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

Lista 5

Professores: Dilson Damião, Eliza Melo e Maurício Thiel *Nome:* José Gonçalves Chaves Junior

EXERCICIO 1

Plotar os histogramas e ler os arquivos correspondentes ao grupo 3.

```
#include "TFile.h"
#include "TTreeReader.h"
#include "TMath.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TChain.h"
#include "TTreeReaderArray.h"
#include <vector>
#include <algorithm>
#include <numeric>
#include <iostream>

void read() {
    std::vector<std::string> diretorios = {
        "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
           UL2016_MiniAODv2_NanoAODv9-v1/100000/*.root",
        "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
           UL2016_MiniAODv2_NanoAODv9-v1/1010000/*.root",
        "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
           UL2016_MiniAODv2_NanoAODv9-v1/250000/*.root"
    };

    TChain chain("Events");
    for (const auto& path : diretorios) {
        chain.Add(path.c_str());
    }

    // Eletrons
    TTreeReader reader(&chain);
    TTreeReaderArray<float> Electron_pt(reader, "Electron_pt");
    TTreeReaderArray<float> Electron_eta(reader, "Electron_eta");
    TTreeReaderArray<float> Electron_phi(reader, "Electron_phi");

    // Muons
    TTreeReaderArray<float> Muon_pt(reader, "Muon_pt");
    TTreeReaderArray<float> Muon_eta(reader, "Muon_eta");
    TTreeReaderArray<float> Muon_phi(reader, "Muon_phi");

    // Taus
    TTreeReaderArray<float> Tau_pt(reader, "Tau_pt");
    TTreeReaderArray<float> Tau_eta(reader, "Tau_eta");
    TTreeReaderArray<float> Tau_phi(reader, "Tau_phi");

    // Jatos
    TTreeReaderArray<float> Jet_pt(reader, "Jet_pt");
    TTreeReaderArray<float> Jet_eta(reader, "Jet_eta");
    TTreeReaderArray<float> Jet_phi(reader, "Jet_phi");

    // Criando histogramas
```

```
47      TH1F* hElectronPt = new TH1F("hElectronPt", "p_{T} eletron distribution", 50, 0,
            200);
48      TH1F* hElectronEta = new TH1F("hElectronEta", "#Eta eletron distribution", 50,
            -3, 3);
49      TH1F* hElectronPhi = new TH1F("hElectronPhi", "#Pi electron distribution", 50, -
            TMath::Pi(), TMath::Pi());
50
51      TH1F* hMuonPt = new TH1F("hMuonPt", "p_{T} muon distribution ", 50, 0, 200);
52      TH1F* hMuonEta = new TH1F("hMuonEta", "#Eta muon distribution", 50, -3, 3);
53      TH1F* hMuonPhi = new TH1F("hMuonPhi", "#Pi muon distribution", 50, -TMath::Pi(),
            TMath::Pi());
54
55      TH1F* hTauPt = new TH1F("hTauPt", "p_{T} tau distribution", 50, 0, 200);
56      TH1F* hTauEta = new TH1F("hTauEta", "#Eta  tau distribution", 50, -3, 3);
57      TH1F* hTauPhi = new TH1F("hTauPhi", "#Phi  tau distribution", 50, -TMath::Pi(),
            TMath::Pi());
58
59      TH1F* hJetPt = new TH1F("hJetPt", "Jets distribution", 50, 0, 200);
60      TH1F* hJetEta = new TH1F("hJetEta", "#Eta Jets distribution", 50, -3, 3);
61      TH1F* hJetPhi = new TH1F("hJetPhi", "#Phi  Jets distribution", 50, -TMath::Pi(),
            TMath::Pi());
62
63      TH1F* hInvariantMassMuon = new TH1F("hInvariantMassMuon", "", 50, 0, 200);
64      TH1F* hInvariantMassTau = new TH1F("hInvariantMassTau", "", 50, 0, 200);
65      TH1F* hInvariantMassElectron = new TH1F("hInvariantMassElectron", "", 50, 0, 200)
            ;
66
67      int eventos_analisados = 0;
68
69      while (reader.Next()) {
70          eventos_analisados++;
71          if (eventos_analisados % 10000 == 0) {
72              std::cout << "Eventos analisados: " << eventos_analisados << std::endl;
73          }
74
75          // Preenchendo histogramas de el trons
76          for (int i = 0; i < Electron_pt.GetSize(); ++i) {
77              hElectronPt->Fill(Electron_pt[i]);
78              hElectronEta->Fill(Electron_eta[i]);
79              hElectronPhi->Fill(Electron_phi[i]);
80          }
81
82          // Preenchendo histogramas de m ons
83          for (int i = 0; i < Muon_pt.GetSize(); ++i) {
84              hMuonPt->Fill(Muon_pt[i]);
85              hMuonEta->Fill(Muon_eta[i]);
86              hMuonPhi->Fill(Muon_phi[i]);
87          }
88
89          // Preenchendo histogramas de taus
90          for (int i = 0; i < Tau_pt.GetSize(); ++i) {
91              hTauPt->Fill(Tau_pt[i]);
92              hTauEta->Fill(Tau_eta[i]);
93              hTauPhi->Fill(Tau_phi[i]);
94          }
95
96          // Preenchendo histogramas de jatos
97          for (int i = 0; i < Jet_pt.GetSize(); ++i) {
98              hJetPt->Fill(Jet_pt[i]);
99              hJetEta->Fill(Jet_eta[i]);
100             hJetPhi->Fill(Jet_phi[i]);
101         }
102
```

```
103         // Analisando massa invariante de el trons
104         if (Electron_pt.GetSize() >= 2) {
105             std::vector<TLorentzVector> electrons;
106             for (int i = 0; i < Electron_pt.GetSize(); ++i) {
107                 TLorentzVector electron;
108                 electron.SetPtEtaPhiM(Electron_pt[i], Electron_eta[i], Electron_phi[i
                        ], 0.000511); // Massa do el tron
109                 electrons.push_back(electron);
110             }
111             std::sort(electrons.begin(), electrons.end(), [](const TLorentzVector& a,
                     const TLorentzVector& b) {
112                 return a.Pt() > b.Pt();
113             });
114             if (electrons.size() >= 2) {
115                 TLorentzVector invMassElectron = electrons[0] + electrons[1];
116                 hInvariantMassElectron->Fill(invMassElectron.M());
117             }
118         }
119 
120         // Analisando massa invariante de muons
121         if (Muon_pt.GetSize() >= 2) {
122             std::vector<TLorentzVector> muons;
123             for (int i = 0; i < Muon_pt.GetSize(); ++i) {
124                 TLorentzVector muon;
125                 muon.SetPtEtaPhiM(Muon_pt[i], Muon_eta[i], Muon_phi[i], 0.105658); //
                         Massa do m on
126                 muons.push_back(muon);
127             }
128             std::sort(muons.begin(), muons.end(), [](const TLorentzVector& a, const
                    TLorentzVector& b) {
129                 return a.Pt() > b.Pt();
130             });
131             if (muons.size() >= 2) {
132                 TLorentzVector invMassMuon = muons[0] + muons[1];
133                 hInvariantMassMuon->Fill(invMassMuon.M());
134             }
135         }
136 
137         // Analisando massa invariante de taus
138         if (Tau_pt.GetSize() >= 2) {
139             std::vector<TLorentzVector> taus;
140             for (int i = 0; i < Tau_pt.GetSize(); ++i) {
141                 TLorentzVector tau;
142                 tau.SetPtEtaPhiM(Tau_pt[i], Tau_eta[i], Tau_phi[i], 1.77682); //
                        Massa do tau
143                 taus.push_back(tau);
144             }
145             std::sort(taus.begin(), taus.end(), [](const TLorentzVector& a, const
                    TLorentzVector& b) {
146                 return a.Pt() > b.Pt();
147             });
148             if (taus.size() >= 2) {
149                 TLorentzVector invMassTau = taus[0] + taus[1];
150                 hInvariantMassTau->Fill(invMassTau.M());
151             }
152         }
153     }
154 
155     // Criando canvas e plotando histogramas
156     //Plot dos momentos
157 TCanvas *c1 = new TCanvas("c1","Electrons distribution",1920,1080);
158     c1->Divide(2, 2);
159     c1->cd(1); hElectronPt->Draw();
```

```
160     c1->cd(2); hMuonPt->Draw();
161     c1->cd(3); hTauPt->Draw();
162     c1->cd(4); hJetPt->Draw();
163     c1->SaveAs("p_transversal.png");
164 
165     // Plot de eta
166     TCanvas *c2 = new TCanvas("c2", "Muons distribuction", 1920, 1080);
167     c2->Divide(2, 2);
168     c2->cd(1); hElectronEta->Draw();
169     c2->cd(2); hMuonEta->Draw();
170     c2->cd(3); hTauEta->Draw();
171     c2->cd(4); hJetEta->Draw();
172     c2->SaveAs("eta.png");
173 
174     // Plot Phi
175     TCanvas *c3 = new TCanvas("c3", "Taus distribuction", 1920, 1080);
176     c3->Divide(2, 2);
177     c3->cd(1); hElectronPhi->Draw();
178     c3->cd(2); hMuonPhi->Draw();
179     c3->cd(3); hTauPhi->Draw();
180     c3->cd(4); hJetPhi->Draw();
181     c3->SaveAs("phi.png");
182 
183     // Plot Massa invariante
184     TCanvas *c4 = new TCanvas("c4", "Jets distribution", 1920, 1080);
185     c4->Divide(2, 2);
186     c4->cd(1); hInvariantMassElectron->Draw();
187     c4->cd(2); hInvariantMassMuon->Draw();
188     c4->cd(3); hInvariantMassTau->Draw();
189     c4->SaveAs("massa_invariante.png");
190 
191 
192 }
```

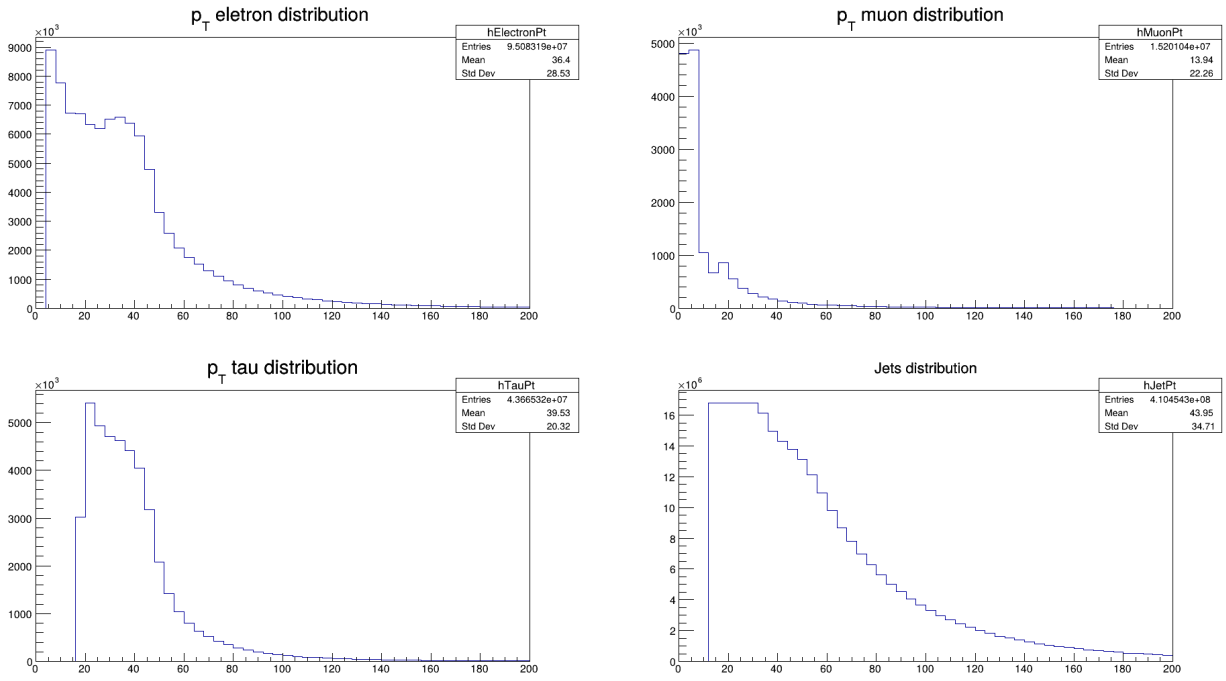

Figura 1: Momento transversal das partículas analisadas.

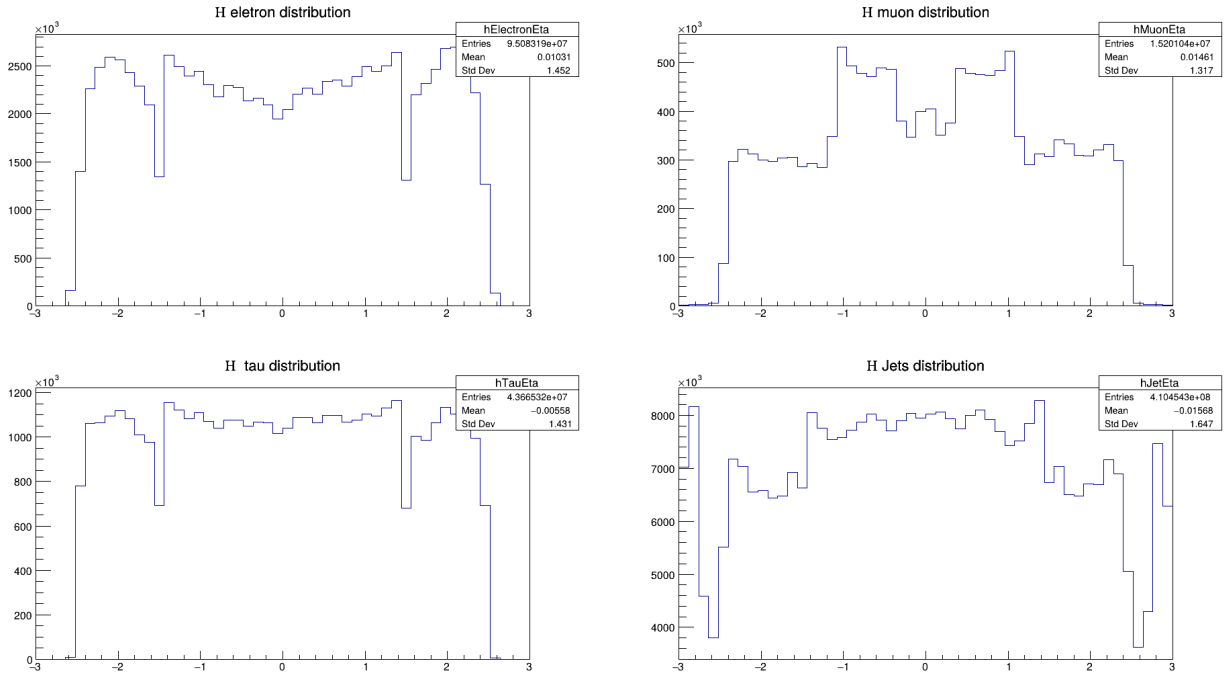

Figura 2: η das partículas analisadas.

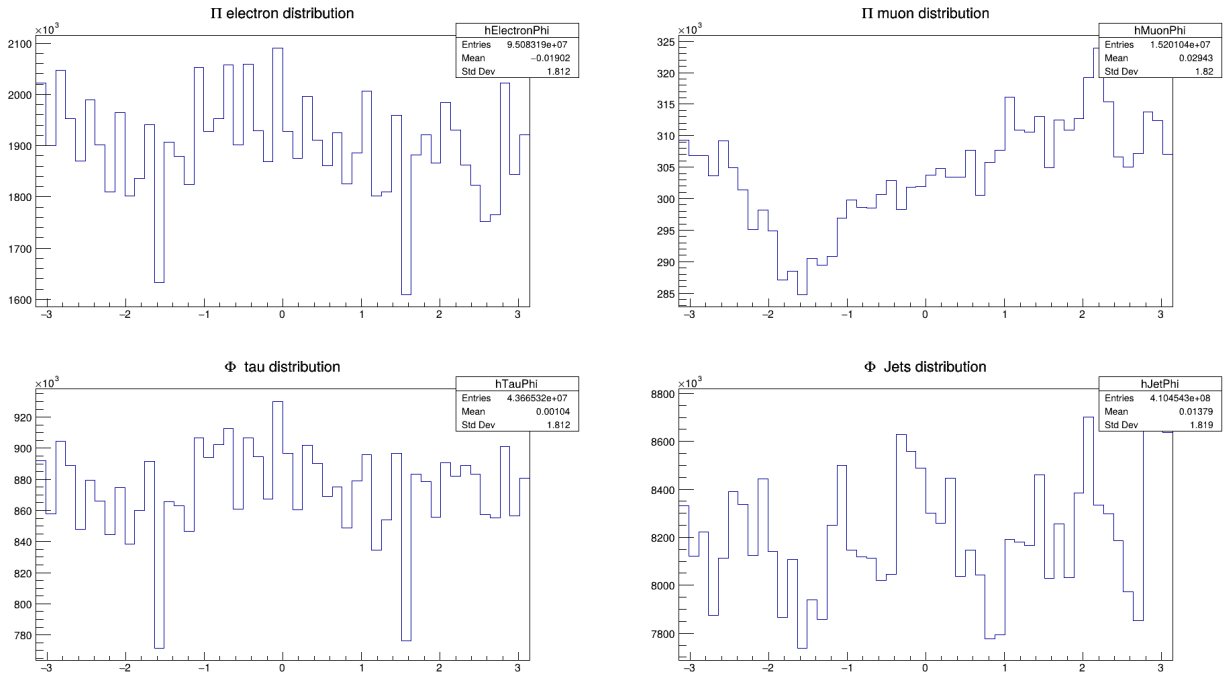

Figura 3: φ das partículas analisadas.

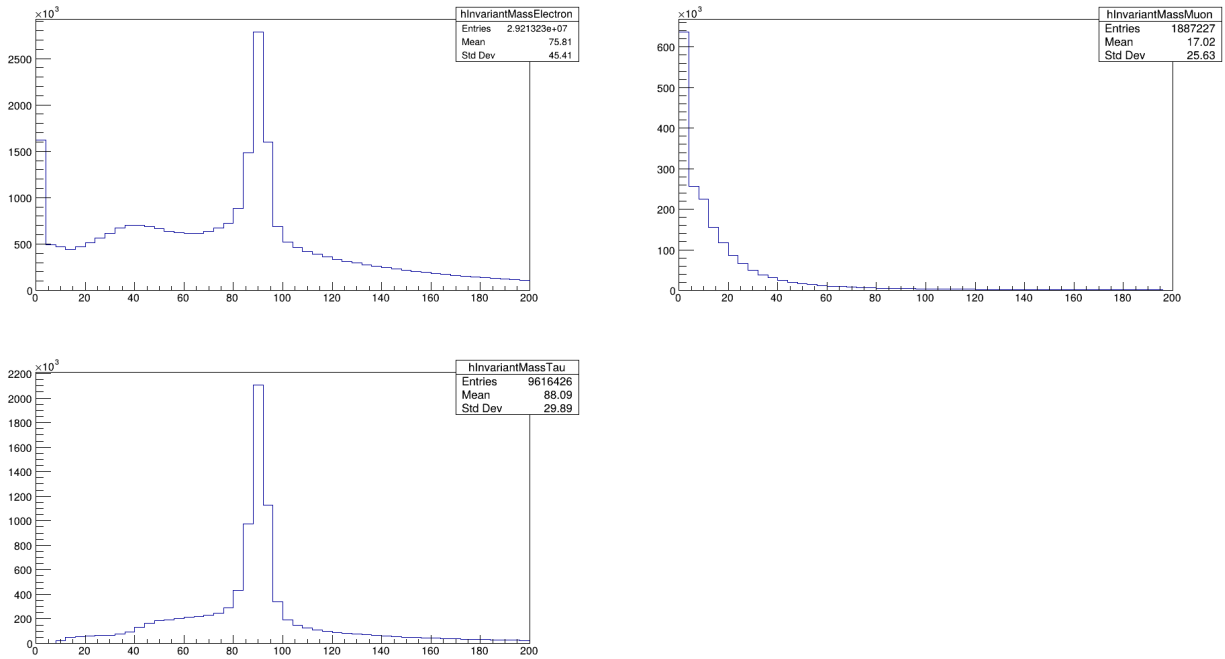

Figura 4: Massa invariante das partículas analisadas.