Introdução à Análise de dados em FAE **(DATA)**

Cinemática Relativística em Física de Partículas

Professores: Dilson de Jesus Damião, Eliza Melo da Costa e Maurcio Thiel
Name: Bruno Kron Guandalini e Pedro Oliveira

EXERCICIO 1

Utilizando as amostras do seu grupo, façam um código que:

- plote as distribuições de pT, η e ϕ dos objetos de estado final (léptons e jatos)
- calcule a massa invariante dos dois léptons com maior pT e plote-a
  $M^2 = 2p_{T1}p_{T2}\left(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2)\right)$
- salve as figuras no formato png
- adicione, tanto o código quanto os plots, no github 18

RESPOSTA:

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

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 Relatividade() {
    std::string diretorio = "/opendata/eos/opendata/cms/Run2016G/DoubleEG/NANOAOD/
        UL2016_MiniAODv2_NanoAODv9-v1/250000";

    std::vector<double> e_massas_invariantes;

    TH1F* hElectronPt = new TH1F("hElectronPt", "Electron p_{T} Distribution", 50, 0,
         200);
    TH1F* hMuonPt = new TH1F("hMuonPt", "Muon p_{T} Distribution", 50, 0, 200);
    TH1F* hTauPt = new TH1F("hTauPt", "Tau p_{T} Distribution", 50, 0, 200);
    TH1F* hJetPt = new TH1F("hJetPt", "Jet p_{T} Distribution", 50, 0, 200);

```

```
    TH1F* hElectronEta = new TH1F("hElectronEta", "Electron #eta Distribution", 50,
        -5, 5);
    TH1F* hMuonEta = new TH1F("hMuonEta", "Muon #eta Distribution", 50, -5, 5);
    TH1F* hTauEta = new TH1F("hTauEta", "Tau #eta Distribution", 50, -5, 5);
    TH1F* hJetEta = new TH1F("hJetEta", "Jet #eta Distribution", 50, -5, 5);

    TH1F* hElectronPhi = new TH1F("hElectronPhi", "Electron #phi Distribution", 50,
        -3.14, 3.14);
    TH1F* hMuonPhi = new TH1F("hMuonPhi", "Muon #phi Distribution", 50, -3.14, 3.14);
    TH1F* hTauPhi = new TH1F("hTauPhi", "Tau #phi Distribution", 50, -3.14, 3.14);
    TH1F* hJetPhi = new TH1F("hJetPhi", "Jet #phi Distribution", 50, -3.14, 3.14);

    for (const auto& entry : std::filesystem::directory_iterator(diretorio)) {
        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> Jet_pt(reader, "Jet_pt");
        TTreeReaderArray<float> Jet_eta(reader, "Jet_eta");
        TTreeReaderArray<float> Jet_phi(reader, "Jet_phi");
        TTreeReaderArray<float> Tau_pt(reader, "Tau_pt");
        TTreeReaderArray<float> Tau_eta(reader, "Tau_eta");
        TTreeReaderArray<float> Tau_phi(reader, "Tau_phi");

        while (reader.Next()) {
            for (size_t i = 0; i < Electron_pt.GetSize(); ++i) {
                hElectronPt->Fill(Electron_pt[i]);
                hElectronEta->Fill(Electron_eta[i]);
                hElectronPhi->Fill(Electron_phi[i]);
            }
            for (size_t i = 0; i < Muon_pt.GetSize(); ++i) {
                hMuonPt->Fill(Muon_pt[i]);
                hMuonEta->Fill(Muon_eta[i]);
                hMuonPhi->Fill(Muon_phi[i]);
            }
            for (size_t i = 0; i < Jet_pt.GetSize(); ++i) {
                hJetPt->Fill(Jet_pt[i]);
                hJetEta->Fill(Jet_eta[i]);
                hJetPhi->Fill(Jet_phi[i]);
            }
            for (size_t i = 0; i < Tau_pt.GetSize(); ++i) {
                hTauPt->Fill(Tau_pt[i]);
                hTauEta->Fill(Tau_eta[i]);
                hTauPhi->Fill(Tau_phi[i]);
            }

            std::vector<std::pair<float, int>> leptons;
            for (size_t i = 0; i < Electron_pt.GetSize(); ++i) {
                leptons.emplace_back(Electron_pt[i], i);
            }
            for (size_t i = 0; i < Muon_pt.GetSize(); ++i) {
                leptons.emplace_back(Muon_pt[i], i + Electron_pt.GetSize());
            }
            for (size_t i = 0; i < Tau_pt.GetSize(); ++i) {
                leptons.emplace_back(Tau_pt[i], i + Electron_pt.GetSize() + Muon_pt.
                    GetSize());
```

```
            }

            std::sort(leptons.rbegin(), leptons.rend());

            if (leptons.size() >= 2) {
                int idx1 = leptons[0].second;
                int idx2 = leptons[1].second;

                float pt1, eta1, phi1, pt2, eta2, phi2;

                if (idx1 < Electron_pt.GetSize()) {
                    pt1 = Electron_pt[idx1];
                    eta1 = Electron_eta[idx1];
                    phi1 = Electron_phi[idx1];
                } else if (idx1 < Electron_pt.GetSize() + Muon_pt.GetSize()) {
                    pt1 = Muon_pt[idx1 - Electron_pt.GetSize()];
                    eta1 = Muon_eta[idx1 - Electron_pt.GetSize()];
                    phi1 = Muon_phi[idx1 - Electron_pt.GetSize()];
                } else {
                    pt1 = Tau_pt[idx1 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                    eta1 = Tau_eta[idx1 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                    phi1 = Tau_phi[idx1 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                }

                if (idx2 < Electron_pt.GetSize()) {
                    pt2 = Electron_pt[idx2];
                    eta2 = Electron_eta[idx2];
                    phi2 = Electron_phi[idx2];
                } else if (idx2 < Electron_pt.GetSize() + Muon_pt.GetSize()) {
                    pt2 = Muon_pt[idx2 - Electron_pt.GetSize()];
                    eta2 = Muon_eta[idx2 - Electron_pt.GetSize()];
                    phi2 = Muon_phi[idx2 - Electron_pt.GetSize()];
                } else {
                    pt2 = Tau_pt[idx2 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                    eta2 = Tau_eta[idx2 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                    phi2 = Tau_phi[idx2 - Electron_pt.GetSize() - Muon_pt.GetSize()];
                }

                std::vector<float> pt_values = {pt1, pt2};
                std::vector<float> eta_values = {eta1, eta2};
                std::vector<float> phi_values = {phi1, phi2};
                double massa_invariante = calcular_massa_invariante(pt_values,
                    eta_values, phi_values);

                if (massa_invariante >= 0) {
                    e_massas_invariantes.push_back(massa_invariante);
                }
            }
        }
    }


    TCanvas* canvas;

    canvas = new TCanvas("canvasElectronPt", "Electron p_{T} Distribution", 800, 600)
        ;
    hElectronPt->SetLineColor(kBlue);
    hElectronPt->Draw();
    hElectronPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
    hElectronPt->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("electron_pt_distribution.png");
    delete canvas;

```

```
151     canvas = new TCanvas("canvasMuonPt", "Muon p_{T} Distribution", 800, 600);
152     hMuonPt->SetLineColor(kRed);
153     hMuonPt->Draw();
154     hMuonPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
155     hMuonPt->GetYaxis()->SetTitle("Events");
156     canvas->SaveAs("muon_pt_distribution.png");
157     delete canvas;
158 
159     canvas = new TCanvas("canvasTauPt", "Tau p_{T} Distribution", 800, 600);
160     hTauPt->SetLineColor(kMagenta);
161     hTauPt->Draw();
162     hTauPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
163     hTauPt->GetYaxis()->SetTitle("Events");
164     canvas->SaveAs("tau_pt_distribution.png");
165     delete canvas;
166 
167     canvas = new TCanvas("canvasJetPt", "Jet p_{T} Distribution", 800, 600);
168     hJetPt->SetLineColor(kGreen);
169     hJetPt->Draw();
170     hJetPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
171     hJetPt->GetYaxis()->SetTitle("Events");
172     canvas->SaveAs("jet_pt_distribution.png");
173     delete canvas;
174 
175     canvas = new TCanvas("canvasElectronEta", "Electron #eta Distribution", 800, 600)
        ;
176     hElectronEta->SetLineColor(kBlue);
177     hElectronEta->Draw();
178     hElectronEta->GetXaxis()->SetTitle("#eta");
179     hElectronEta->GetYaxis()->SetTitle("Events");
180     canvas->SaveAs("electron_eta_distribution.png");
181     delete canvas;
182 
183     canvas = new TCanvas("canvasMuonEta", "Muon #eta Distribution", 800, 600);
184     hMuonEta->SetLineColor(kRed);
185     hMuonEta->Draw();
186     hMuonEta->GetXaxis()->SetTitle("#eta");
187     hMuonEta->GetYaxis()->SetTitle("Events");
188     canvas->SaveAs("muon_eta_distribution.png");
189     delete canvas;
190 
191     canvas = new TCanvas("canvasTauEta", "Tau #eta Distribution", 800, 600);
192     hTauEta->SetLineColor(kMagenta);
193     hTauEta->Draw();
194     hTauEta->GetXaxis()->SetTitle("#eta");
195     hTauEta->GetYaxis()->SetTitle("Events");
196     canvas->SaveAs("tau_eta_distribution.png");
197     delete canvas;
198 
199     canvas = new TCanvas("canvasJetEta", "Jet #eta Distribution", 800, 600);
200     hJetEta->SetLineColor(kGreen);
201     hJetEta->Draw();
202     hJetEta->GetXaxis()->SetTitle("#eta");
203     hJetEta->GetYaxis()->SetTitle("Events");
204     canvas->SaveAs("jet_eta_distribution.png");
205     delete canvas;
206 
207     canvas = new TCanvas("canvasElectronPhi", "Electron #phi Distribution", 800, 600)
        ;
208     hElectronPhi->SetLineColor(kBlue);
209     hElectronPhi->Draw();
210     hElectronPhi->GetXaxis()->SetTitle("#phi");
211     hElectronPhi->GetYaxis()->SetTitle("Events");
```

```
212     canvas->SaveAs("electron_phi_distribution.png");
213     delete canvas;
214 
215     canvas = new TCanvas("canvasMuonPhi", "Muon #phi Distribution", 800, 600);
216     hMuonPhi->SetLineColor(kRed);
217     hMuonPhi->Draw();
218     hMuonPhi->GetXaxis()->SetTitle("#phi");
219     hMuonPhi->GetYaxis()->SetTitle("Events");
220     canvas->SaveAs("muon_phi_distribution.png");
221     delete canvas;
222 
223     canvas = new TCanvas("canvasTauPhi", "Tau #phi Distribution", 800, 600);
224     hTauPhi->SetLineColor(kMagenta);
225     hTauPhi->Draw();
226     hTauPhi->GetXaxis()->SetTitle("#phi");
227     hTauPhi->GetYaxis()->SetTitle("Events");
228     canvas->SaveAs("tau_phi_distribution.png");
229     delete canvas;
230 
231     canvas = new TCanvas("canvasJetPhi", "Jet #phi Distribution", 800, 600);
232     hJetPhi->SetLineColor(kGreen);
233     hJetPhi->Draw();
234     hJetPhi->GetXaxis()->SetTitle("#phi");
235     hJetPhi->GetYaxis()->SetTitle("Events");
236     canvas->SaveAs("jet_phi_distribution.png");
237     delete canvas;
238 
239     TH1F* hMassaInvariante = new TH1F("hMassaInvariante", "Invariant Mass
            Distribution", 50, 0, 200);
240     for (const auto& massa : e_massas_invariantes) {
241         if (massa >= 0) hMassaInvariante->Fill(massa);
242     }
243 
244     canvas = new TCanvas("canvasInvariantMass", "Invariant Mass Distribution", 800,
            600);
245     hMassaInvariante->SetLineColor(kBlack);
246     canvas->SetLogy();
247     hMassaInvariante->Draw();
248     hMassaInvariante->GetXaxis()->SetTitle("Invariant Mass (GeV/c^{2})");
249     hMassaInvariante->GetYaxis()->SetTitle("Events");
250     canvas->SaveAs("invariant_mass_distribution.png");
251     delete canvas;
252 
253 
254     delete hElectronPt;
255     delete hMuonPt;
256     delete hTauPt;
257     delete hJetPt;
258     delete hElectronEta;
259     delete hMuonEta;
260     delete hTauEta;
261     delete hJetEta;
262     delete hElectronPhi;
263     delete hMuonPhi;
264     delete hTauPhi;
265     delete hJetPhi;
266     delete hMassaInvariante;
267 }
```

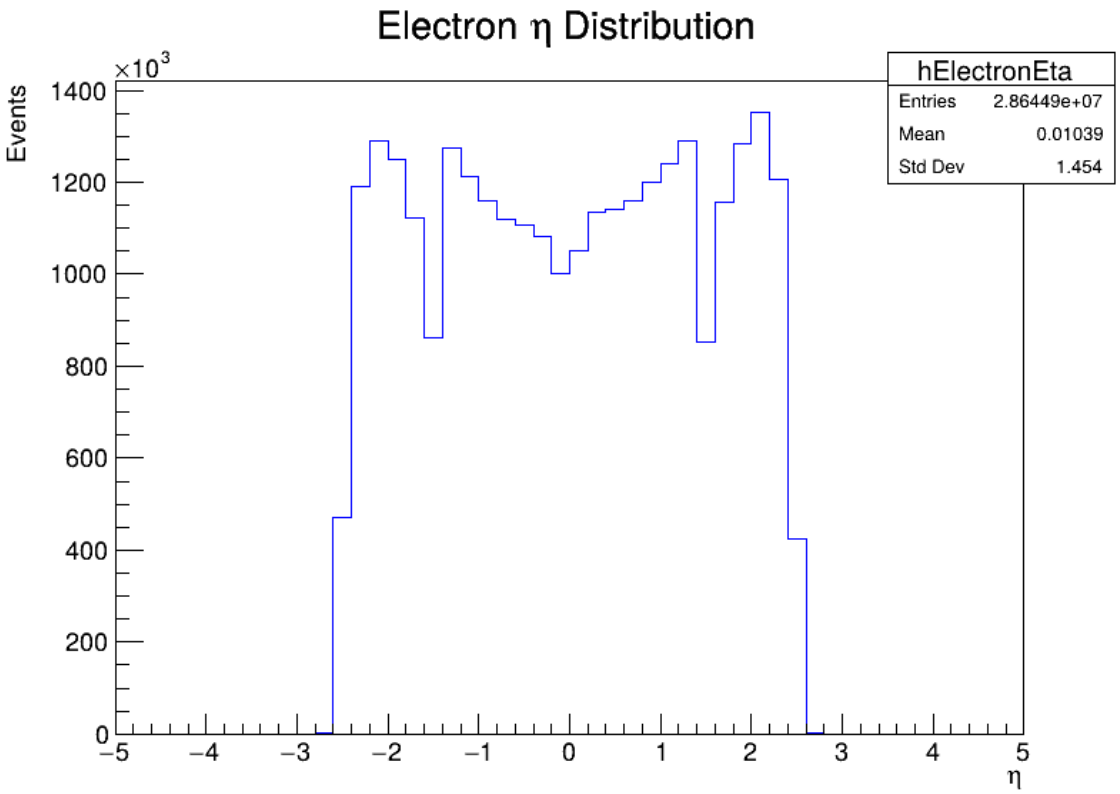

Figura 1: η do Elétron

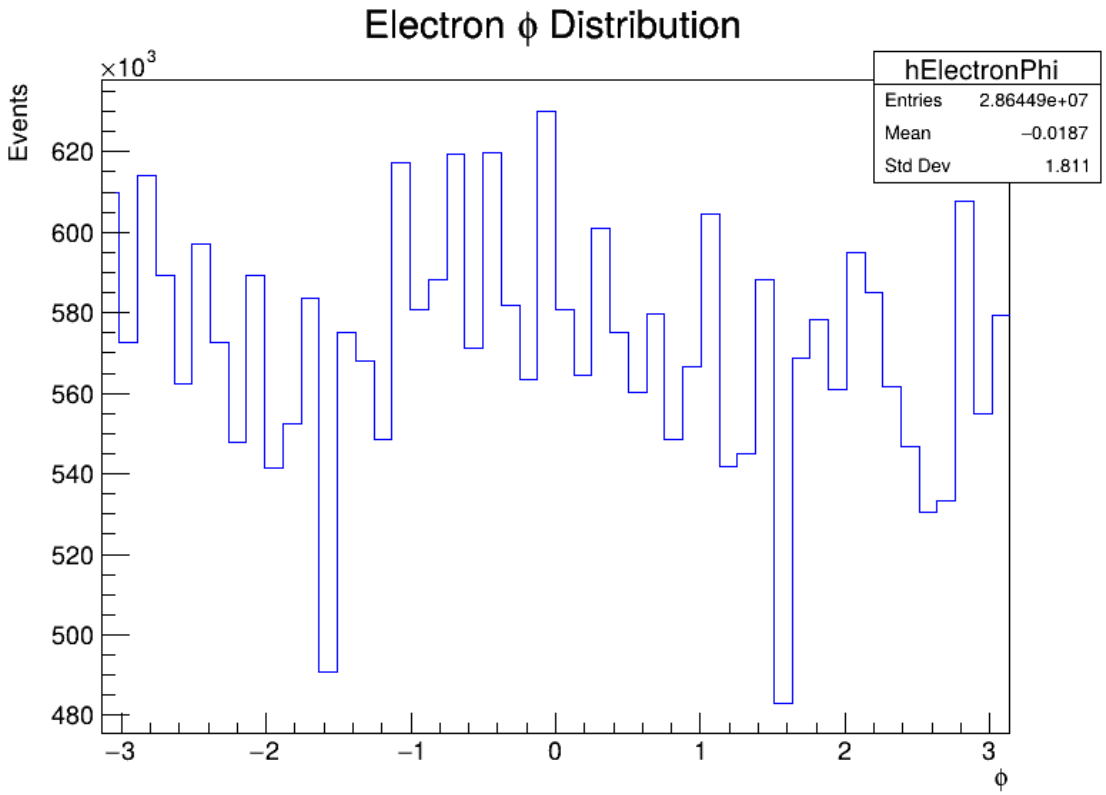

Figura 2: ϕ do Elétron

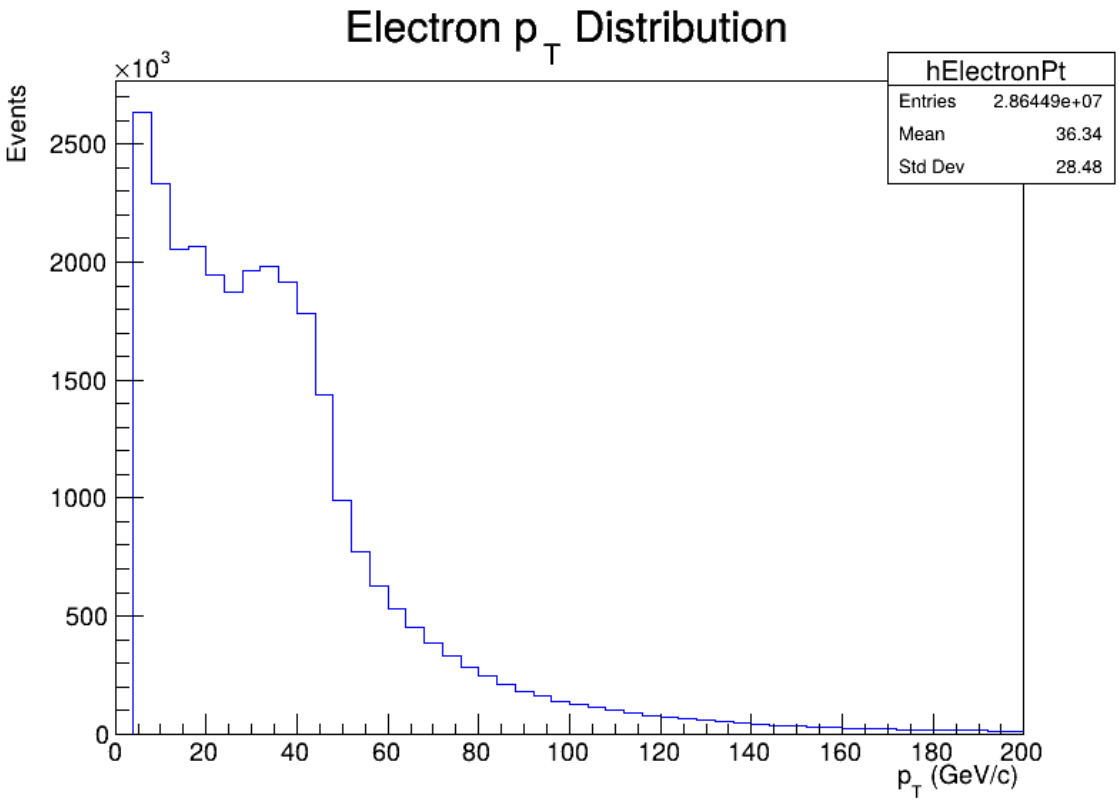

Figura 3: p_T do Elétron

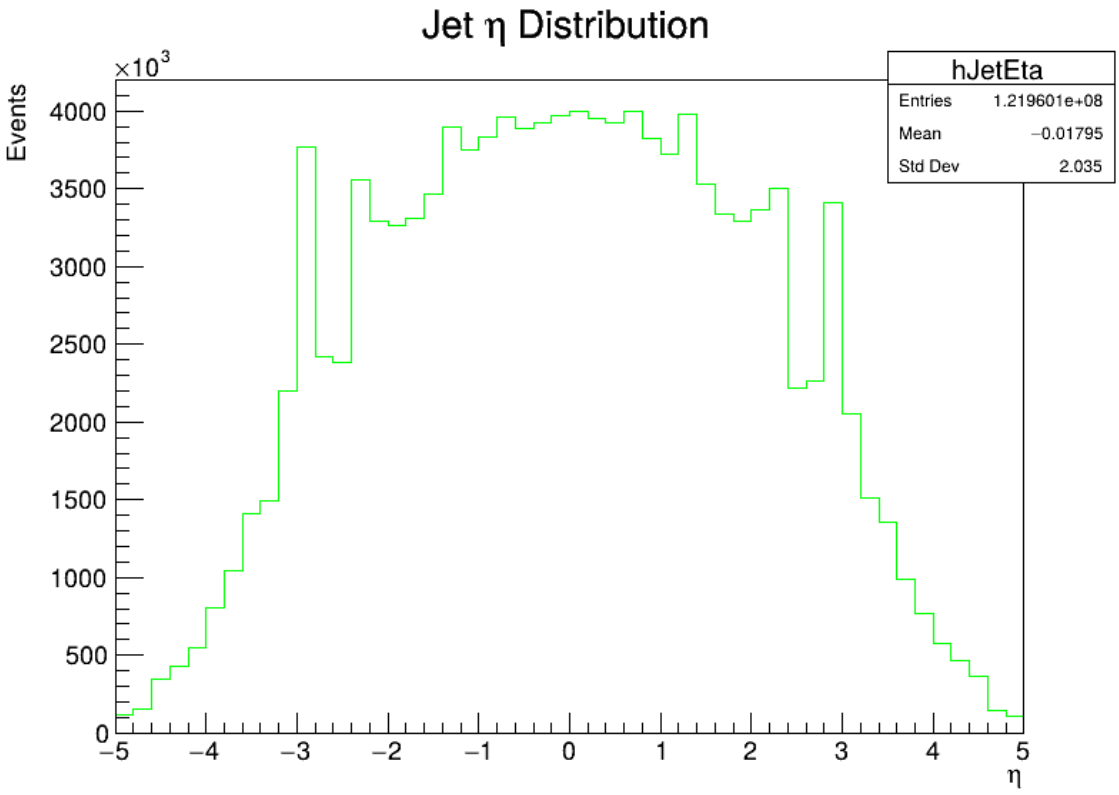

Figura 4: η do Jato

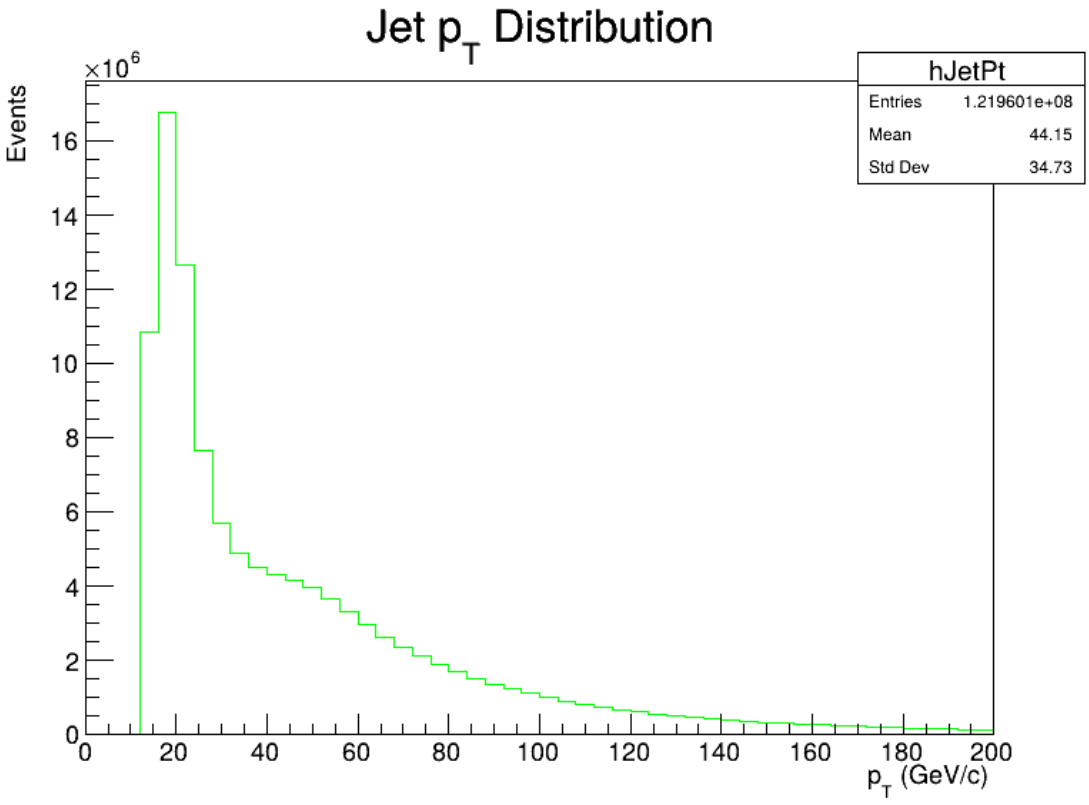

Figura 5: p_T do Jato

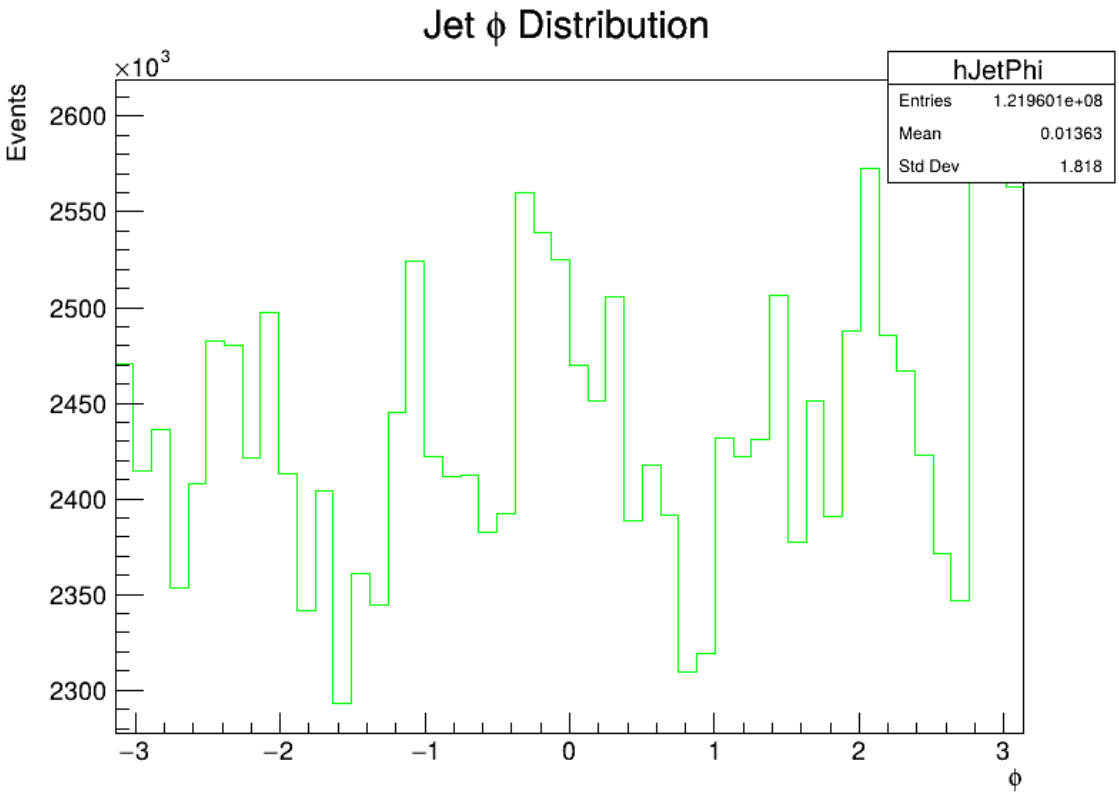

Figura 6: ϕ do Jato

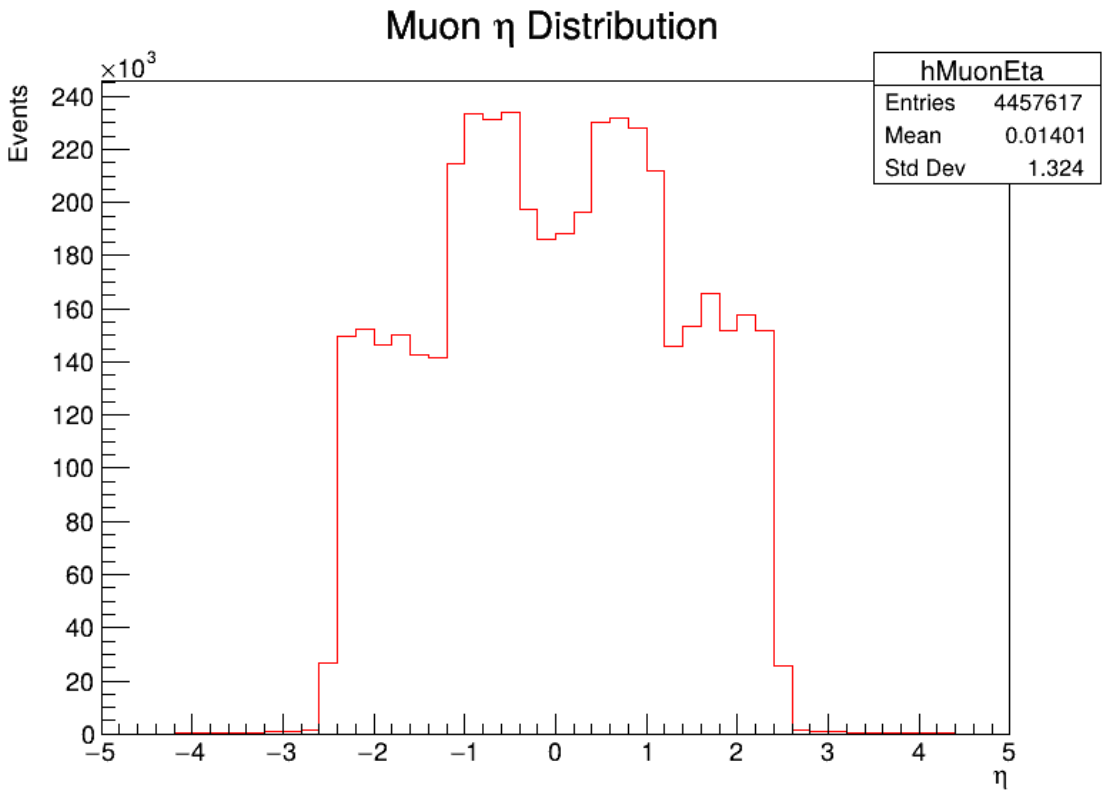

Figura 7: η do Múon

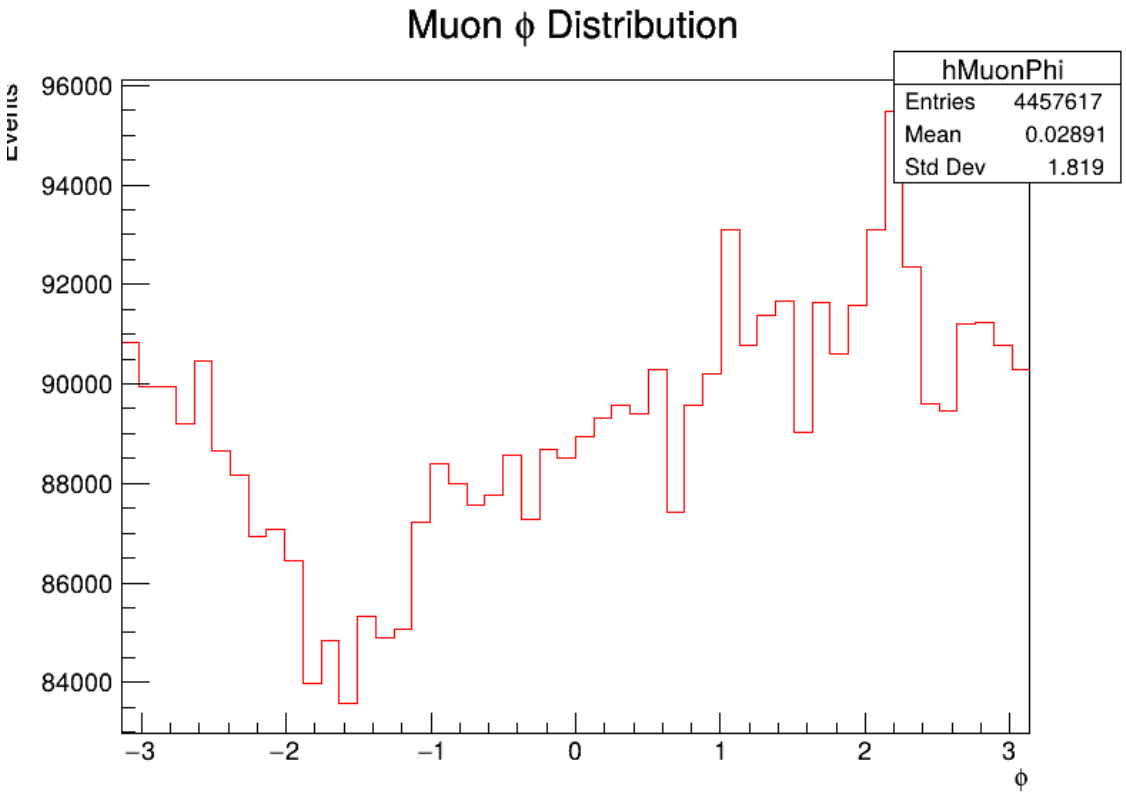

Figura 8: ϕ do Múon

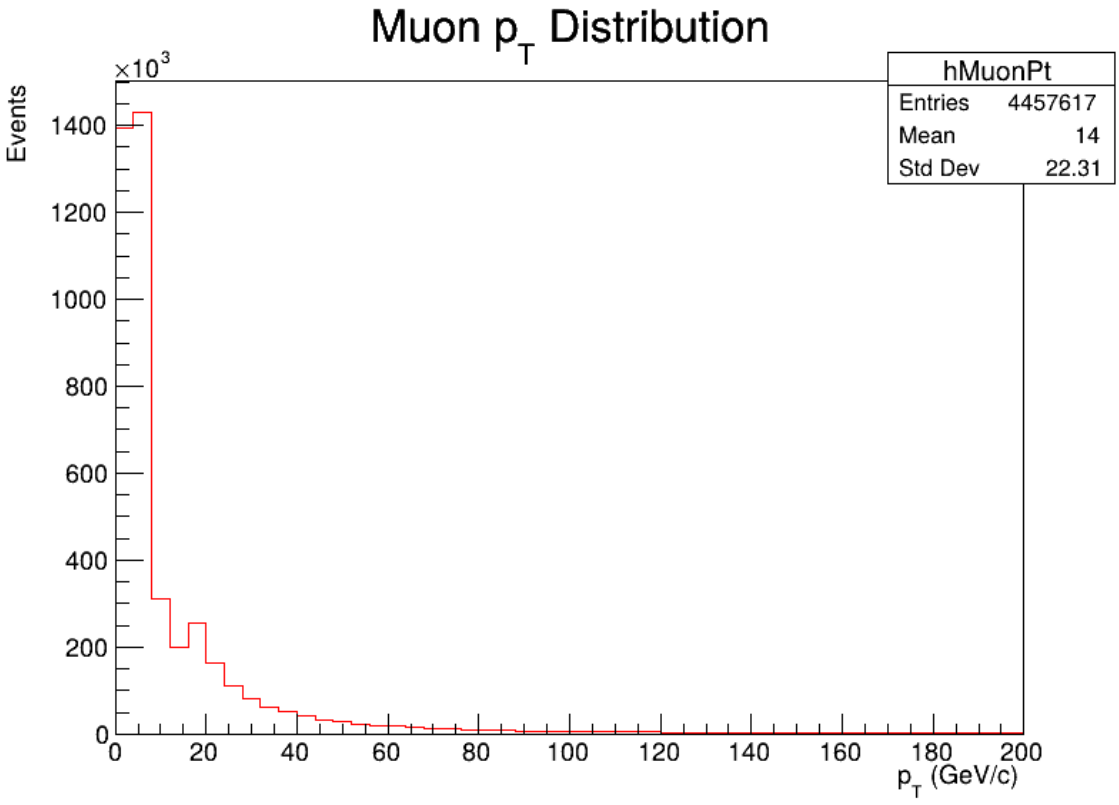

Figura 9: p_T do Múon

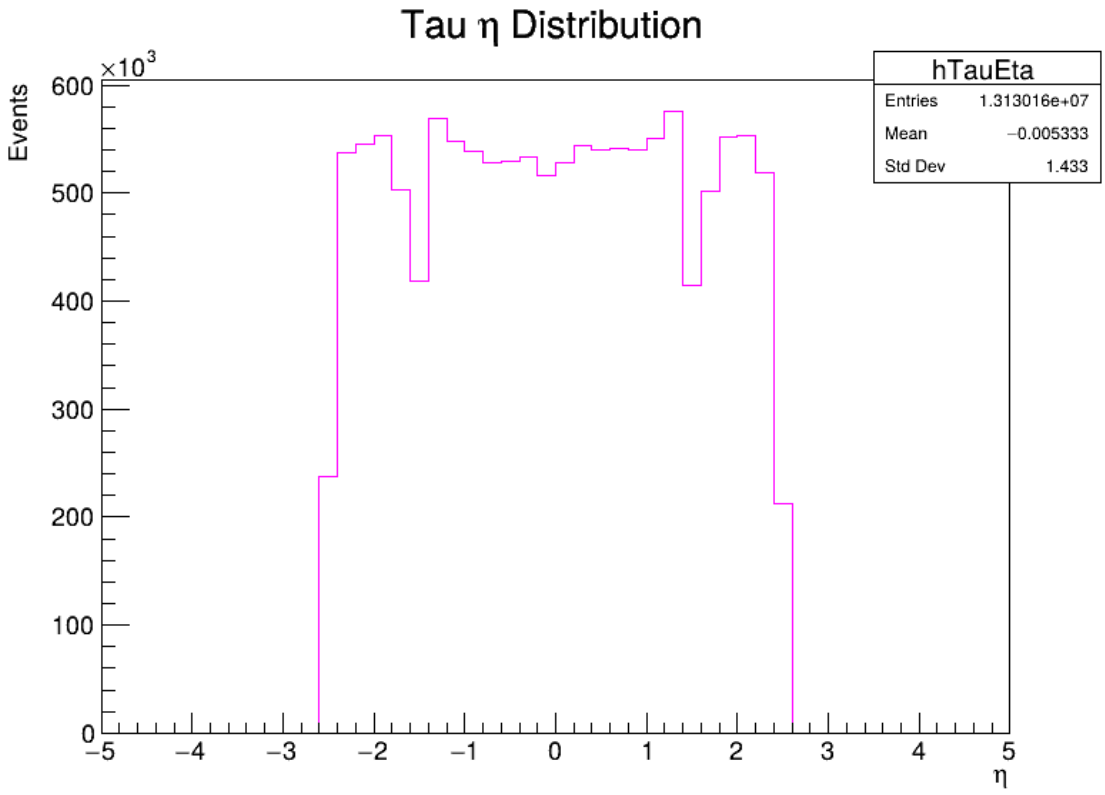

Figura 10: η do Tau

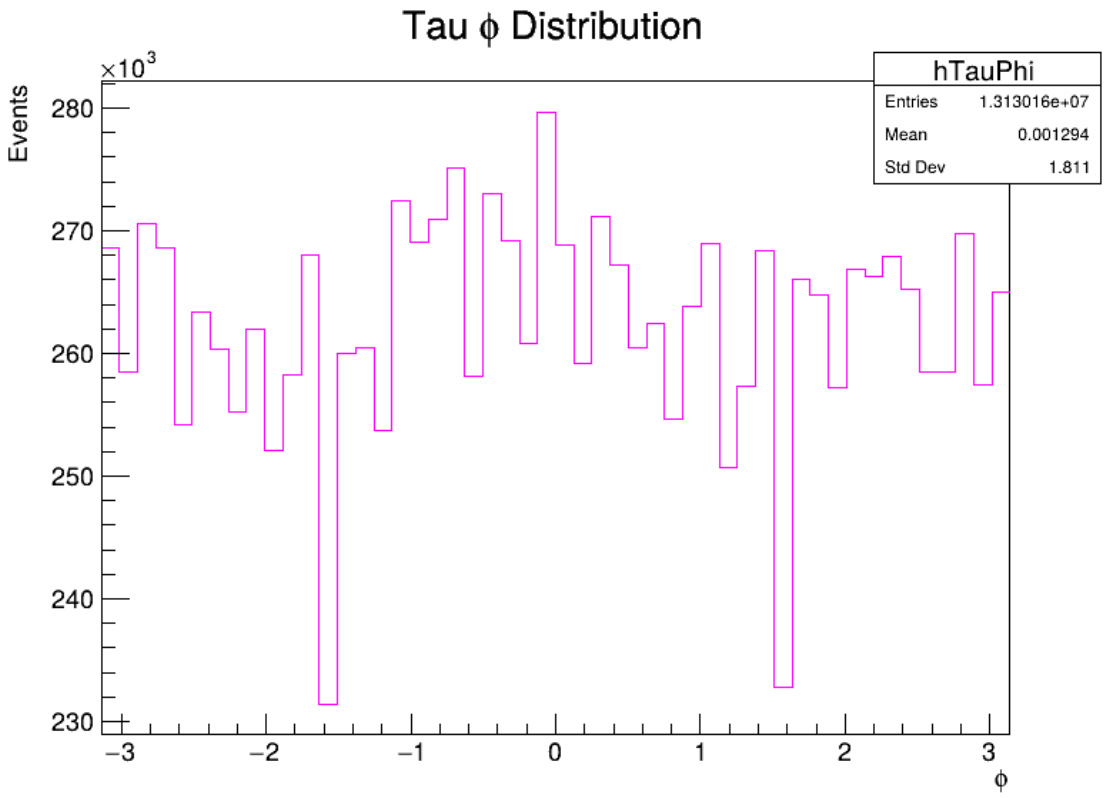

Figura 11: ϕ do Tau

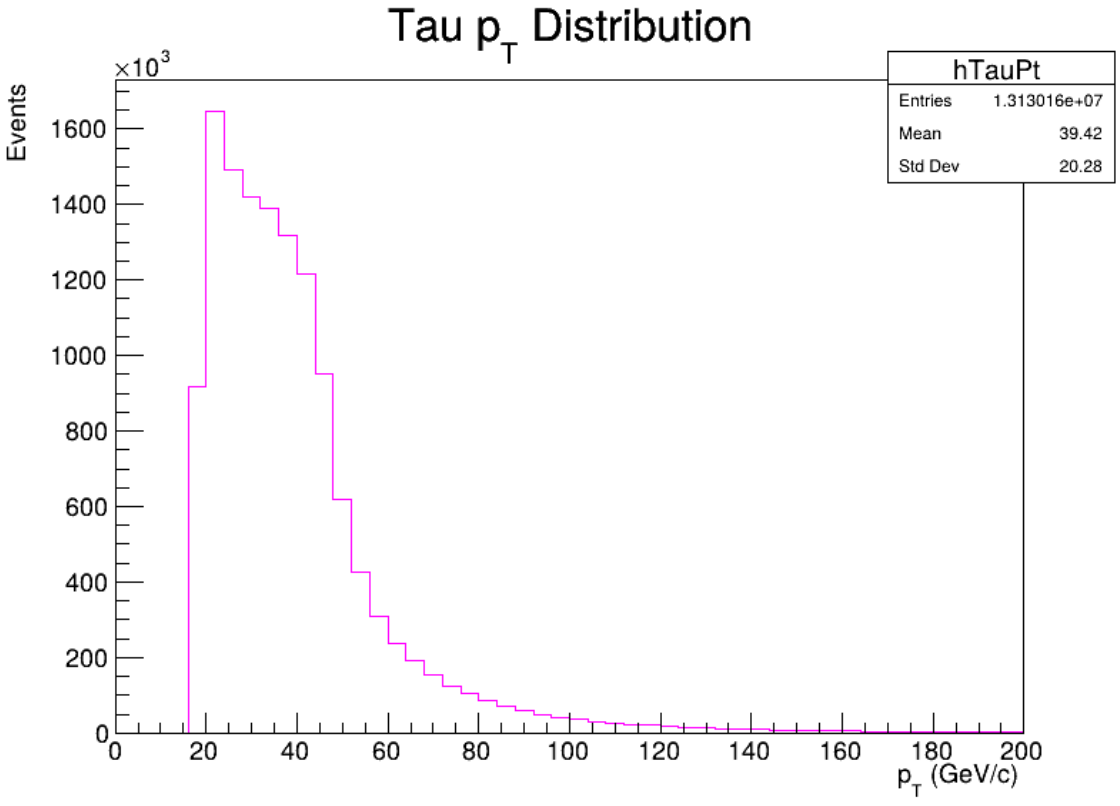

Figura 12: p_T do Tau

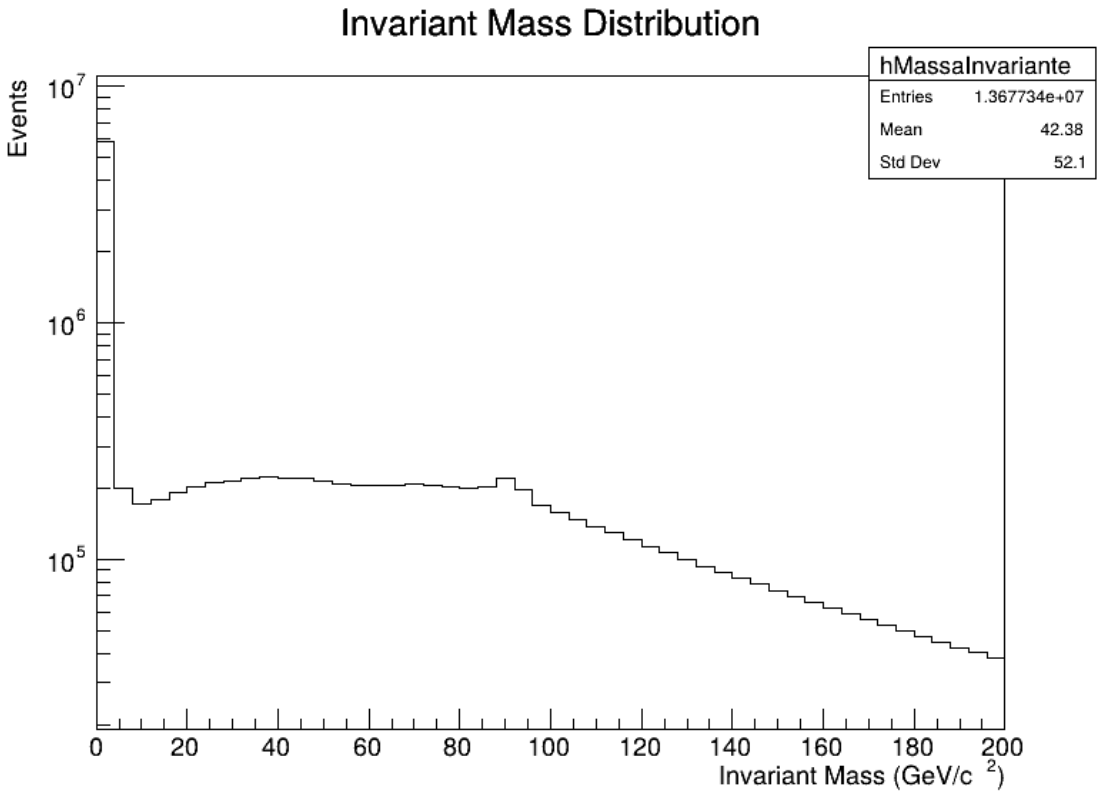

Figura 13: Massa Invariante