**Introdução à Análise de dados em FAE** **(01/11/2024)**

# Exercicios de estatística para análise de dados em HEP

*Professores:* Eliza Melo, Dilson Damião e Mauricio Thiel *Name:* Thiago de Andrade Rangel Monteiro

## EXERCÍCIO 1

```
1
2  #include <TChain.h>
3  #include <TTreeReader.h>
4  #include <TTreeReaderArray.h>
5  #include <TCanvas.h>
6  #include <TH1F.h>
7  #include <TLorentzVector.h>
8  #include <TMath.h>
9  #include <iostream>
10 #include <vector>
11 #include <algorithm>
12 #include <numeric>
13
14 void analise() {
15
16     std::vector<std::string> diretorios = {
17         "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
               UL2016_MiniAODv2_NanoAODv9-v1/100000/*.root",
18         "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
               UL2016_MiniAODv2_NanoAODv9-v1/1010000/*.root",
19         "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
               UL2016_MiniAODv2_NanoAODv9-v1/250000/*.root"
20     };
21
22     TChain chain("Events");
23     for (const auto& path : diretorios) {
24         chain.Add(path.c_str());
25     }
26
27     // Eletrons
28     TTreeReader reader(&chain);
29     TTreeReaderArray<float> Electron_pt(reader, "Electron_pt");
30     TTreeReaderArray<float> Electron_eta(reader, "Electron_eta");
31     TTreeReaderArray<float> Electron_phi(reader, "Electron_phi");
32
33     // Muons
34     TTreeReaderArray<float> Muon_pt(reader, "Muon_pt");
35     TTreeReaderArray<float> Muon_eta(reader, "Muon_eta");
36     TTreeReaderArray<float> Muon_phi(reader, "Muon_phi");
37
38     // Taus
39     TTreeReaderArray<float> Tau_pt(reader, "Tau_pt");
40     TTreeReaderArray<float> Tau_eta(reader, "Tau_eta");
41     TTreeReaderArray<float> Tau_phi(reader, "Tau_phi");
42
43     // Jatos
44     TTreeReaderArray<float> Jet_pt(reader, "Jet_pt");
45     TTreeReaderArray<float> Jet_eta(reader, "Jet_eta");
46     TTreeReaderArray<float> Jet_phi(reader, "Jet_phi");
```

```
    // Criando histogramas
    TH1F* hElectronPt = new TH1F("hElectronPt", "Distribuicao de p_{T} dos Eletrons; p_{T} (GeV/c); Eventos", 50, 0, 200);
    TH1F* hElectronEta = new TH1F("hElectronEta", "Distribuicao de #eta dos Eletrons; #eta; Eventos", 50, -3, 3);
    TH1F* hElectronPhi = new TH1F("hElectronPhi", "Distribuicao de #phi dos Eletrons; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hMuonPt = new TH1F("hMuonPt", "Distribuicao de p_{T} dos Muons; p_{T} (GeV/c); Eventos", 50, 0, 200);
    TH1F* hMuonEta = new TH1F("hMuonEta", "Distribuicao de #eta dos Muons; #eta; Eventos", 50, -3, 3);
    TH1F* hMuonPhi = new TH1F("hMuonPhi", "Distribuicao de #phi dos Muons; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hTauPt = new TH1F("hTauPt", "Distribuicao de p_{T} dos Taus; p_{T} (GeV/c); Eventos", 50, 0, 200);
    TH1F* hTauEta = new TH1F("hTauEta", "Distribuicao de #eta dos Taus; #eta; Eventos", 50, -3, 3);
    TH1F* hTauPhi = new TH1F("hTauPhi", "Distribuicao de #phi dos Taus; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hJetPt = new TH1F("hJetPt", "Distribuicao de p_{T} dos Jatos; p_{T} (GeV/c); Eventos", 50, 0, 200);
    TH1F* hJetEta = new TH1F("hJetEta", "Distribuicao de #eta dos Jatos; #eta; Eventos", 50, -3, 3);
    TH1F* hJetPhi = new TH1F("hJetPhi", "Distribuicao de #phi dos Jatos; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hInvariantMassMuon = new TH1F("hInvariantMassMuon", "Massa Invariante dos Dois Muons de Maior p_{T}; m_{ll} (GeV/c^{2}); Eventos", 50, 0, 200);
    TH1F* hInvariantMassTau = new TH1F("hInvariantMassTau", "Massa Invariante dos Dois Taus de Maior p_{T}; m_{ll} (GeV/c^{2}); Eventos", 50, 0, 200);
    TH1F* hInvariantMassElectron = new TH1F("hInvariantMassElectron", "Massa Invariante dos Dois Eletrons de Maior p_{T}; m_{ll} (GeV/c^{2}); Eventos", 50, 0, 200);

    int eventos_analisados = 0;

    while (reader.Next()) {
        eventos_analisados++;
        if (eventos_analisados % 10000 == 0) {
            std::cout << "Eventos analisados: " << eventos_analisados << std::endl;
        }

        // Preenchendo histogramas de el trons
        for (int i = 0; i < Electron_pt.GetSize(); ++i) {
            hElectronPt->Fill(Electron_pt[i]);
            hElectronEta->Fill(Electron_eta[i]);
            hElectronPhi->Fill(Electron_phi[i]);
        }

        // Preenchendo histogramas de m ons
        for (int i = 0; i < Muon_pt.GetSize(); ++i) {
            hMuonPt->Fill(Muon_pt[i]);
            hMuonEta->Fill(Muon_eta[i]);
            hMuonPhi->Fill(Muon_phi[i]);
        }

        // Preenchendo histogramas de taus
        for (int i = 0; i < Tau_pt.GetSize(); ++i) {
            hTauPt->Fill(Tau_pt[i]);
            hTauEta->Fill(Tau_eta[i]);
```

```
48              hTauPhi->Fill(Tau_phi[i]);
49          }
50
51          // Preenchendo histogramas de jatos
52          for (int i = 0; i < Jet_pt.GetSize(); ++i) {
53              hJetPt->Fill(Jet_pt[i]);
54              hJetEta->Fill(Jet_eta[i]);
55              hJetPhi->Fill(Jet_phi[i]);
56          }
57
58          // Analisando massa invariante dos el trons
59          if (Electron_pt.GetSize() >= 2) {
60              std::vector<TLorentzVector> electrons;
61              for (int i = 0; i < Electron_pt.GetSize(); ++i) {
62                  TLorentzVector electron;
63                  electron.SetPtEtaPhiM(Electron_pt[i], Electron_eta[i], Electron_phi[i
                        ], 0.000511); // Massa do el tron
64                  electrons.push_back(electron);
65              }
66              std::sort(electrons.begin(), electrons.end(), [](const TLorentzVector& a,
                     const TLorentzVector& b) {
67                  return a.Pt() > b.Pt();
68              });
69              if (electrons.size() >= 2) {
70                  TLorentzVector invMassElectron = electrons[0] + electrons[1];
71                  hInvariantMassElectron->Fill(invMassElectron.M());
72              }
73          }
74
75          // Analisando massa invariante dos muons
76          if (Muon_pt.GetSize() >= 2) {
77              std::vector<TLorentzVector> muons;
78              for (int i = 0; i < Muon_pt.GetSize(); ++i) {
79                  TLorentzVector muon;
80                  muon.SetPtEtaPhiM(Muon_pt[i], Muon_eta[i], Muon_phi[i], 0.105658); //
                         Massa do m on
81                  muons.push_back(muon);
82              }
83              std::sort(muons.begin(), muons.end(), [](const TLorentzVector& a, const
                    TLorentzVector& b) {
84                  return a.Pt() > b.Pt();
85              });
86              if (muons.size() >= 2) {
87                  TLorentzVector invMassMuon = muons[0] + muons[1];
88                  hInvariantMassMuon->Fill(invMassMuon.M());
89              }
90          }
91
92          // Analisando massa invariante dos taus
93          if (Tau_pt.GetSize() >= 2) {
94              std::vector<TLorentzVector> taus;
95              for (int i = 0; i < Tau_pt.GetSize(); ++i) {
96                  TLorentzVector tau;
97                  tau.SetPtEtaPhiM(Tau_pt[i], Tau_eta[i], Tau_phi[i], 1.77682); //
                        Massa do tau
98                  taus.push_back(tau);
99              }
100             std::sort(taus.begin(), taus.end(), [](const TLorentzVector& a, const
                    TLorentzVector& b) {
101                 return a.Pt() > b.Pt();
102             });
103             if (taus.size() >= 2) {
104                 TLorentzVector invMassTau = taus[0] + taus[1];
```

```
                hInvariantMassTau->Fill(invMassTau.M());
            }
        }
    }

    // Criando canvas e plotando histogramas
    TCanvas *c1 = new TCanvas("c1", "Distribui  es el trons", 800, 600);
    c1->Divide(2, 2);
    c1->cd(1); hElectronPt->Draw();
    c1->cd(2); hElectronEta->Draw();
    c1->cd(3); hElectronPhi->Draw();
    c1->cd(4); hInvariantMassElectron->Draw();
    c1->SaveAs("distribuicoes_electron.png");
    delete c1;

    TCanvas *c2 = new TCanvas("c2", "Distribui  es m ons", 800, 600);
    c2->Divide(2, 2);
    c2->cd(1); hMuonPt->Draw();
    c2->cd(2); hMuonEta->Draw();
    c2->cd(3); hMuonPhi->Draw();
    c2->cd(4); hInvariantMassMuon->Draw();
    c2->SaveAs("distribuicoes_muons.png");
    delete c2;

    TCanvas *c3 = new TCanvas("c3", "Distribui  es Ta s", 800, 600);
    c3->Divide(2, 2);
    c3->cd(1); hTauPt->Draw();
    c3->cd(2); hTauEta->Draw();
    c3->cd(3); hTauPhi->Draw();
    c3->cd(4); hInvariantMassTau->Draw();
    c3->SaveAs("distribuicoes_tau.png");
    delete c3;

    TCanvas *c4 = new TCanvas("c4", "Distribui  es Jatos", 800, 600);
    c4->Divide(2, 2);
    c4->cd(1); hJetPt->Draw();
    c4->cd(2); hJetEta->Draw();
    c4->cd(3); hJetPhi->Draw();
    c4->SaveAs("distribuicoes_jatos.png");
    delete c4;

    // Deletando histogramas
    delete hElectronPt;
    delete hElectronEta;
    delete hElectronPhi;
    delete hMuonPt;
    delete hMuonEta;
    delete hMuonPhi;
    delete hTauPt;
    delete hTauEta;
    delete hTauPhi;
    delete hJetPt;
    delete hJetEta;
    delete hJetPhi;
    delete hInvariantMassMuon;
    delete hInvariantMassTau;
    delete hInvariantMassElectron;
}
```

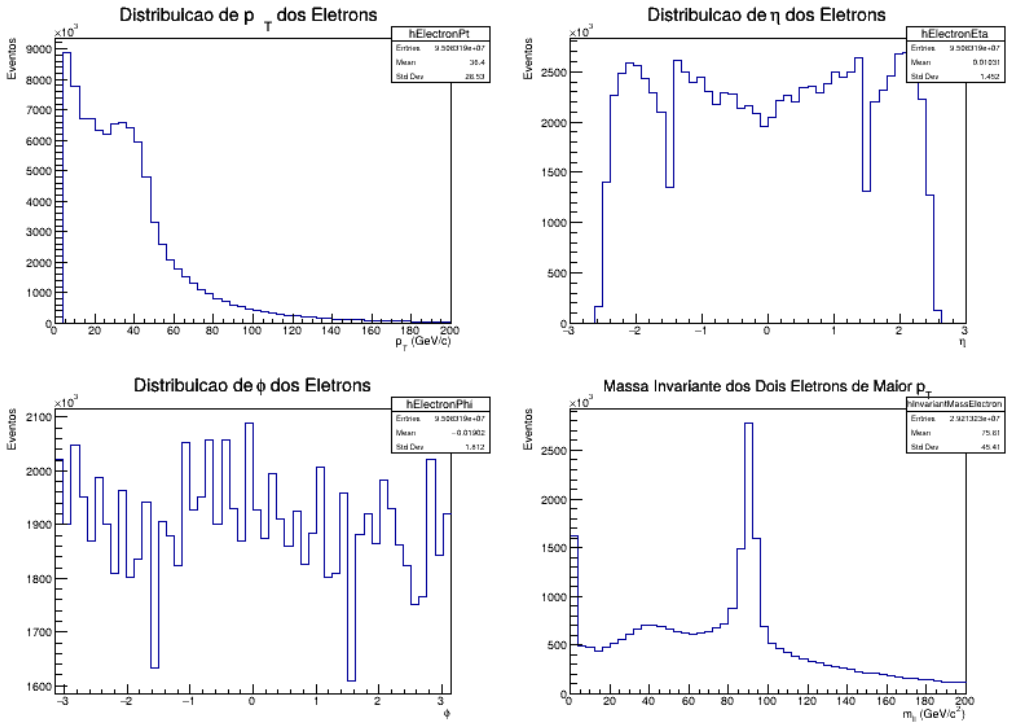


Figura 1: Caption

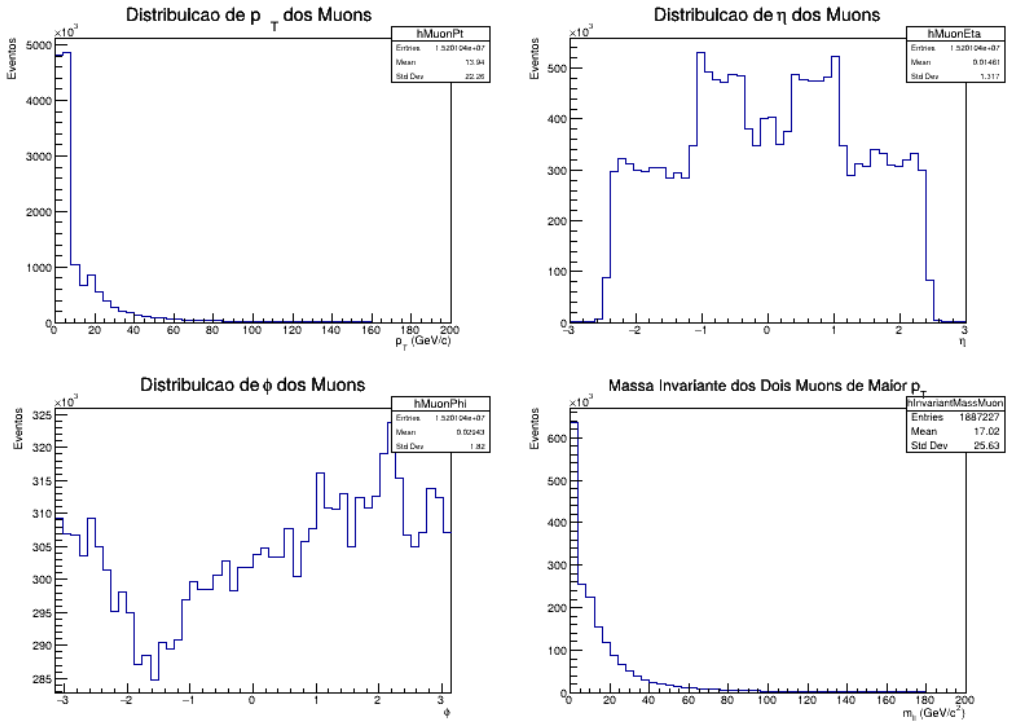


Figura 2: Caption

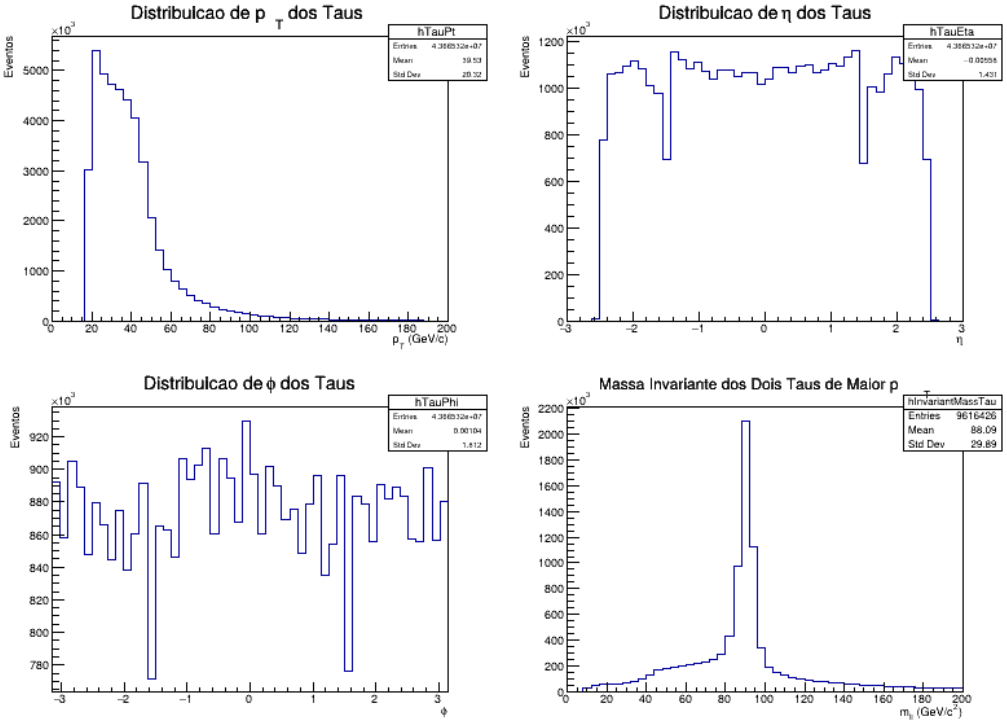


Figura 3: Caption

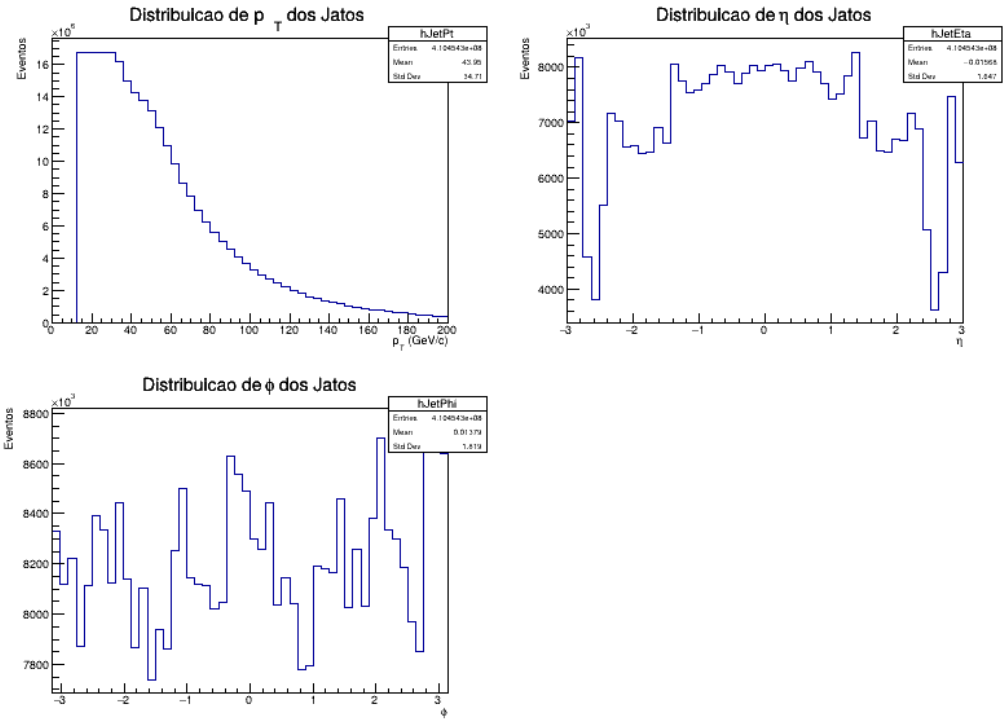


Figura 4: Caption