Análise de Dados em FAE (01/11/2024)

Estudos de Estatística para Análise de Dados em HEP

Professores: Eliza Melo, Dilson Damião e Mauricio Thiel

Nome: Matheus Figueiredo de Paiva Nascimento

Introdução

Neste exercício, realizaremos uma análise de dados simulados do processo `ZZ` com 13 TeV, no formato `NANOAODSIM`, utilizando o ROOT. Esse tipo de simulação representa colisões entre partículas, que produzem eventos contendo múons, elétrons, fótons e *jets*. Vamos explorar variáveis específicas, como o momento transverso dos múons, pseudorrapidez dos múons e energia transversa faltante (`MET`).

Objetivo

O objetivo é criar histogramas das variáveis de interesse para obter uma visão inicial das distribuições, essenciais para identificar padrões e preparar análises mais detalhadas.

Procedimento

Passo 1: Geração da Classe com `MakeClass`

Primeiramente, carregamos o arquivo `192691B8-1350-C741-86E5-AB514B2A5E91.root` e geramos a classe de análise com o seguinte comando:

```
TFile *file = TFile::Open("192691B8-1350-C741-86E5-AB514B2A5E91.root");
TTree *tree = (TTree*)file->Get("Events");
tree->MakeClass("MassaZZ");
```

Esse comando cria a classe `TesteML`, que permite acesso mais fácil às variáveis da árvore de eventos.

Passo 2: Implementação do Método `Loop`

O método `Loop` foi modificado para preencher os histogramas com as variáveis de interesse:

```
#define MassaZZ_cxx
#include "MassaZZ.h"
#include <TH1F.h>
#include <TCanvas.h>
#include <iostream>

void TesteML::Loop() {
    if (fChain == 0) return;

    TH1F *h_Muon_pt = new TH1F("h_Muon_pt", "Distribuicao de pT dos Muons; pT (G
    TH1F *h_Muon_eta = new TH1F("h_Muon_eta", "Distribuicao de eta dos Muons; et
    TH1F *h_MET_pt = new TH1F("h_MET_pt", "Distribuicao de MET; MET (GeV); Event

    Long64_t nentries = fChain->GetEntriesFast();
    for (Long64_t jentry = 0; jentry < nentries; jentry++) {
        Long64_t ientry = LoadTree(jentry);
        if (ientry < 0) break;
        fChain->GetEntry(jentry);

        h_MET_pt->Fill(MET_pt);
        for (size_t i = 0; i < Muon_pt->size(); i++) {
            h_Muon_pt->Fill(Muon_pt->at(i));
            h_Muon_eta->Fill(Muon_eta->at(i));
        }
    }

    TCanvas *c1 = new TCanvas("c1", "Distribuicoes", 900, 700);
    c1->Divide(2, 2);
    c1->cd(1); h_Muon_pt->Draw();
    c1->cd(2); h_Muon_eta->Draw();
    c1->cd(3); h_MET_pt->Draw();
    c1->SaveAs("distribuicoes_ZZ.png");
}
```

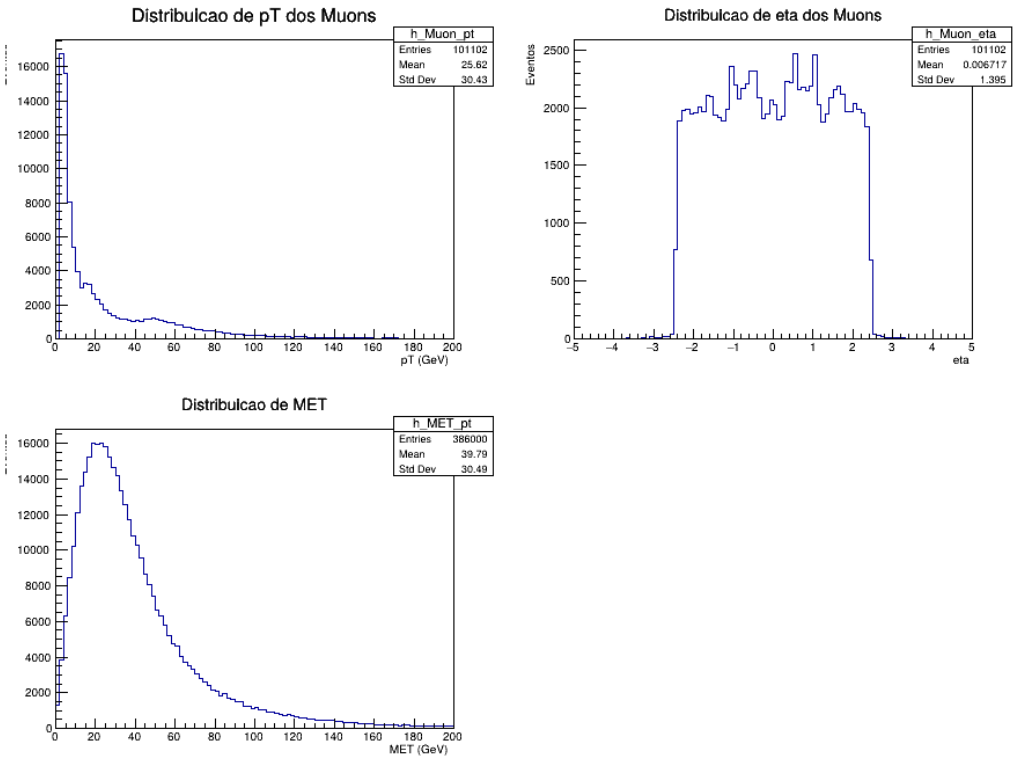

Figure 1: Distribuições geradas a partir dos dados de eventos simulados do processo `ZZ`. Os histogramas mostram (1) a distribuição do momento transverso dos múons (`pT`) em GeV, (2) a pseudorrapidez dos múons (`eta`), e (3) a energia transversa faltante (`MET`) em GeV.

Objetivo pt2

O objetivo é criar histogramas das variáveis de interesse para obter uma visão inicial das distribuições, essenciais para identificar padrões e preparar análises mais detalhadas.

Exercícios

Exercício 1: Definição e Análise de Função

Neste exercício, definimos uma função $f(x) = p_0 \cdot \frac{\sin(p_1 \cdot x)}{x}$ no ROOT, com os parâmetros $p_0 = 1$ e $p_1 = 2$, e calculamos o valor da função, sua derivada e sua integral em diferentes pontos.

```
#include <iostream>
#include <TMath.h>
#include <TF1.h>

void exercise1() {
    TF1 *func = new TF1("func", "[0] * sin([1] * x) / x", 0, 10);
    func->SetParameters(1, 2);
    func->SetLineColor(kBlue);
    func->Draw();

    double value_at_1 = func->Eval(1);
    std::cout << "Valor da função em x = 1: " << value_at_1 << std::endl;

    double derivative_at_1 = func->Derivative(1);
    std::cout << "Derivada da função em x = 1: " << derivative_at_1 << std::endl

    double integral = func->Integral(0, 3);
    std::cout << "Integral da função entre 0 e 3: " << integral << std::endl;
}
```

Resultados:

- Valor da função em $x = 1$: 0.909297
- Derivada da função em $x = 1$: -1.74159
- Integral da função entre 0 e 3: 1.42469

Exercício 2: Gráficos de Dados com e sem Erros

Neste exercício, utilizamos ROOT para carregar e plotar dados a partir de arquivos, representando-os com um gráfico de pontos conectados por uma linha. Além disso, carregamos um conjunto de dados com erros e os sobrepomos ao gráfico original.

```
void exercise2() {
    TCanvas *c10 = new TCanvas("c10", "Exercicio 02", 800, 600);

    TGraph *graph = new TGraph("graphdata.txt");
    graph->SetMarkerStyle(21);
    graph->SetMarkerColor(kBlack);
    graph->SetLineColor(kBlack);
    graph->Draw("APL");

    TGraphErrors *graphWithErrors = new TGraphErrors("graphdata_error.txt");
    graphWithErrors->SetMarkerStyle(21);
    graphWithErrors->SetMarkerColor(kBlack);
    graphWithErrors->SetLineColor(kRed);
    graphWithErrors->Draw("P_SAME");
}
```

Exercício 3: Geração de Histograma com Distribuição Gaussiana

Neste exercício, criamos um histograma para uma distribuição gaussiana de 10,000 números aleatórios com média 5 e desvio padrão 2, e configuramos o detalhamento das estatísticas exibidas.

```
void exercise3() {
    TCanvas *c2 = new TCanvas("c2", "Exercicio 3", 800, 600);

    TH1F *hist = new TH1F("hist", "Distribuicao Gaussiana;X;Entradas", 50, 0, 10
    TRandom randGen;
    for (int i = 0; i < 10000; ++i) {
        hist->Fill(randGen.Gaus(5, 2));
    }

    hist->Draw();
    hist->SetStats(1);
```

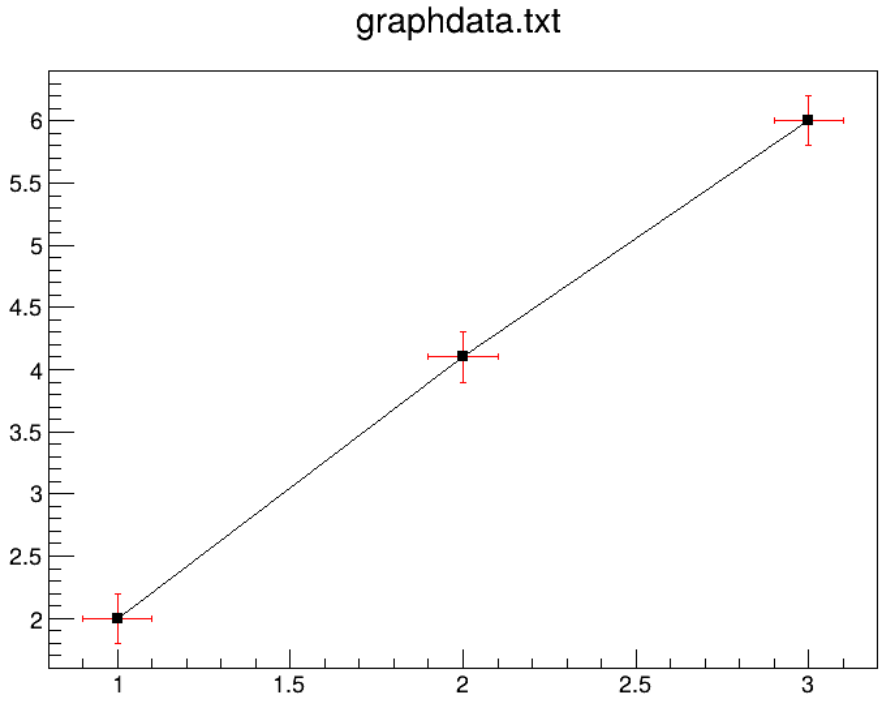

Figure 2: Dados carregados de um arquivo e desenhados em um gráfico de dispersão.

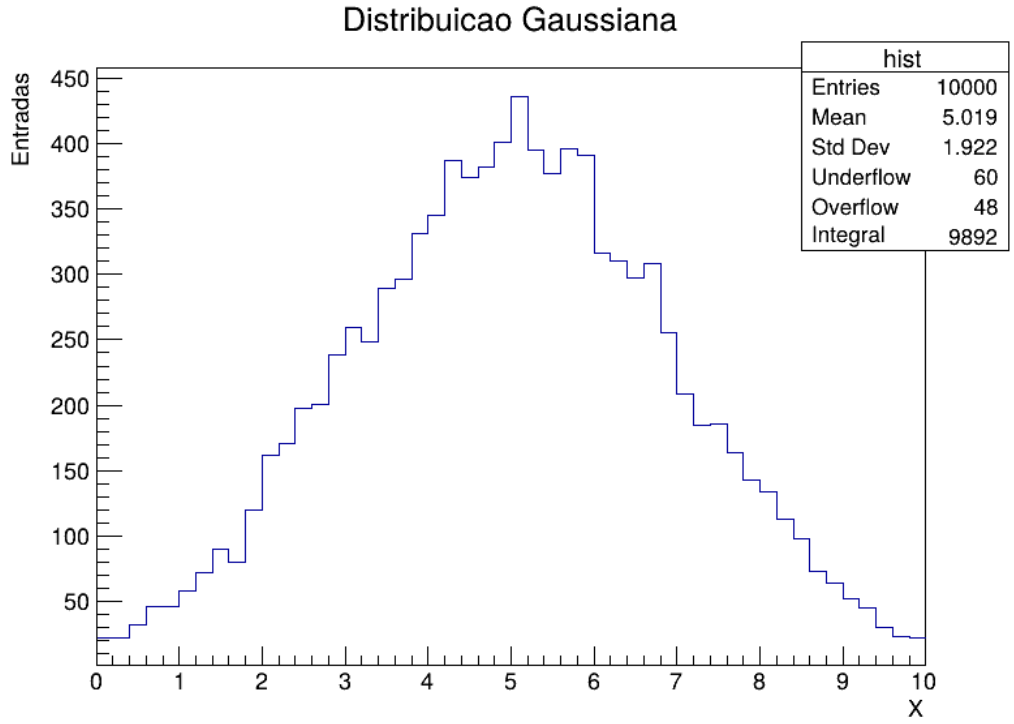

Figure 3: Distribuição Gaussiana simulada com 10.000 amostras aleatórias.

```
    gStyle->SetOptStat(1111111);
    c2->Update();
}
```

Resultado: O histograma gerado mostra a distribuição dos dados simulados, destacando as estatísticas detalhadas, como média, RMS, e contagem de entradas.

Conclusão

Os exercícios realizados permitiram explorar diferentes funcionalidades do ROOT para análise de dados simulados, como o uso de funções, gráficos e histogramas. Cada etapa envolveu o uso de recursos estatísticos que são fundamentais na análise de dados de experimentos de física de partículas.