Introdução à Análise de Dados em FAE
(20/10/2024)

Exercícios de RooFit (aula 4)

Professores:
Eliza Melo, Dilson Damião e Mauricio Thiel

Nome:
Matheus da Costa Geraldes

1 Aviso

os códigos, as imagens e assim como os arquivos usados se encontram no meu Git: `https://github.com/Geraldes-Matheus/cruso-analise-de-dados-2024-2/tree/main/exerc%C3%ADcio4/Data`

Exercício 1

No exercício 1, foi criado um modelo de probabilidade usando a função Crystal Ball.

Uma variável observável chamada x foi criada, com limites de -10 a 10. Os parâmetros da função Crystal Ball foram definidos da seguinte forma:

- **mean (Média)**: inicializada em 0.
- **sigma (Desvio Padrão)**: inicializada em 1.
- **alpha**: inicializada em 1.
- **n**: inicializada em 1.

Uma amostra de dados foi gerada a partir da PDF Crystal Ball, consistindo em 1000 eventos.

Foi utilizado o método `fitTo()` para ajustar a função Crystal Ball aos dados gerados. O resultado do ajuste foi visualizado em um gráfico.

Uma caixa de informação estatística foi adicionada ao gráfico, exibindo os parâmetros ajustados:

- **Mean**: valor ajustado da média.
- **Sigma**: valor ajustado do desvio padrão.
- **Alpha**: valor ajustado de alpha.
- **n**: valor ajustado de n.

O gráfico resultante foi salvo no arquivo `ex1.png`.

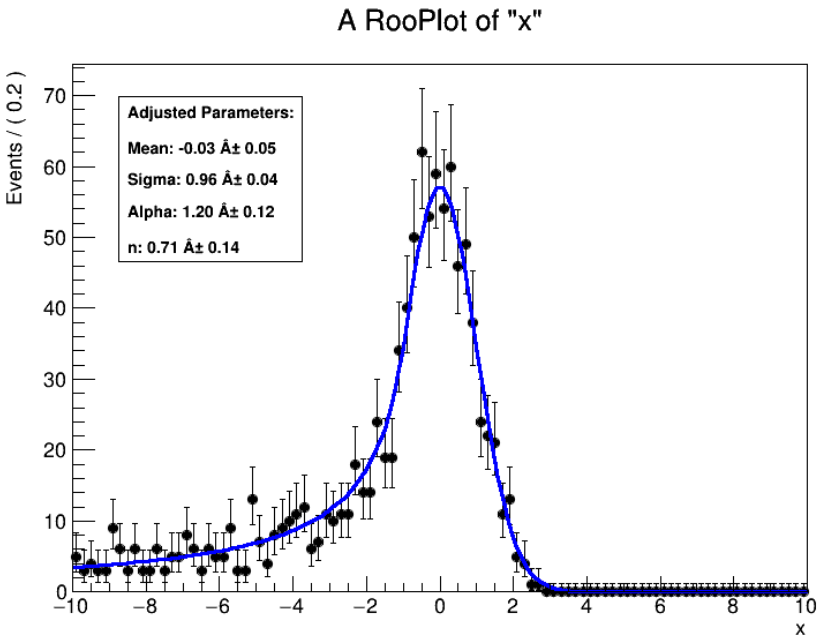

Figure 1: Exercício 1

Exercício 2

Neste exercício, foi realizado um ajuste de uma função exponencial decrescente usando a biblioteca RooFit. Os resultados obtidos são os seguintes:

- **Valor ajustado para o parâmetro lambda:**

$$\lambda \approx 0.9727 \pm 0.0252$$

- **Número total de eventos ajustados:**

$$\text{Total de Eventos} = 1500$$

Comparando os valores ajustados com os valores gerados, temos:

- O valor ajustado para lambda ($\lambda \approx 0.9727$) está próximo do valor gerado ($\lambda = 1$), o que é esperado. O ajuste deve convergir para um valor próximo do verdadeiro parâmetro gerado, especialmente em um número significativo de eventos.

- O número total de eventos ajustados (1500) corresponde exatamente ao número de eventos gerados, o que também é esperado, pois o ajuste estendido considera o número total de eventos durante o ajuste.

O gráfico resultante foi salvo no arquivo `ex2.png`.

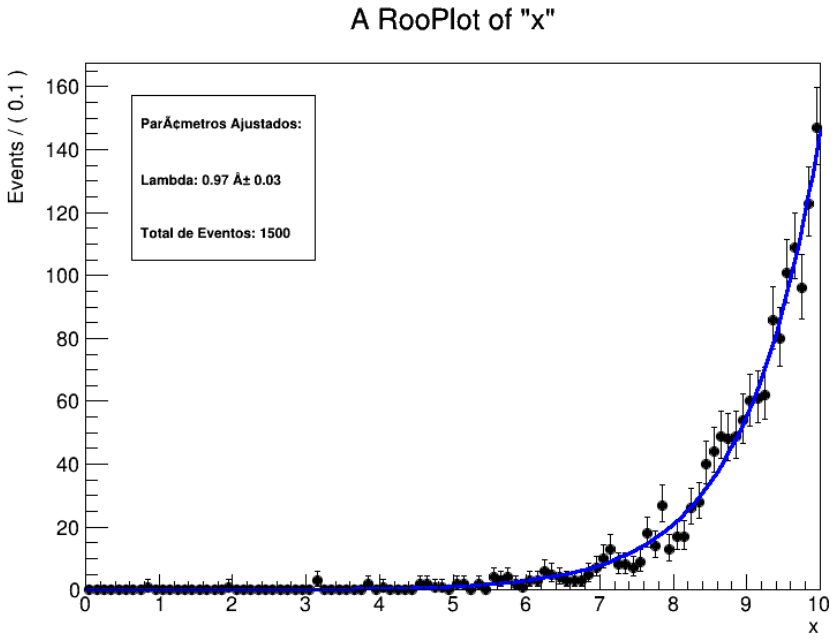

Figure 2: Exercício 2

Exercício 3

No exercício, foi construído um modelo para a distribuição de massa da ressonância J/ψ utilizando a biblioteca RooFit.

Foi utilizado o arquivo `DataSet_lowstat.root` para acessar os dados de massa.

- Foi criada uma variável chamada `mass`, que representa a massa em GeV/c^2, com limites ajustados de 2.5 a 3.5.
- A função de sinal foi definida usando a função Crystal Ball, configurando seus parâmetros (média, desvio padrão, alpha e n) para capturar o pico da ressonância em 3.096916 GeV/c^2.
- Para o fundo, foi utilizada uma função polinomial de segundo grau com três coeficientes.

As PDFs do sinal e do fundo foram combinadas em um modelo único.

Foi utilizado o método `fitTo()` para ajustar o modelo aos dados.

Após o ajuste, foram calculados χ^2 e os graus de liberdade (ndf). O número de dados foi 500 e o número de parâmetros ajustados foi 8, resultando em 492 graus de liberdade.

- $\chi^2 = 1.37869$
- $ndf = 492$
- $\chi^2/ndf = 0.00280222$

O gráfico resultante foi salvo no arquivo `plot_jpsi.png`.

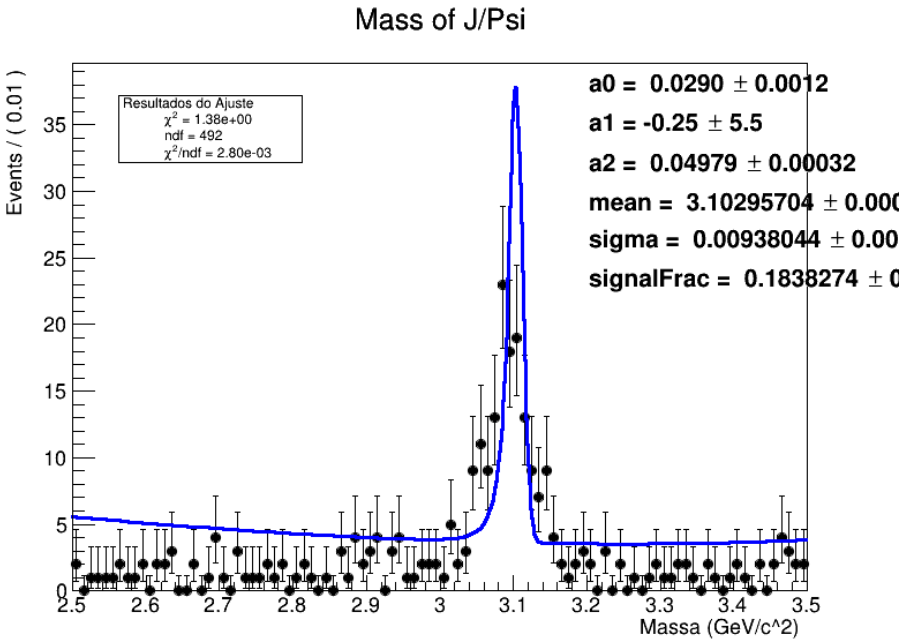

Figure 3: Exercício 3

`Interpretação dos Resultados:`

O valor de χ^2/ndf é muito baixo, indicando que o modelo se ajustou bem aos dados, com pouca discrepância entre as observações e o que o modelo previu. No entanto, um valor tão baixo pode sugerir que as incertezas nos dados foram subestimadas ou que o modelo é mais complexo do que o necessário.

Na caixa de estatísticas, os seguintes parâmetros foram destacados:

- **Mean (Média)**: Representa o valor médio da distribuição do sinal, que é 3.096916 GeV/c², o pico da ressonância J/ψ.

- **Sigma (Desvio Padrão)**: Indica a largura da distribuição em torno da média; valores menores sugerem uma distribuição mais concentrada.
- **SignalFrac (Frações do Sinal)**: Refere-se à proporção do sinal em relação ao fundo; um valor de 0.1 sugere que o sinal é uma pequena fração da distribuição total.

A Códigos dos exercícios em C

A.1 Código 1: Exercícios 1

Código do Exercícios 1

```
void ex1() {
    // Criar uma vari vel observ vel
    RooRealVar x("x", "x", -10, 10);

    // Par metros da Crystal Ball
    RooRealVar mean("mean", "mean", 0, -10, 10);
    RooRealVar sigma("sigma", "sigma", 1, 0.1, 10);
    RooRealVar alpha("alpha", "alpha", 1, 0, 5);
    RooRealVar n("n", "n", 1, 0, 10);

    // Criar a fun   o Crystal Ball
    RooCrystalBall cb("cb", "Crystal Ball PDF", x, mean, sigma, alpha
        , n);

    // Gerar dados simulados
    RooDataSet* data = cb.generate(RooArgSet(x), 1000);

    // Criar um canvas para plotar
    TCanvas* canvas = new TCanvas("canvas", "Canvas", 800, 600);

    // Criar um RooPlot para a vari vel x
    RooPlot* frame = x.frame();
    data->plotOn(frame);
    cb.plotOn(frame);

    // Ajuste da fun   o aos dados
    cb.fitTo(*data);

    // Adicionar caixa de informa   o estat stica
    TPaveText* statsBox = new TPaveText(0.15, 0.6, 0.35, 0.85, "NDC")
        ;
    statsBox->SetBorderSize(1);
    statsBox->SetFillColor(0);
    statsBox->SetTextAlign(12);
    statsBox->AddText("Adjusted Parameters:");
    statsBox->AddText(Form("Mean: %.2f    %.2f", mean.getVal(), mean.
        getError()));
    statsBox->AddText(Form("Sigma: %.2f    %.2f", sigma.getVal(),
        sigma.getError()));
    statsBox->AddText(Form("Alpha: %.2f    %.2f", alpha.getVal(),
        alpha.getError()));
    statsBox->AddText(Form("n: %.2f    %.2f", n.getVal(), n.getError
        ()));

    // Adicionar a caixa ao frame
    frame->addObject(statsBox);

    // Mostrar o resultado
    frame->Draw();
    canvas->SaveAs("ex1.png");

    // Limpar
    delete canvas;
    delete data;
}
```

A.2 Código 2: Exercícios 2

Código do Exercícios 2

```
#include "RooFit.h"
#include "RooRealVar.h"
#include "RooExponential.h"
#include "RooDataSet.h"
#include "RooPlot.h"
#include "TCanvas.h"
#include "TPaveText.h"

using namespace RooFit;

void ex2() {
    //  vari vel cont nua x
    RooRealVar x("x", "x", 0, 10);

    //  par metro lambda
    RooRealVar lambda("lambda", "Taxa de Decaimento", 1, 0.1, 2);

    //  fun  o exponencial decrescente
    RooExponential expFunc("expFunc", "Fun  o Exponencial", x,
        lambda);

    // Gerar dados simulados
    RooDataSet* data = expFunc.generate(RooArgSet(x), 1500); // 1500
        eventos

    //  canvas para plotar
    TCanvas* canvas = new TCanvas("canvas", "Ajuste Exponencial",
        800, 600);

    //  RooPlot para a vari vel x
    RooPlot* frame = x.frame();
    data->plotOn(frame);

    // Ajuste da fun  o exponencial
    expFunc.fitTo(*data);

    //  fun  o ajustada
    expFunc.plotOn(frame);

    //  caixa de informa  o estat stica
    TPaveText* statsBox = new TPaveText(0.15, 0.6, 0.35, 0.85, "NDC")
        ;
    statsBox->SetBorderSize(1);
    statsBox->SetFillColor(0);
    statsBox->SetTextAlign(12); // Alinhamento do texto
    statsBox->AddText("Par metros Ajustados:");
    statsBox->AddText(Form("Lambda: %.2f  %.2f", lambda.getVal(),
        lambda.getError()));
    statsBox->AddText(Form("Total de Eventos: %d", (int)data->
        numEntries()));

    // Adicionar a caixa ao frame
    frame->addObject(statsBox);


    frame->Draw();
    canvas->SaveAs("ex2.png");


    delete canvas;
    delete data;
}
```

A.3 Código 3: Exercícios 3

```
#include <RooGlobalFunc.h>
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooPlot.h>
#include <RooCrystalBall.h>
#include <RooPolynomial.h>
#include <RooAddPdf.h>
#include <TCanvas.h>
#include <TFile.h>
#include <TLegend.h>
#include <iostream>


void ex3() {
    // Carregar o arquivo ROOT
    TFile *file = TFile::Open("DataSet_lowstat.root");
    RooDataSet *data = (RooDataSet*)file->Get("data");

    // Definir a vari vel de massa
    RooRealVar mass("mass", "Massa (GeV/c^2)", 2.5, 3.5);  // Ajuste os
        limites conforme necess rio

    // Definir a PDF do sinal (Crystal Ball)
    RooRealVar mean("mean", "M dia", 3.096916, 3.0, 3.2);
    RooRealVar sigma("sigma", "Desvio Padr o", 0.01, 0.001, 0.1);
    RooRealVar alpha("alpha", "Alpha", 1.0);
    RooRealVar n("n", "n", 5.0);
    RooCrystalBall signal("signal", "Sinal J/  ", mass, mean, sigma, alpha, n
        );

    // Definir a PDF do fundo
    RooRealVar a0("a0", "a0", 1.0, -10.0, 10.0);
    RooRealVar a1("a1", "a1", 0.0, -10.0, 10.0);
    RooRealVar a2("a2", "a2", 0.0, -10.0, 10.0);
    RooPolynomial background("background", "Fundo", mass, RooArgList(a0, a1,
        a2));

    // Combinar as PDFs
    RooRealVar signalFrac("signalFrac", "Fra  es do Sinal", 0.1, 0.0, 1.0);
    RooAddPdf model("model", "Modelo Sinal + Fundo", RooArgList(signal,
        background), RooArgList(signalFrac));

    // Ajustar o modelo aos dados
    model.fitTo(*data);

    // Calcular     /ndf
    RooPlot *frame = mass.frame();
    data->plotOn(frame);
    model.plotOn(frame);
    model.paramOn(frame);

    // Exibir o resultado do ajuste
    TCanvas *c1 = new TCanvas("c1", "Ajuste J/  ", 800, 600);
    frame->SetTitle("Mass of J/Psi");
    frame->Draw();

    // Calcular      e ndf
    double chi2 = frame->chiSquare();
    int ndata = data->numEntries(); // N mero de eventos no dataset
    int nparams = model.getParameters(mass)->getSize(); // N mero de
        par metros ajustados
    int ndf = ndata - nparams; // Calculando ndf

    // Calcular chi /ndf
    double chi2_ndf = chi2 / ndf;

```

```
62      // Adicionar os valores de     , ndf e     /ndf ao gr fico usando
            TLegend
63      TLegend *legend = new TLegend(0.15, 0.75, 0.35, 0.85); // Coordenadas
            ajustadas para a esquerda
64      legend->AddEntry((TObject*)0, Form("#chi^{2}␣=␣%.2e", chi2), "");
65      legend->AddEntry((TObject*)0, Form("ndf␣=␣%d", ndf), "");
66      legend->AddEntry((TObject*)0, Form("#chi^{2}/ndf␣=␣%.2e", chi2_ndf), "");
67      legend->SetHeader("Resultados␣do␣Ajuste");
68      legend->SetBorderSize(1);
69      legend->SetFillColor(kWhite); // Cor de fundo branca
70      legend->Draw();
71
72      std::cout << "N mero␣de␣pontos␣de␣dados␣(ndata):␣" << ndata << std::endl
            ;
73      std::cout << "N mero␣de␣par metros␣ajustados␣(nparams):␣" << nparams <<
             std::endl;
74
75
76      std::cout << "    ␣=␣" << chi2 << std::endl;
77      std::cout << "ndf␣=␣" << ndf << std::endl;
78      std::cout << "chi^2␣/␣ndf␣=␣" << chi2_ndf << std::endl;
79
80      // Salvar o gr fico em formato PNG
81      c1->SaveAs("plot_jpsi.png");
82
83      // Fechar o arquivo
84      file->Close();
85  }
```