Introdução à Análise de dados em FAE **(18 de Outubro de 2024)**

Lista 5

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

EXERCICIO 1

Plotar os histogramas correspondentes à amostra do grupo (Grupo 3 - ZZ 4e)

```cpp
#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());
    }

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

    // M ons
    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
    TH1F* hElectronPt = new TH1F("hElectronPt", "p_{T} eletron distribution", 50, 0, 200);
    TH1F* hElectronEta = new TH1F("hElectronEta", "#Eta eletron distribution", 50, -3, 3);
    TH1F* hElectronPhi = new TH1F("hElectronPhi", "#Pi electron distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hMuonPt = new TH1F("hMuonPt", "p_{T} muon distribution ", 50, 0, 200);
    TH1F* hMuonEta = new TH1F("hMuonEta", "#Eta muon distribution", 50, -3, 3);
    TH1F* hMuonPhi = new TH1F("hMuonPhi", "#Pi muon distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hTauPt = new TH1F("hTauPt", "p_{T} tau distribution", 50, 0, 200);
    TH1F* hTauEta = new TH1F("hTauEta", "#Eta  tau distribution", 50, -3, 3);
    TH1F* hTauPhi = new TH1F("hTauPhi", "#Phi  tau distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hJetPt = new TH1F("hJetPt", "Jets distribution", 50, 0, 200);
    TH1F* hJetEta = new TH1F("hJetEta", "#Eta Jets distribution", 50, -3, 3);
    TH1F* hJetPhi = new TH1F("hJetPhi", "#Phi  Jets distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hInvariantMassMuon = new TH1F("hInvariantMassMuon", "", 50, 0, 200);
    TH1F* hInvariantMassTau = new TH1F("hInvariantMassTau", "", 50, 0, 200);
    TH1F* hInvariantMassElectron = new TH1F("hInvariantMassElectron", "", 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]);
            hTauPhi->Fill(Tau_phi[i]);
        }

        // Preenchendo histogramas de jatos
        for (int i = 0; i < Jet_pt.GetSize(); ++i) {
            hJetPt->Fill(Jet_pt[i]);
            hJetEta->Fill(Jet_eta[i]);
            hJetPhi->Fill(Jet_phi[i]);
        }

        // Analisando massa invariante de el trons
        if (Electron_pt.GetSize() >= 2) {
            std::vector<TLorentzVector> electrons;
            for (int i = 0; i < Electron_pt.GetSize(); ++i) {
                TLorentzVector electron;
                electron.SetPtEtaPhiM(Electron_pt[i], Electron_eta[i], Electron_phi[i], 0.000511); // Massa do el tron
                electrons.push_back(electron);
            }
            std::sort(electrons.begin(), electrons.end(), [](const TLorentzVector& a, const TLorentzVector& b) {
                return a.Pt() > b.Pt();
            });
            if (electrons.size() >= 2) {
                TLorentzVector invMassElectron = electrons[0] + electrons[1];
                hInvariantMassElectron->Fill(invMassElectron.M());
            }
        }

        // Analisando massa invariante de muons
        if (Muon_pt.GetSize() >= 2) {
            std::vector<TLorentzVector> muons;
            for (int i = 0; i < Muon_pt.GetSize(); ++i) {
                TLorentzVector muon;
                muon.SetPtEtaPhiM(Muon_pt[i], Muon_eta[i], Muon_phi[i], 0.105658); // Massa do m on
                muons.push_back(muon);
            }
            std::sort(muons.begin(), muons.end(), [](const TLorentzVector& a, const TLorentzVector& b) {
                return a.Pt() > b.Pt();
            });
            if (muons.size() >= 2) {
                TLorentzVector invMassMuon = muons[0] + muons[1];
                hInvariantMassMuon->Fill(invMassMuon.M());
            }
        }

        // Analisando massa invariante de taus
        if (Tau_pt.GetSize() >= 2) {
            std::vector<TLorentzVector> taus;
            for (int i = 0; i < Tau_pt.GetSize(); ++i) {
                TLorentzVector tau;
                tau.SetPtEtaPhiM(Tau_pt[i], Tau_eta[i], Tau_phi[i], 1.77682); // Massa do tau
                taus.push_back(tau);
            }
            std::sort(taus.begin(), taus.end(), [](const TLorentzVector& a, const TLorentzVector& b) {
                return a.Pt() > b.Pt();
            });
            if (taus.size() >= 2) {
                TLorentzVector invMassTau = taus[0] + taus[1];
                hInvariantMassTau->Fill(invMassTau.M());
            }
        }
    }

    // Criando canvas e plotando histogramas
    //Plot dos momentos
 TCanvas *c1 = new TCanvas("c1","Electrons distribution",1920,1080);
    c1->Divide(2, 2);
    c1->cd(1); hElectronPt->Draw();
    c1->cd(2); hMuonPt->Draw();
    c1->cd(3); hTauPt->Draw();
    c1->cd(4); hJetPt->Draw();
    c1->SaveAs("p_transversal.png");

    // Plot de eta
    TCanvas *c2 = new TCanvas("c2", "Muons distribuction", 1920, 1080);
    c2->Divide(2, 2);
    c2->cd(1); hElectronEta->Draw();
    c2->cd(2); hMuonEta->Draw();
    c2->cd(3); hTauEta->Draw();
    c2->cd(4); hJetEta->Draw();
    c2->SaveAs("eta.png");

    // Plot Phi
    TCanvas *c3 = new TCanvas("c3", "Taus distribuction", 1920, 1080);
    c3->Divide(2, 2);
    c3->cd(1); hElectronPhi->Draw();
    c3->cd(2); hMuonPhi->Draw();
    c3->cd(3); hTauPhi->Draw();
    c3->cd(4); hJetPhi->Draw();
    c3->SaveAs("phi.png");

    // Plot Massa invariante
    TCanvas *c4 = new TCanvas("c4", "Jets distribution", 1920, 1080);
    c4->Divide(2, 2);
    c4->cd(1); hInvariantMassElectron->Draw();
    c4->cd(2); hInvariantMassMuon->Draw();
    c4->cd(3); hInvariantMassTau->Draw();
    c4->SaveAs("massa_invariante.png");


}
```

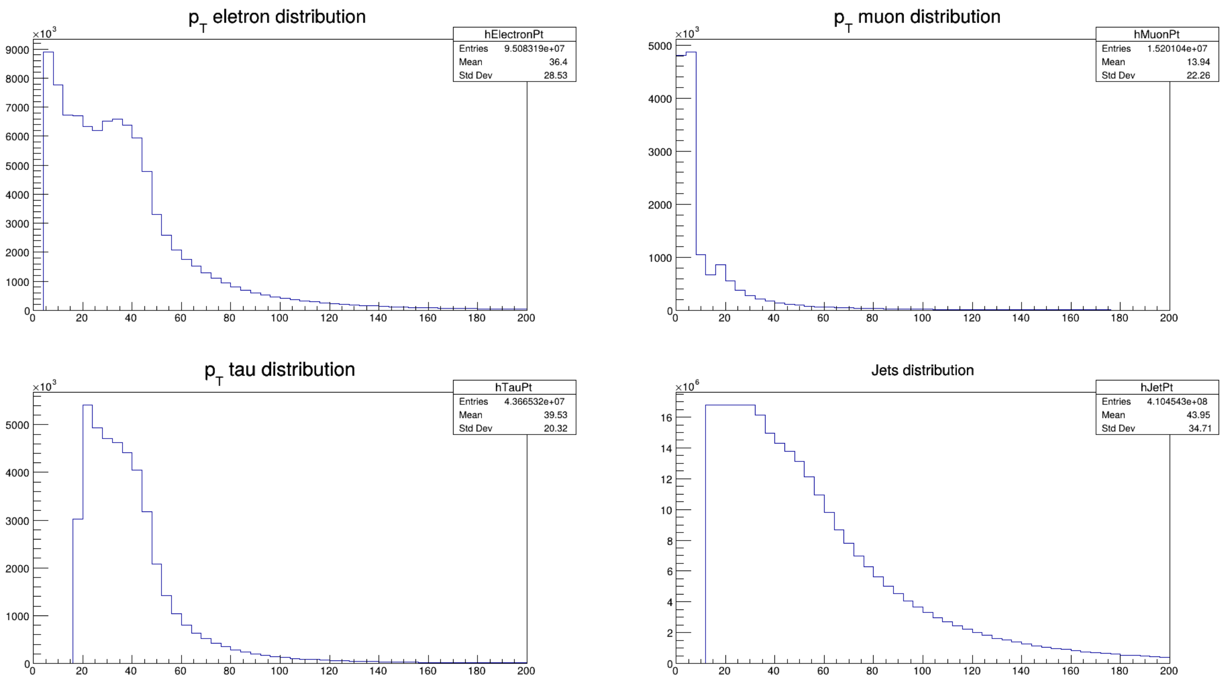

Figura 1: Momento transversal das partículas analisadas.

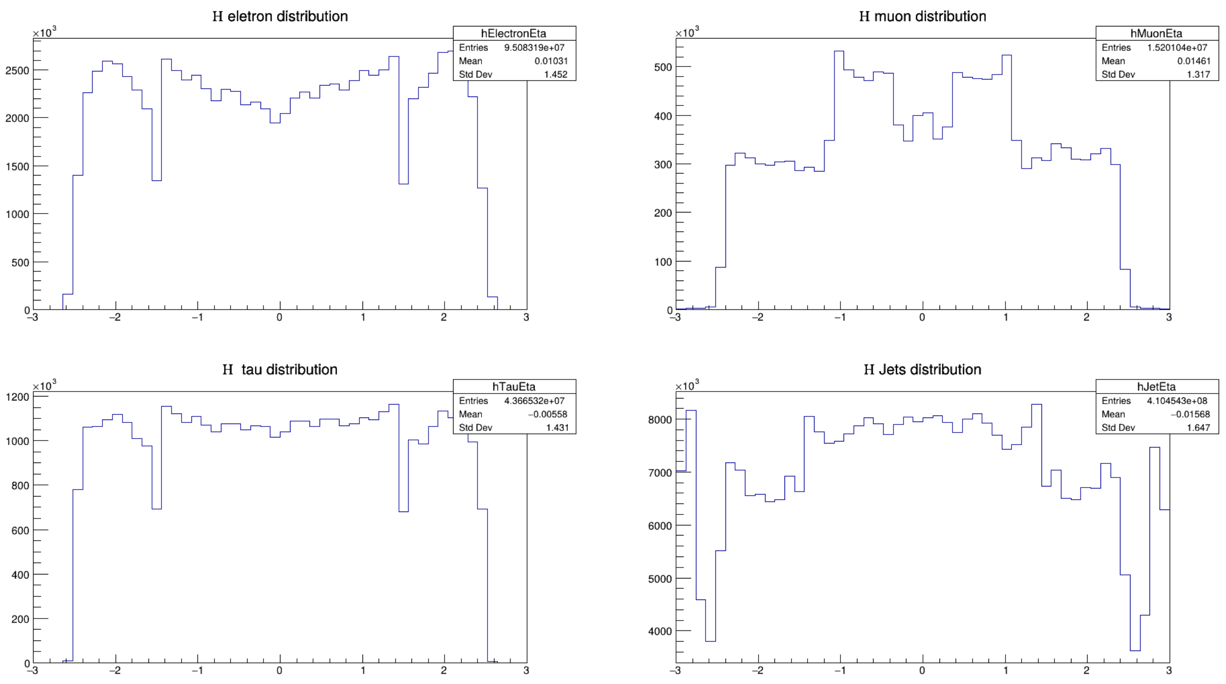

Figura 2: η das partículas analisadas.

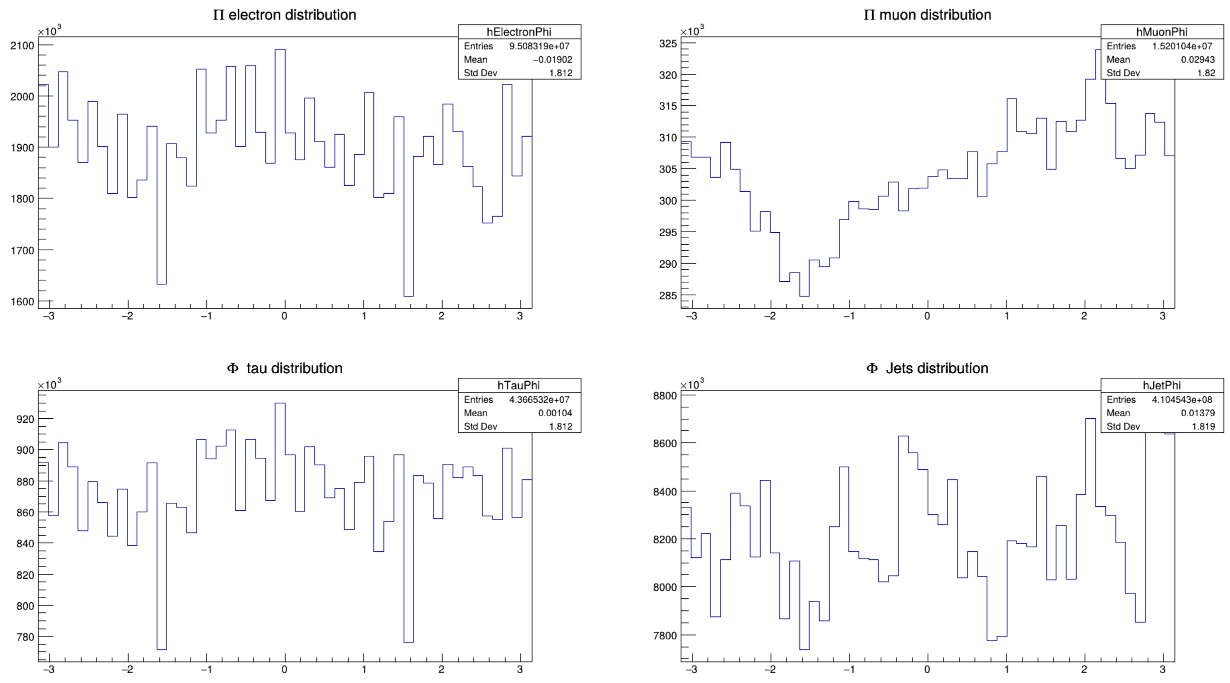

Figura 3: φ das partículas analisadas.

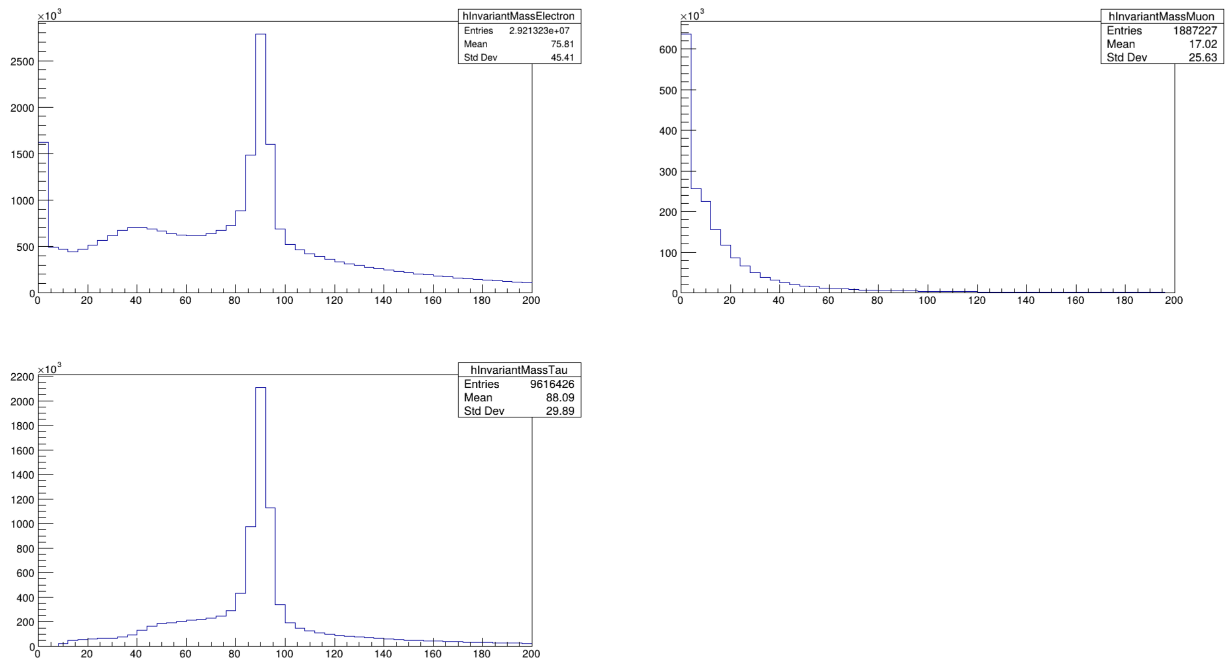

Figura 4: Massa invariante das partículas analisadas.