Introdução à Análise de dados em FAE e Tecnologias associadas **(DATA)**

CINEMATICA RELATIVISTICA

Professores: Dilson Damião.Eliza Costa e Mauricio Thiel *Name:* João Marcos Modesto Ribeiro

Solução

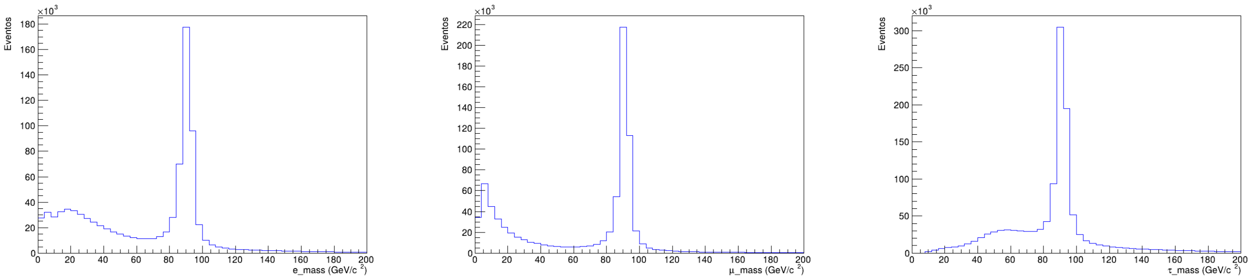
CÓDIGO:

```cpp
#include <TChain.h>
#include <TFile.h>
#include <TTreeReader.h>
#include <TTreeReaderArray.h>
#include <TTreeReaderValue.h>
#include <TCanvas.h>
#include <TH1F.h>
#include <TMath.h>
#include <iostream>
#include <vector>
#include <filesystem>
#include <TSystem.h> // Necess rio para o carregamento manual de bibliotecas

// Fun  o para calcular a massa invariante fora da fun  o cinrel
double calcular_massa_invariante(const TTreeReaderArray<float>& pt, const TTreeReaderArray<float>& eta, const TTreeReaderArray<float>& phi) {
    if (pt.GetSize() >= 2) {
        return sqrt(2 * pt[0] * pt[1] * (TMath::CosH(eta[0] - eta[1]) - TMath::Cos(phi[0] - phi[1])));
    }
    return -1.0;  // Valor inv lido caso n o haja pelo menos dois elementos
}

void cinrel() {
    // Carregar bibliotecas adicionais para resolver poss veis problemas de s mbolos n o encontrados
    gSystem->Load("libTreePlayer.so");
    gSystem->Load("libTree.so");
    gSystem->Load("libc++");      // Para sistemas baseados em libc++
    gSystem->Load("libstdc++.so"); // Para sistemas Linux com libstdc++

    std::vector<std::string> diretorios = {
        "/opendata/eos/opendata/cms/mc/RunIISummer20UL16NanoAODv9/ZZTo2Q2L_mllmin4p0_TuneCP5_13TeV-amcatnloFXFX-pythia8/NANOAODSIM/106X_mcRun2_asymptotic_v17-v1/2540000/",
    };

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

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

    // Inicializando histogramas
    TH1F* hElectronPt = new TH1F("hElectronPt", "Electron p_{T} Distribution", 50, 0, 200);
    TH1F* hElectronEta = new TH1F("hElectronEta", "Electron #eta Distribution", 50, -5, 5);
    TH1F* hElectronPhi = new TH1F("hElectronPhi", "Electron #phi Distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hMuonPt = new TH1F("hMuonPt", "Muon p_{T} Distribution", 50, 0, 200);
    TH1F* hMuonEta = new TH1F("hMuonEta", "Muon #eta Distribution", 50, -5, 5);
    TH1F* hMuonPhi = new TH1F("hMuonPhi", "Muon #phi Distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hJetPt = new TH1F("hJetPt", "Jet p_{T} Distribution", 50, 0, 200);
    TH1F* hJetEta = new TH1F("hJetEta", "Jet #eta Distribution", 50, -5, 5);
    TH1F* hJetPhi = new TH1F("hJetPhi", "Jet #phi Distribution", 50, -TMath::Pi(), TMath::Pi());

    TH1F* hTauPt = new TH1F("hTauPt", "Tau p_{T} Distribution", 50, 0, 200);
    TH1F* hTauEta = new TH1F("hTauEta", "Tau #eta Distribution", 50, -5, 5);
    TH1F* hTauPhi = new TH1F("hTauPhi", "Tau #phi Distribution", 50, -TMath::Pi(), TMath::Pi());

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

            while (reader.Next()) {
                if (Electron_pt.GetSize() >= 2) {
                    e_massas_invariantes.push_back(calcular_massa_invariante(Electron_pt, Electron_eta, Electron_phi));
                }
                if (Muon_pt.GetSize() >= 2) {
                    m_massas_invariantes.push_back(calcular_massa_invariante(Muon_pt, Muon_eta, Muon_phi));
                }
                if (Tau_pt.GetSize() >= 2) {
                    t_massas_invariantes.push_back(calcular_massa_invariante(Tau_pt, Tau_eta, Tau_phi));
                }

                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]);
                }
            }
        }
    }

    // Canvas e gr ficos
    TCanvas* canvas = new TCanvas("canvas", "Distribui  es de Massas Invariantes", 800, 600);
    TH1F* hEletron = new TH1F("hEletron", "", 50, 0, 200);
    TH1F* hMuon = new TH1F("hMuon", "", 50, 0, 200);
    TH1F* hTau = new TH1F("hTau", "", 50, 0, 200);

    for (const auto& massa : e_massas_invariantes) if (massa >= 0) hEletron->Fill(massa);
    for (const auto& massa : m_massas_invariantes) if (massa >= 0) hMuon->Fill(massa);
    for (const auto& massa : t_massas_invariantes) if (massa >= 0) hTau->Fill(massa);

    hEletron->SetLineColor(kBlue);
    hEletron->SetStats(0);
    hEletron->GetXaxis()->SetTitle("e_mass (GeV/c^{2})");
    hEletron->GetYaxis()->SetTitle("Eventos");
    hEletron->Draw();
    canvas->SaveAs("e_massa_invariante.png");

    hMuon->SetLineColor(kBlue);
    hMuon->SetStats(0);
    hMuon->GetXaxis()->SetTitle("#mu_mass (GeV/c^{2})");
    hMuon->GetYaxis()->SetTitle("Eventos");
    hMuon->Draw();
    canvas->SaveAs("m_massa_invariante.png");

    hTau->SetLineColor(kBlue);
    hTau->SetStats(0);
    hTau->GetXaxis()->SetTitle("#tau_mass (GeV/c^{2})");
    hTau->GetYaxis()->SetTitle("Eventos");
    hTau->Draw();
    canvas->SaveAs("t_massa_invariante.png");

    canvas = new TCanvas("canvasJetEta", "Jet #eta Distribution", 800, 600);
    hJetEta->SetLineColor(kGreen);
    hJetEta->Draw();
    hJetEta->GetXaxis()->SetTitle("#eta");
    hJetEta->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("jet_eta_distribution.png");

    canvas = new TCanvas("canvasJetPhi", "Jet #phi Distribution", 800, 600);
    hJetPhi->SetLineColor(kGreen);
    hJetPhi->Draw();
    hJetPhi->GetXaxis()->SetTitle("#phi");
    hJetPhi->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("jet_phi_distribution.png");

    canvas = new TCanvas("canvasJetPt", "Jet p_{T} Distribution", 800, 600);
    hJetPt->SetLineColor(kGreen);
    hJetPt->Draw();
    hJetPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
    hJetPt->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("jet_pt_distribution.png");

    // plot da Massa Invariante
    TH1F* hMassaInvariante = new TH1F("hMassaInvariante", "Invariant Mass Distribution", 50, 0, 200);
    for (const auto& massa : e_massas_invariantes) {
        if (massa >= 0) hMassaInvariante->Fill(massa);
    }

    canvas = new TCanvas("canvasInvariantMass", "Invariant Mass Distribution", 800, 600);
    hMassaInvariante->SetLineColor(kBlack);
    canvas->SetLogy();
    hMassaInvariante->Draw();
    hMassaInvariante->GetXaxis()->SetTitle("Invariant Mass (GeV/c^{2})");
    hMassaInvariante->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("invariant_mass_distribution.png");

    // Limpar recursos
    delete hJetPt;
    delete hJetEta;
    delete hJetPhi;
    delete hElectronPt;
    delete hElectronEta;
    delete hElectronPhi;
    delete hMuonPt;
    delete hMuonEta;
    delete hMuonPhi;
    delete hTauPt;
    delete hTauEta;
    delete hTauPhi;
    delete hEletron;
    delete hMuon;
    delete hTau;
    delete canvas;
}
```

Onde temos os seguintes gráficos:

Plots

1) Distribuições p_T, ϕ e η:



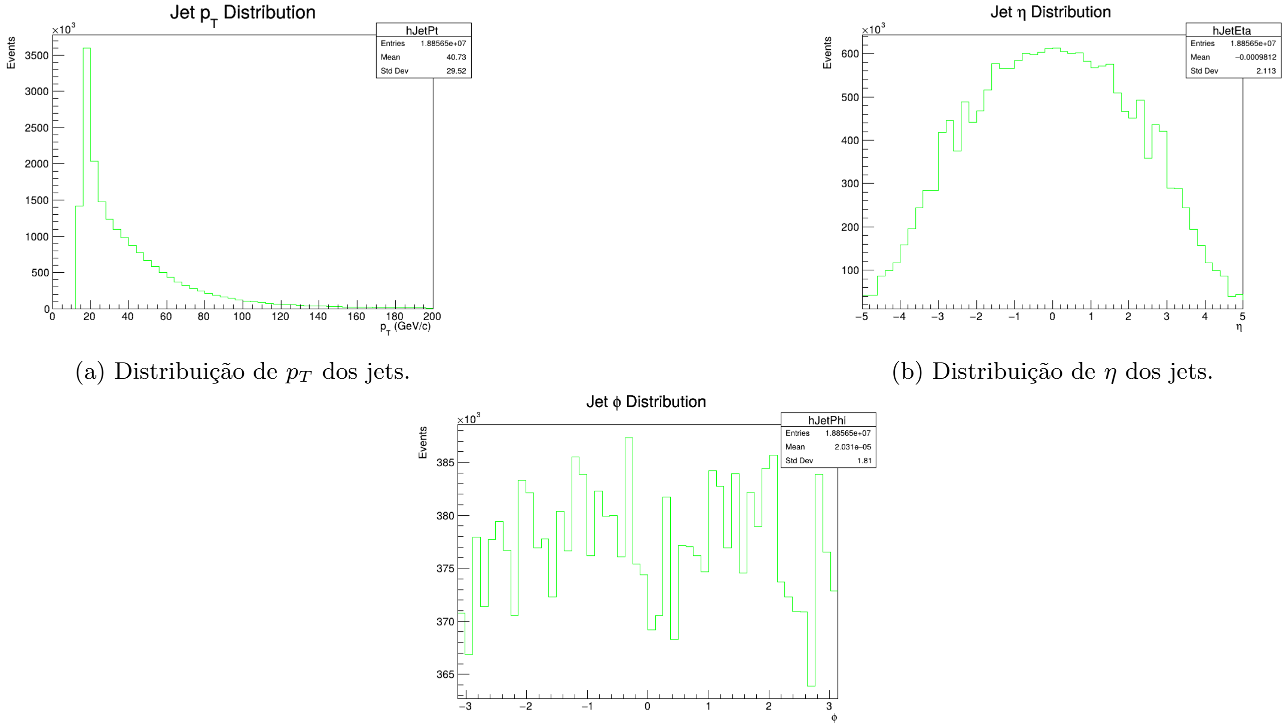
(a) Distribuição e_T.

(b) Distribuição de múons.

(c) Distribuição dos taus.

Figura 1: Distribuições de massas invariantes para e_T, múons e taus.

2) Distribuições dos jets:



(a) Distribuição de p_T dos jets.

(b) Distribuição de η dos jets.

(c) Distribuição de ϕ dos jets.

Figura 2: Distribuições dos jets: p_T, η e ϕ.

3) Distribuição de massa invariante dois léptons de maior p_t:

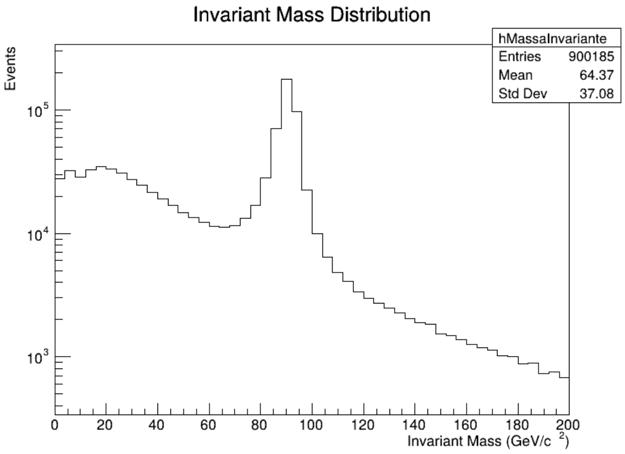


Figura 3: Distribuição da massa invariante.