

## EXERCÍCIOS - MANIPULANDO DADOS REAIS

*Professores:* Sandro Fonseca, Maurício Thiel, Eliza Melo*Name:* Isis Prazeres Mota

## EXERCICIO 1

- o pico escolhido foi o Z;
- criei uma função que carrega os dados, aplica cortes e cria o espectro de massa invariante de dimuons;
- criei uma função Cut() para aplicar cortes de seleção aos eventos e selecionei um pico específico (o Z);
- ajustei o pico selecionado com o RooFit e extraí os parâmetros do sinal.

```
1  #include <TFile.h>
2  #include <TTree.h>
3  #include <TH1F.h>
4  #include <TLorentzVector.h>
5  #include <TCanvas.h>
6  #include <RooFit.h>
7  #include <RooRealVar.h>
8  #include <RooDataHist.h>
9  #include <RooPlot.h>
10 #include <RooGaussian.h>
11 #include <RooChebychev.h>
12 #include <RooFitResult.h>
13 #include <RooAddPdf.h>
14 #include <iostream>
15 #include <fstream>
16
17 using namespace RooFit;
18
19 void GetRawSpectrum() {
20     // Carregar o arquivo ROOT com os dados brutos
21     TFile *file = TFile::Open("DataSkim4.root");
22     TTree *tree;
23     file->GetObject("oniaTree", tree);
24
25     // Criar um histograma para o espectro de dimuon
26     TH1F *h_dimuon_mass = new TH1F("h_dimuon_mass", "Dimuon Mass Spectrum;Mass (GeV/c
      ^2);Events", 1000, 0, 200);
27
28     // Variaveis para armazenar os dados
29     TLorentzVector *dimuon_p4 = 0;
30     tree->SetBranchAddress("dimuon_p4", &dimuon_p4);
31
32     // Loop sobre os eventos da arvore
33     Long64_t nEntries = tree->GetEntries();
34     for (Long64_t i = 0; i < nEntries; ++i) {
35         tree->GetEntry(i);
36         h_dimuon_mass->Fill(dimuon_p4->M());
37     }
38
39     // Desenhar o histograma
40     TCanvas *canvas = new TCanvas("canvas", "Dimuon Mass Spectrum", 800, 600);
```

```

41     h_dimuon_mass->Draw();
42     canvas->SaveAs("plots/dimuon_mass_spectrum.png");
43
44     // Salvar o histograma em um arquivo ROOT
45     TFile outputFile("dimuon_mass_spectrum.root", "RECREATE");
46     h_dimuon_mass->Write();
47     outputFile.Close();
48 }
49
50 void ApplySelectionCuts() {
51     // Carregar o arquivo ROOT com os dados brutos
52     TFile *file = TFile::Open("DataSkim4.root");
53     TTree *tree;
54     file->GetObject("oniaTree", tree);
55
56     // Criar um histograma para o espectro de dimuon com cortes de selecao
57     TH1F *h_dimuon_mass_cut = new TH1F("h_dimuon_mass_cut", "Dimuon Mass Spectrum
        with Cuts;Mass (GeV/c^2);Events", 1000, 85, 97);
58
59     // Variaveis para armazenar os dados
60     TLorentzVector *dimuon_p4 = 0;
61     TLorentzVector *muonP_p4 = 0;
62     TLorentzVector *muonN_p4 = 0;
63     tree->SetBranchAddress("dimuon_p4", &dimuon_p4);
64     tree->SetBranchAddress("muonP_p4", &muonP_p4);
65     tree->SetBranchAddress("muonN_p4", &muonN_p4);
66
67     // Loop sobre os eventos da arvore com cortes de selecao
68     Long64_t nEntries = tree->GetEntries();
69     for (Long64_t i = 0; i < nEntries; ++i) {
70         tree->GetEntry(i);
71         if (muonP_p4->Pt() >= 8 && fabs(muonP_p4->Eta()) < 2.4 && muonN_p4->Pt() >= 8
            && fabs(muonN_p4->Eta()) < 2.4) {
72             h_dimuon_mass_cut->Fill(dimuon_p4->M());
73         }
74     }
75
76     // Desenhar o histograma com cortes de selecao
77     TCanvas *canvas = new TCanvas("canvas_cut", "Dimuon Mass Spectrum with Selection
        Cuts", 800, 600);
78     h_dimuon_mass_cut->Draw();
79     canvas->SaveAs("plots/dimuon_mass_spectrum_cut.png");
80
81     // Salvar o histograma com cortes de selecao em um arquivo ROOT
82     TFile outputFile("dimuon_mass_spectrum_cut.root", "RECREATE");
83     h_dimuon_mass_cut->Write();
84     outputFile.Close();
85 }
86
87 void SelectPeakAndFit() {
88     // Carregar o histograma com cortes de selecao
89     TFile *file = TFile::Open("dimuon_mass_spectrum_cut.root");
90     TH1F *h_dimuon_mass_cut;
91     file->GetObject("h_dimuon_mass_cut", h_dimuon_mass_cut);
92
93     // Ajustar o pico Z com RooGaussian e RooChebychev
94     RooRealVar mass("mass", "Dimuon Mass", 85, 97);
95     RooDataHist data("data", "Dimuon Mass Data", mass, Import(*h_dimuon_mass_cut));
96     RooRealVar mean("mean", "Mean Mass", 91.0, 85.0, 97.0);
97     RooRealVar sigma("sigma", "Sigma", 2.5, 0.1, 10.0);
98     RooGaussian gauss("gauss", "Gaussian", mass, mean, sigma);
99     RooRealVar c0("c0", "c0", 0.0, -1.0, 1.0);
100    RooChebychev poly("poly", "Polynomial", mass, RooArgSet(c0));

```

```

101 RooRealVar nsig("nsig", "Signal yield", 1000, 0, 10000);
102 RooRealVar nbkg("nbkg", "Background yield", 1000, 0, 10000);
103 RooAddPdf model("model", "Signal + Background", RooArgList(gauss, poly),
    RooArgList(nsig, nbkg));
104 RooFitResult *fitResult = model.fitTo(data, Save());
105
106 // Extrair a massa do sinal e as incertezas estatisticas
107 double signalMass = mean.getVal();
108 double signalMassErr = mean.getError();
109
110 // Plotar o ajuste
111 RooPlot *frame = mass.frame();
112 data.plotOn(frame);
113 model.plotOn(frame);
114 model.plotOn(frame, Components(poly), LineStyle(kDashed));
115 model.plotOn(frame, Components(gauss), LineColor(kRed));
116
117 // Desenhar o histograma e o ajuste
118 TCanvas *canvas = new TCanvas("canvas_fit", "Dimuon Mass Spectrum Fit", 800, 600)
    ;
119 frame->Draw();
120 canvas->SaveAs("plots/dimuon_mass_spectrum_fit.png");
121
122 // Imprimir os parametros do ajuste
123 fitResult->Print("v");
124
125 // Salvar os resultados em um arquivo de texto
126 ofstream outputFile("fit_results.txt");
127 outputFile << "Signal Mass: " << signalMass << " GeV/c^2" << endl;
128 outputFile << "Signal Mass Error: " << signalMassErr << " GeV/c^2" << endl;
129 outputFile.close();
130 }
131
132 void DifferentialYield() {
133     // Carregar o arquivo ROOT com os dados brutos
134     TFile *file = TFile::Open("DataSkim4.root");
135     TTree *tree;
136     file->GetObject("oniaTree", tree);
137
138     // Criar histogramas para yield em funcao do pT
139     TH1F *h_yield_pt = new TH1F("h_yield_pt", "Yield vs pT; pT (GeV/c); Yield", 50,
        0, 100);
140
141     // Variaveis para armazenar os dados
142     TLorentzVector *dimuon_p4 = 0;
143     TLorentzVector *muonP_p4 = 0;
144     TLorentzVector *muonN_p4 = 0;
145     tree->SetBranchAddress("dimuon_p4", &dimuon_p4);
146     tree->SetBranchAddress("muonP_p4", &muonP_p4);
147     tree->SetBranchAddress("muonN_p4", &muonN_p4);
148
149     // Loop sobre os eventos da arvore com cortes de selecao
150     Long64_t nEntries = tree->GetEntries();
151     for (Long64_t i = 0; i < nEntries; ++i) {
152         tree->GetEntry(i);
153         if (muonP_p4->Pt() >= 8 && fabs(muonP_p4->Eta()) < 2.4 && muonN_p4->Pt() >= 8
            && fabs(muonN_p4->Eta()) < 2.4) {
154             h_yield_pt->Fill(dimuon_p4->Pt());
155         }
156     }
157
158     // Desenhar o histograma de yield em funcao do pT
159     TCanvas *canvas = new TCanvas("canvas_yield_pt", "Yield vs pT", 800, 600);

```

```

160 h_yield_pt->Draw();
161 canvas->SaveAs("plots/yield_vs_pt.png");
162
163 // Salvar o histograma de yield em um arquivo ROOT
164 TFile outputFile("yield_vs_pt.root", "RECREATE");
165 h_yield_pt->Write();
166 outputFile.Close();
167 }
168
169 void RunAnalysis() {
170     GetRawSpectrum();
171     ApplySelectionCuts();
172     SelectPeakAndFit();
173     DifferentialYield();
174 }

```



