Introdução à Análise de dados em FAE **(12/11/2024)**

Exercício - Manipulando dados com o ROOT - Parte I

Professores: Dilson, Eliza & Maurício *Name:* Diogo Gomes Santos Caffonso de Morais

EXERCICIO

O código utilizado está abaixo. Foi executado de dentro do JupyterNotebook. Se trata do mesmo código do exercício anterior, com algumas alterações.

```
#include <TFile.h>
#include <TTree.h>
#include <TTreeReader.h>
#include <TTreeReaderArray.h>
#include <TCanvas.h>
#include <TH1F.h>
#include <TF1.h>
#include <TMath.h>
#include <iostream>
#include <vector>
#include <filesystem>

double calcular_massa_invariante(const std::vector<float>& pt, const std::vector<float>& eta, const std::vector<float>& phi) {
    if (pt.size() == 2) {
        return sqrt(2 * pt[0] * pt[1] * (TMath::CosH(eta[0] - eta[1]) - TMath::Cos(phi[0] - phi[1])));
    }
    return -1.0;
}

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

    std::vector<double> e_massas_invariantes, m_massas_invariantes, t_massas_invariantes;

    for (const auto& dir : diretorios) {
        for (const auto& entry : std::filesystem::directory_iterator(dir)) {
            std::string file_path = entry.path();
            TFile file(file_path.c_str(), "READ");
            if (!file.IsOpen()) continue;

            TTreeReader reader("Events", &file);
            TTreeReaderArray<float> Electron_pt(reader, "Electron_pt");
            TTreeReaderArray<float> Electron_eta(reader, "Electron_eta");
            TTreeReaderArray<float> Electron_phi(reader, "Electron_phi");
            TTreeReaderArray<float> Muon_pt(reader, "Muon_pt");
            TTreeReaderArray<float> Muon_eta(reader, "Muon_eta");
            TTreeReaderArray<float> Muon_phi(reader, "Muon_phi");
            TTreeReaderArray<float> Tau_pt(reader, "Tau_pt");
            TTreeReaderArray<float> Tau_eta(reader, "Tau_eta");
```

```
44              TTreeReaderArray<float> Tau_phi(reader, "Tau_phi");
45
46              while (reader.Next()) {
47                  for (size_t i = 0; i < Electron_pt.GetSize(); ++i) {
48                      for (size_t j = i + 1; j < Electron_pt.GetSize(); ++j) {
49                          if (Electron_pt[i] > 2 && Electron_pt[j] > 2 && abs(
                                Electron_eta[i]) > 2.4 && abs(Electron_eta[j]) > 2.4){
50                              e_massas_invariantes.push_back(
51                              calcular_massa_invariante(
52                              {Electron_pt[i], Electron_pt[j]},
53                              {Electron_eta[i], Electron_eta[j]},
54                              {Electron_phi[i], Electron_phi[j]}
55                              )
56                              );
57                          }
58                      }
59                  }
60                  for (size_t i = 0; i < Muon_pt.GetSize(); ++i) {
61                      for (size_t j = i + 1; j < Muon_pt.GetSize(); ++j) {
62                          if (Muon_pt[i] > 2 && Muon_pt[j] > 2 && abs(Muon_eta[i]) >
                                2.4 && abs(Muon_eta[j]) > 2.4){
63                              m_massas_invariantes.push_back(
64                              calcular_massa_invariante(
65                              {Muon_pt[i], Muon_pt[j]},
66                              {Muon_eta[i], Muon_eta[j]},
67                              {Muon_phi[i], Muon_phi[j]}
68                              )
69                              );
70                          }
71                      }
72                  }
73                  for (size_t i = 0; i < Tau_pt.GetSize(); ++i) {
74                      for (size_t j = i + 1; j < Tau_pt.GetSize(); ++j) {
75                          if (Tau_pt[i] > 2 && Tau_pt[j] > 2 && abs(Tau_eta[i]) > 2.4
                                && abs(Tau_eta[j]) > 2.4){
76                              t_massas_invariantes.push_back(
77                              calcular_massa_invariante(
78                              {Tau_pt[i], Tau_pt[j]},
79                              {Tau_eta[i], Tau_eta[j]},
80                              {Tau_phi[i], Tau_phi[j]}
81                              )
82                              );
83                          }
84                      }
85                  }
86              }
87          }
88      }
89
90      TCanvas* canvas = new TCanvas("canvas", "Distribui  es de Massas Invariantes",
            800, 600);
91      TH1F* hEletron = new TH1F("hEletron", "", 50, 0, 200);
92      TH1F* hMuon = new TH1F("hMuon", "", 50, 0, 200);
93      TH1F* hTau = new TH1F("hTau", "", 50, 0, 200);
94
95      for (const auto& massa : e_massas_invariantes) if (massa >= 0) hEletron->Fill(
            massa);
96      for (const auto& massa : m_massas_invariantes) if (massa >= 0) hMuon->Fill(massa)
            ;
97      for (const auto& massa : t_massas_invariantes) if (massa >= 0) hTau->Fill(massa);
98
99      TF1* gaussEletron = new TF1("gaussEletron", "[0]*exp(-0.5*((x-[1])/[2])^2) + [3]"
            , 80, 100);
```

```
100     gaussEletron->SetParameters(2000, 90, 15, 1000);
101     gaussEletron->SetLineColor(0);
102     hEletron->Fit(gaussEletron, "R");
103     double massaEletron = gaussEletron->GetParameter(1);
104     double erroEletron = gaussEletron->GetParError(1);
105     std::cout << "Massa do sinal (Eletron): " << massaEletron << " GeV/c^2" << std::
            endl;
106     std::cout << "Incerteza estat stica (Eletron): " << erroEletron << " GeV/c^2" <<
             std::endl;
107
108     TF1* gaussTau = new TF1("gaussTau", "[0]*exp(-0.5*((x-[1])/[2])^2) + [3]", 80,
            100);
109     gaussTau->SetParameters(800, 90, 15, 200);
110     gaussTau->SetLineColor(0);
111     hTau->Fit(gaussTau, "R");
112     double massaTau = gaussTau->GetParameter(1);
113     double erroTau = gaussTau->GetParError(1);
114     std::cout << "Massa do sinal (Tau): " << massaTau << " GeV/c^2" << std::endl;
115     std::cout << "Incerteza estat stica (Tau): " << erroTau << " GeV/c^2" << std::
            endl;
116
117     hEletron->SetLineColor(kBlue);
118     hEletron->SetStats(0);
119     hEletron->GetXaxis()->SetTitle("e_mass (GeV/c^{2})");
120     hEletron->GetYaxis()->SetTitle("Eventos");
121     hEletron->Draw();
122     canvas->SaveAs("e_massa_invariante.png");
123
124     hMuon->SetLineColor(kBlue);
125     hMuon->SetStats(0);
126     hMuon->GetXaxis()->SetTitle("#mu_mass (GeV/c^{2})");
127     hMuon->GetYaxis()->SetTitle("Eventos");
128     hMuon->Draw();
129     canvas->SaveAs("m_massa_invariante.png");
130
131     hTau->SetLineColor(kBlue);
132     hTau->SetStats(0);
133     hTau->GetXaxis()->SetTitle("#tau_mass (GeV/c^{2})");
134     hTau->GetYaxis()->SetTitle("Eventos");
135     hTau->Draw();
136     canvas->SaveAs("t_massa_invariante.png");
137
138     delete hEletron;
139     delete hMuon;
140     delete hTau;
141     delete canvas;
142 }
143
144 analise();
```

Agora o código analisa as combinações de pares presentes por evento, em vez de somente o primeiro par. Os histogramas obtidos foram os publicados neste repositório GitHub. Como pode-se ver, os resultados obtidos não conferem com a realidade. Não foi possível identificar um excesso de contagens para os muons, enquanto que para eletrons e taus, apesar de haver um excesso, este não está centrado na janela de massa esperada. As estimativas encontradas para a massa invariante do eletron e do tau foram, respectivamente,

$$m_e = (78,28 \pm 0,38) GeV/c^2$$

$$m_\tau = (79,32 \pm 0,60) GeV/c^2$$

Uma análise mais cuidadosa deve ser feita para identificar o motivo do erro.