream>

void exercicio_2() {
    // Gráfico 1: dispersão dos dados de graphdata.txt
    TCanvas *c1 = new TCanvas("c1", "", 800, 600);
    std::ifstream infile("graphdata.txt");
    int n = 0;
    double x[100], y[100];
    while (infile >> x[n] >> y[n]) {
        n++;
    }
    infile.close();
    TGraph *graphDisp = new TGraph(n, x, y);
    graphDisp->SetMarkerStyle(21);
    graphDisp->SetMarkerColor(kBlack);
    graphDisp->SetTitle(";X;Y");
    graphDisp->Draw("AP");

    c1->SaveAs("graph_dispersao.png");

    // Gráfico 2: pontos conectados por uma linha
    TCanvas *c2 = new TCanvas("c2", "", 800, 600);

    TGraph *graphLine = new TGraph(n, x, y);
    graphLine->SetLineColor(kBlue);
    graphLine->SetTitle(";X;Y");
    graphLine->Draw("AL");
    c2->SaveAs("graph_conexao.png");
    // Gráfico 3: dados com erro de graphdata_error.txt
    TCanvas *c3 = new TCanvas("c3", "", 800, 600);
    std::ifstream infile_error("graphdata_error.txt");
    double x_err[100], y_err[100], ex[100], ey[100];
    int n_error = 0;

    while (infile_error >> x_err[n_error]>> y_err[n_error] >> ex[n_error] >> ey[n_error])
    infile_error.close();
    TGraphErrors *graphErrors = new TGraphErrors(n_error, x_err, y_err, ex, ey);
    graphErrors->SetMarkerStyle(21);
    graphErrors->SetMarkerColor(kBlack);
    graphErrors->SetTitle(";X;Y");
    graphErrors->Draw("AP");

    c3->SaveAs("graph_dados_com_erro.png");
}
```

Após isso, foi compilado utilizando o Root, com o comando

```
.x exercicio_2.C
```

resultando o que foi pedido



((a)) Gráfico da Dispersão dos Dados ((b)) Gráfico da Conexão dos Dados

Figura 2: Resultado do exercício 2

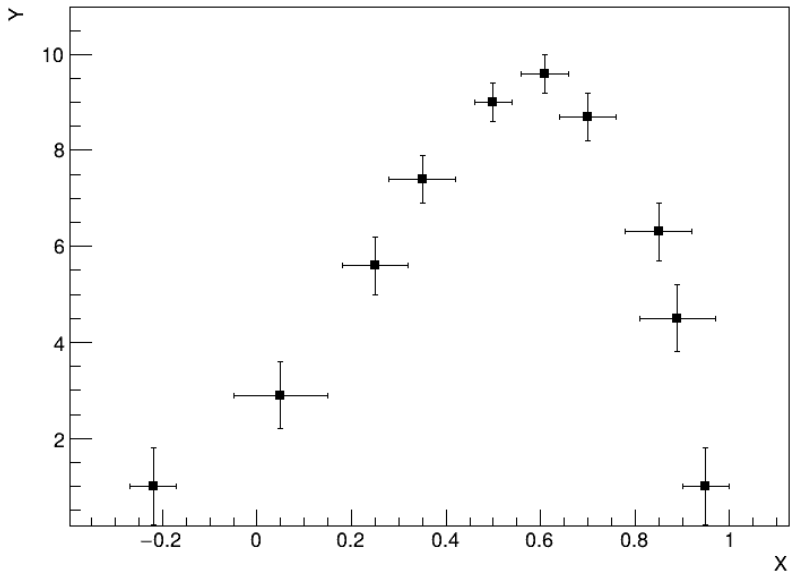


Figura 3: Resultado do exercício 2

EXERCICIO 3

Código em C

```
void exercicio_3() {

    TCanvas *c1 = new TCanvas("c1", "Canvas", 800, 600);
    TH1F *hist = new TH1F("hist", "", 50, 0, 10);
    TRandom3 *rand = new TRandom3();

    for (int i = 0; i < 10000; ++i) {
        double value = rand->Gaus(5, 2);
        hist->Fill(value);
    }

    hist->Draw();
    hist->SetStats(0);
    TLegend *legend = new TLegend(0.7, 0.7, 0.9, 0.9);
    legend->SetHeader("Estatistica", "C");

    legend->AddEntry((TObject*)0, Form("Entries: %d", (int)hist->GetEntries()), "");
    legend->AddEntry((TObject*)0, Form("Mean: %.2f", hist->GetMean()), "");
    legend->AddEntry((TObject*)0, Form("RMS: %.2f", hist->GetRMS()), "");
    legend->AddEntry((TObject*)0, Form("Integral: %.2f", hist->Integral()), "");
    legend->AddEntry((TObject*)0, Form("Underflows: %d", (int)hist->GetBinContent(0)), "");
    legend->AddEntry((TObject*)0, Form("Overflows: %d", (int)hist->GetBinContent(hist->GetNbinsX() +
    legend->AddEntry((TObject*)0, Form("Skewness: %.2f", hist->GetSkewness()), "");
    legend->AddEntry((TObject*)0, Form("Kurtosis: %.2f", hist->GetKurtosis()), "");
    legend->Draw();

    c1->SaveAs("histogram_estatistica.png");
}
}
```

Após isso, foi compilado utilizando o Root, com o comando

```
.x exercicio_3.C
```

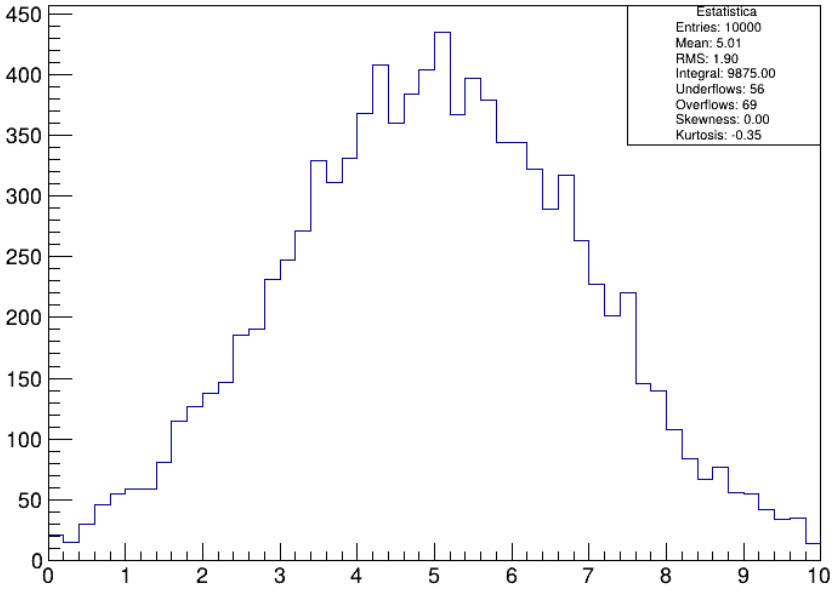


Figura 4: Resultado do exercício 3

EXERCICIO 4

Código em C

```
void exercicio_4()
{


    TCanvas *c1 = new TCanvas("c1", "Histograma de Momento", 800, 600);
    TFile *file = new TFile("tree.root");
    TTree *tree = (TTree*)file->Get("tree1");
    TH1F *hist = new TH1F("hist", "", 100, 100, 200);
    TH1F *histEbeam = new TH1F("histEbeam", "", 100, 0, 1000);
    float px, py, pz, ebeam;

    tree->SetBranchAddress("ebeam", &ebeam);
    tree->SetBranchAddress("px", &px);
    tree->SetBranchAddress("py", &py);
    tree->SetBranchAddress("pz", &pz);

    Int_t nEntries = tree->GetEntries();

    for (Int_t i = 0; i < nEntries; i++) {
        tree->GetEntry(i);
        histEbeam->Fill(ebeam);
    }

    float meanEbeam = histEbeam->GetMean();
    for (Int_t i = 0; i < nEntries; i++) {
        tree->GetEntry(i);
        if (ebeam > meanEbeam * 0.2) {
            float pMagnitude = sqrt(px * px + py * py + pz * pz);
            hist->Fill(pMagnitude);
        }
    }

    hist->Draw();
    c1->SaveAs("histograma_momento.png");
}
```

Após isso, foi compilado utilizando o Root, com o comando

```
.x exercicio_4.C
```

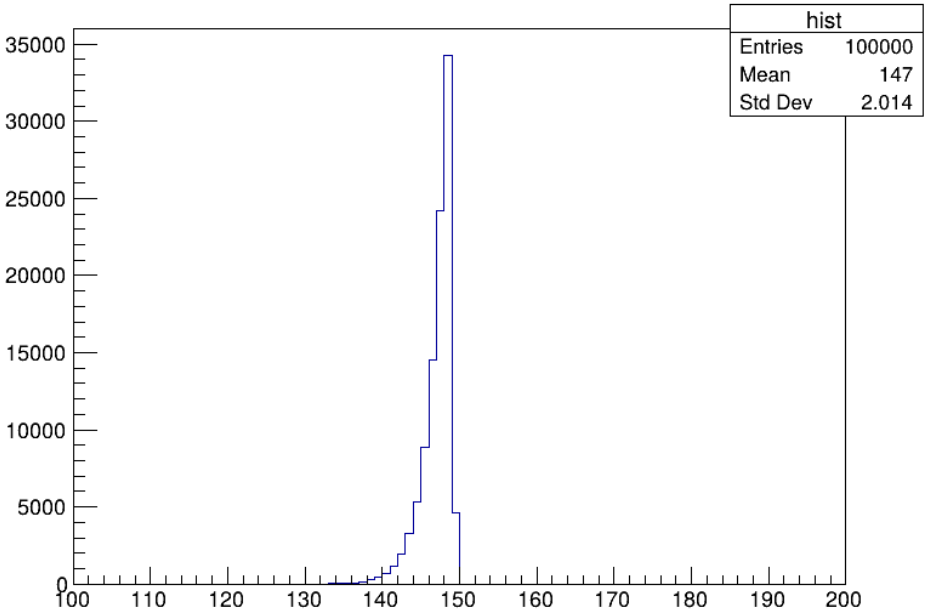


Figura 5: Resultado do exercício 3