Introdução à Análise de dados em FAE **(05/09/2024)**

Exercicios de estatística para análise de dados em HEP

Professores: Eliza Melo, Dilson Damião e Mauricio Thiel *Name:* Jorge Júlio Barreiros Venuto de Siqueira

Leitura dos arquivos

Para ler os arquivos disponíveis e determinar as variáveis cinemáticas da partículas, utilizou se o seguinte código em C.

Lendo o arquivo

```
void analise() {

    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());
    }

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

    // Múons
    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");
```

Distribuições de η, P_t, e ϕ dos léptons e Jatos

Definindo as variáveis cinemáticas

```
TH1F* hElectronPt = new TH1F("hElectronPt",
"Distribuicao de p_{T} dos Eletrons; p_{T} (GeV/c); Eventos", 50, 0, 200);
TH1F* hElectronEta = new TH1F("hElectronEta",
"Distribuicao de #eta dos Eletrons; #eta; Eventos", 50, -3, 3);
TH1F* hElectronPhi = new TH1F("hElectronPhi",
"Distribuicao de #phi dos Eletrons; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

TH1F* hMuonPt = new TH1F("hMuonPt",
"Distribuicao de p_{T} dos Muons; p_{T} (GeV/c); Eventos", 50, 0, 200);
TH1F* hMuonEta = new TH1F("hMuonEta",
"Distribuicao de #eta dos Muons; #eta; Eventos", 50, -3, 3);
TH1F* hMuonPhi = new TH1F("hMuonPhi",
"Distribuicao de #phi dos Muons; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

TH1F* hTauPt = new TH1F("hTauPt",
"Distribuicao de p_{T} dos Taus; p_{T} (GeV/c); Eventos", 50, 0, 200);
TH1F* hTauEta = new TH1F("hTauEta",
"Distribuicao de #eta dos Taus; #eta; Eventos", 50, -3, 3);
TH1F* hTauPhi = new TH1F("hTauPhi",
"Distribuicao de #phi dos Taus; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());

TH1F* hJetPt = new TH1F("hJetPt",
"Distribuicao de p_{T} dos Jatos; p_{T} (GeV/c); Eventos", 50, 0, 200);
TH1F* hJetEta = new TH1F("hJetEta",
"Distribuicao de #eta dos Jatos; #eta; Eventos", 50, -3, 3);
TH1F* hJetPhi = new TH1F("hJetPhi",
"Distribuicao de #phi dos Jatos; #phi; Eventos", 50, -TMath::Pi(), TMath::Pi());
int eventos_analisados = 0;

while (reader.Next()) {
    eventos_analisados++;
    if (eventos_analisados % 10000 == 0) {
        std::cout << "Eventos analisados: " << eventos_analisados << std::endl;
    }
```

Preenchendo os histogramas

```
        // Preenchendo histogramas de elétrons
        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 múons
        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]);
        }

        // Plotando histogramas de elétrons
    TCanvas *cElectronPt = new TCanvas("cElectronPt",
    "Distribuição Pt dos Elétrons", 800, 600);
        hElectronPt->Draw();
        cElectronPt->SaveAs("distribuicao_electron_pt.png");
    delete cElectronPt;

    TCanvas *cElectronEta = new TCanvas("cElectronEta",
    "Distribuição Eta dos Elétrons", 800, 600);
        hElectronEta->Draw();
        cElectronEta->SaveAs("distribuicao_electron_eta.png");
    delete cElectronEta;

    TCanvas *cElectronPhi = new TCanvas("cElectronPhi",
    "Distribuição Phi dos Elétrons", 800, 600);
        hElectronPhi->Draw();
        cElectronPhi->SaveAs("distribuicao_electron_phi.png");
    delete cElectronPhi;

}
```

Preenchendo os histogramas

```
    // Plotando histogramas de múons
    TCanvas *cMuonPt = new TCanvas("cMuonPt",
    "Distribuição Pt dos Múons", 800, 600);
        hMuonPt->Draw();
        cMuonPt->SaveAs("distribuicao_muon_pt.png");
    delete cMuonPt;

    TCanvas *cMuonEta = new TCanvas("cMuonEta",
    "Distribuição Eta dos Múons", 800, 600);
        hMuonEta->Draw();
        cMuonEta->SaveAs("distribuicao_muon_eta.png");
    delete cMuonEta;

    TCanvas *cMuonPhi = new TCanvas("cMuonPhi",
    "Distribuição Phi dos Múons", 800, 600);
        hMuonPhi->Draw();
        cMuonPhi->SaveAs("distribuicao_muon_phi.png");
    delete cMuonPhi;


    // Plotando histogramas de taús
    TCanvas *cTauPt = new TCanvas("cTauPt",
    "Distribuição Pt dos Taús", 800, 600);
        hTauPt->Draw();
        cTauPt->SaveAs("distribuicao_tau_pt.png");
    delete cTauPt;

    TCanvas *cTauEta = new TCanvas("cTauEta",
    "Distribuição Eta dos Taús", 800, 600);
        hTauEta->Draw();
        cTauEta->SaveAs("distribuicao_tau_eta.png");
    delete cTauEta;

    TCanvas *cTauPhi = new TCanvas("cTauPhi",
    "Distribuição Phi dos Taús", 800, 600);
        hTauPhi->Draw();
        cTauPhi->SaveAs("distribuicao_tau_phi.png");
    delete cTauPhi;

    // Plotando histogramas de jatos
    TCanvas *cJetPt = new TCanvas("cJetPt",
    "Distribuição Pt dos Jatos", 800, 600);
        hJetPt->Draw();
        cJetPt->SaveAs("distribuicao_jet_pt.png");
    delete cJetPt;

    TCanvas *cJetEta = new TCanvas("cJetEta",
    "Distribuição Eta dos Jatos", 800, 600);
        hJetEta->Draw();
        cJetEta->SaveAs("distribuicao_jet_eta.png");
    delete cJetEta;

    TCanvas *cJetPhi = new TCanvas("cJetPhi",
    "Distribuição Phi dos Jatos", 800, 600);
        hJetPhi->Draw();
        cJetPhi->SaveAs("distribuicao_jet_phi.png");
    delete cJetPhi;

}
```

Elétrons

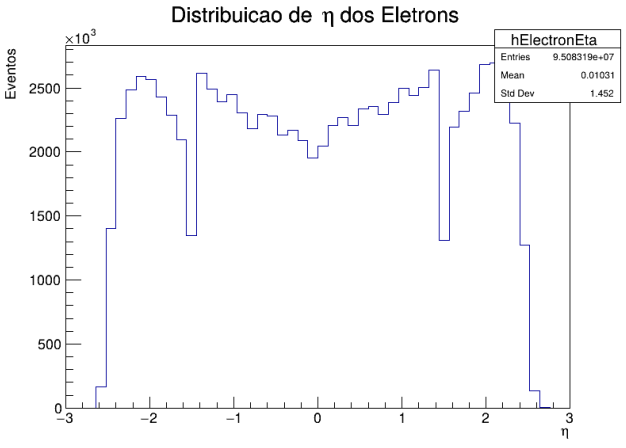

Figura 1: Distribuição do η dos elétrons

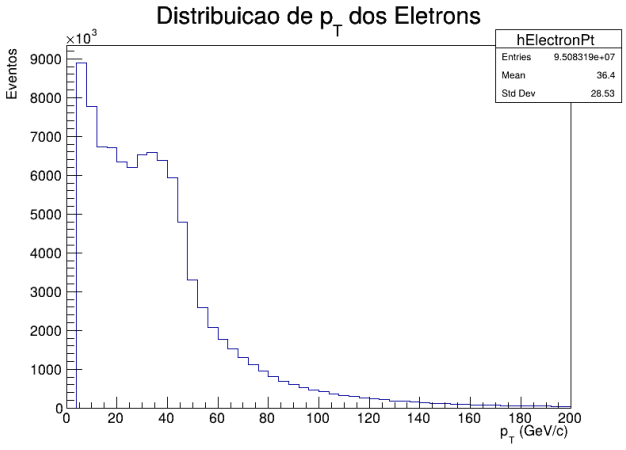

Figura 2: Distribuição do P_t dos elétrons

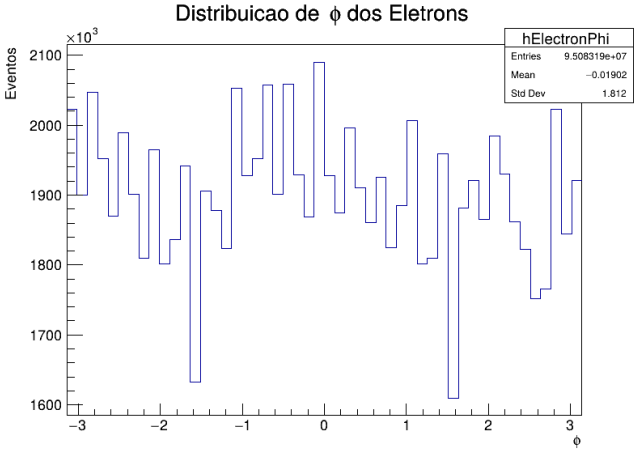

Figura 3: Distribuição do ϕ dos elétrons

Múons

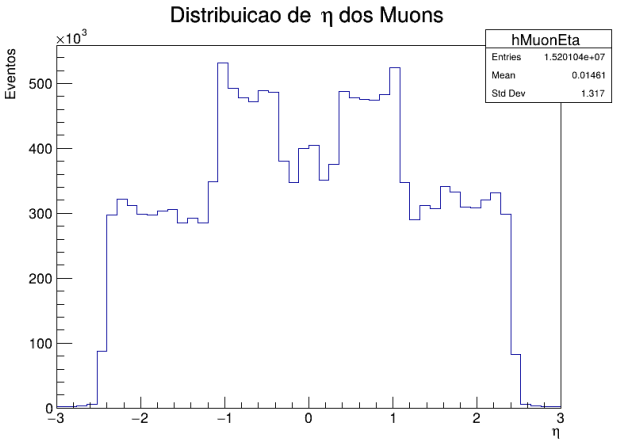

Figura 4: Distribuição do η dos múons

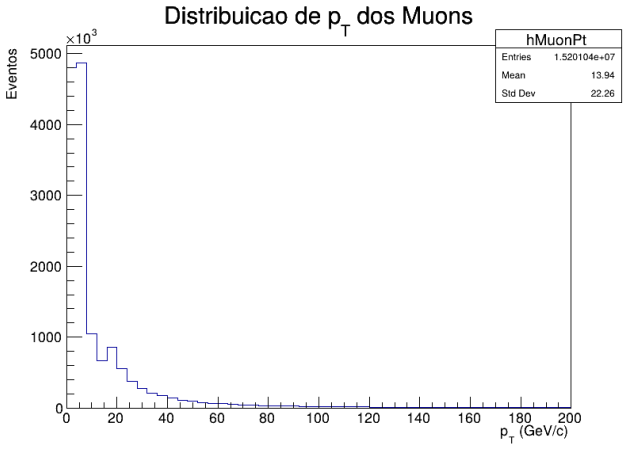

Figura 5: Distribuição do P_t dos múons

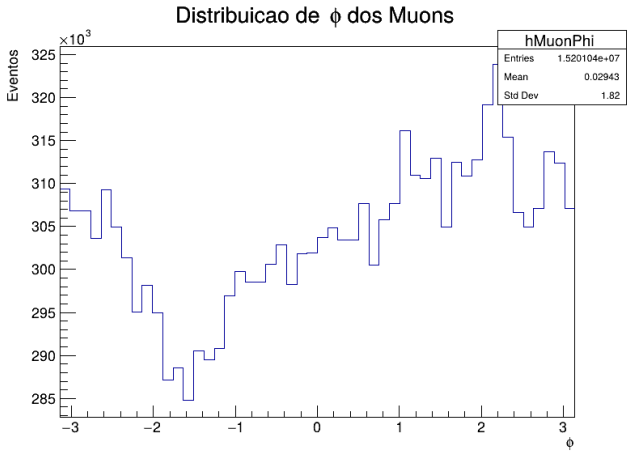

Figura 6: Distribuição do ϕ dos múons

Taus

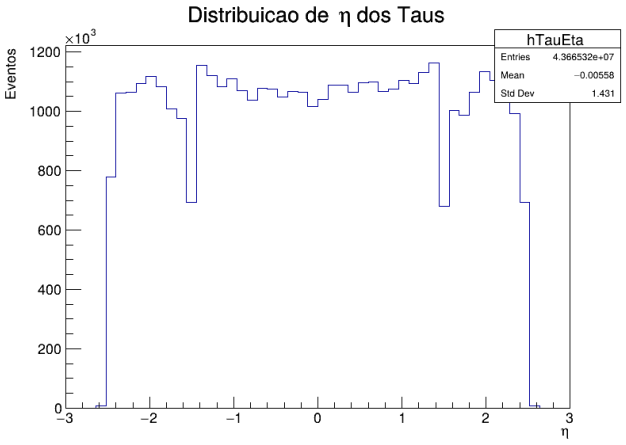

Figura 7: Distribuição do η dos taus

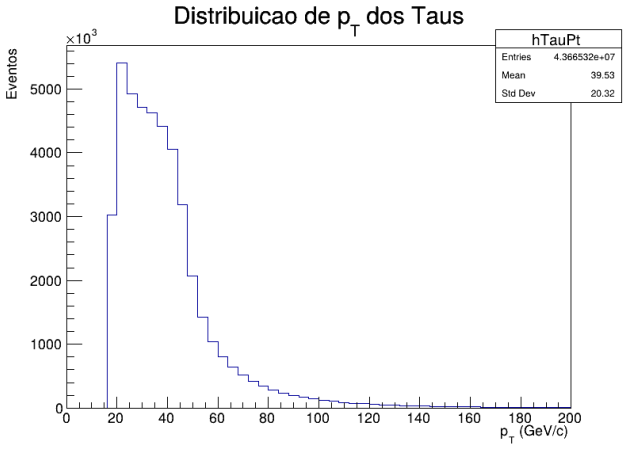

Figura 8: Distribuição do P_t dos taus

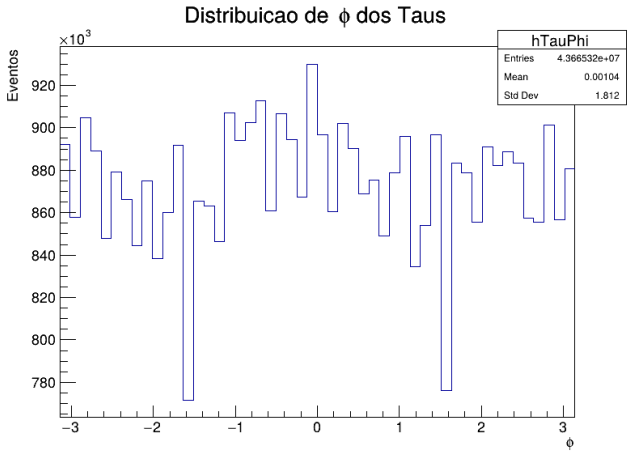

Figura 9: Distribuição do ϕ dos taus

Massa invariante dos léptons de maior P_t

Preenchendo os histogramas

```
// Criando histogramas

   TH1F* hInvariantMassMuon = new TH1F("hInvariantMassMuon",
   "Massa Invariante dos Dois Muons de Maior p_{T}; m_{ll} (GeV/c^{2});
   Eventos", 50, 0, 200);
   TH1F* hInvariantMassTau = new TH1F("hInvariantMassTau",
   "Massa Invariante dos Dois Taus de Maior p_{T}; m_{ll} (GeV/c^{2});
   Eventos", 50, 0, 200);
   TH1F* hInvariantMassElectron = new TH1F("hInvariantMassElectron",
   "Massa Invariante dos Dois Eletrons de Maior p_{T}; m_{ll} (GeV/c^{2});
   Eventos", 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;
       }
       // Analisando massa invariante de elétrons
       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 elétron
               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());
           }
       }
```

Preenchendo os histogramas

```
        // 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 múon
                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());
            }
        }
    }


    TCanvas *cInvariantMassElectron = new TCanvas("cInvariantMassElectron",
    "Massa Invariante dos Elétrons", 800, 600);
        hInvariantMassElectron->Draw();
        cInvariantMassElectron->SaveAs("distribuicao_electron_invariant_mass.png");
    delete cInvariantMassElectron;

    TCanvas *cInvariantMassMuon = new TCanvas("cInvariantMassMuon",
    "Massa Invariante dos Múons", 800, 600);
        hInvariantMassMuon->Draw();
        cInvariantMassMuon->SaveAs("distribuicao_muon_invariant_mass.png");
    delete cInvariantMassMuon;

    TCanvas *cInvariantMassTau = new TCanvas("cInvariantMassTau",
    "Massa Invariante dos Taús", 800, 600);
        hInvariantMassTau->Draw();
        cInvariantMassTau->SaveAs("distribuicao_tau_invariant_mass.png");
    delete cInvariantMassTau;
```

Resultados

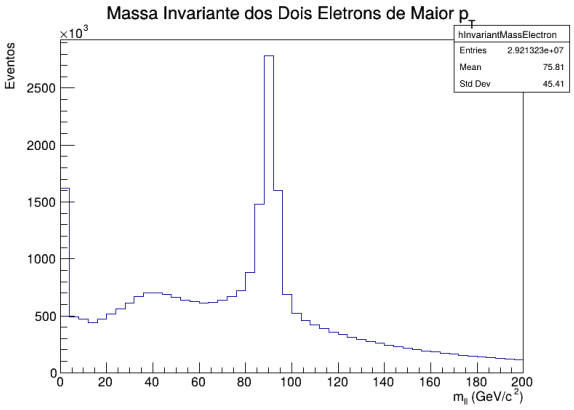

Figura 10: Distribuição de massa invariante dos elétrons de maior P_t

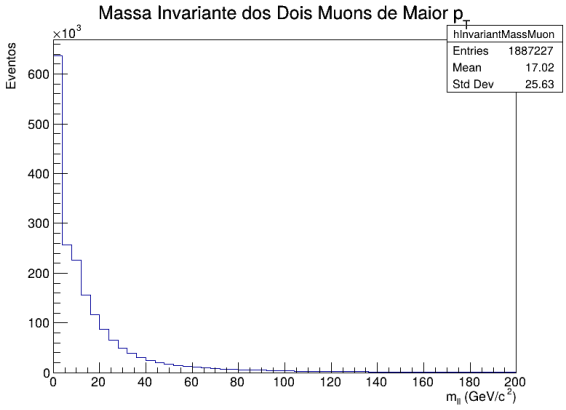

Figura 11: Distribuição de massa invariante dos múons de maior P_t

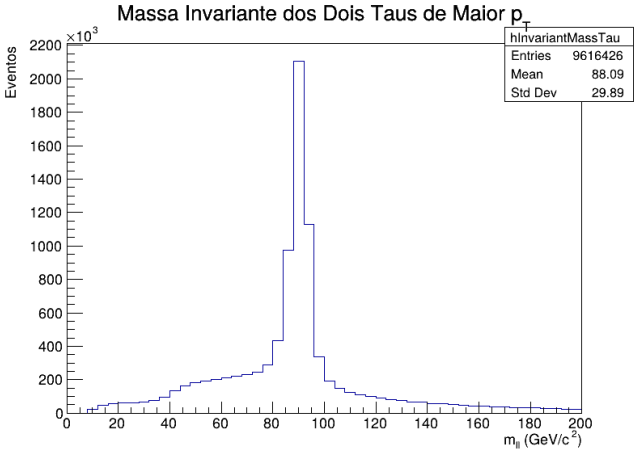

Figura 12: Distribuição de massa invariante dos taus de maior P_t