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

Exercício 1

Enunciado: Crie uma p.d.f Crystal Ball, gere uma amostra de dados a partir dessa pdf e ajuste aos dados. Adicione a caixa de informação estatística dos dados e do modelo. Quais foram os valores ajustados para os parâmetros dessa pdf?

Resolução

Neste exercício, utilizaremos o pacote `RooFit` do `ROOT` para criar uma função de densidade de probabilidade (p.d.f.) do tipo Crystal Ball, gerar uma amostra de dados a partir dessa p.d.f., realizar o ajuste e exibir as informações estatísticas dos parâmetros ajustados.

Definição da p.d.f. Crystal Ball

A função Crystal Ball é uma p.d.f. que combina uma distribuição Gaussiana com uma cauda assimétrica. Ela é amplamente usada em física de partículas para modelar distribuições com caudas longas. A fórmula geral da função Crystal Ball é dada por:

$$f(x;\mu,\sigma,\alpha,n) = \begin{cases} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), & \text{se } \frac{x-\mu}{\sigma} > -\alpha \\ A\left(B - \frac{x-\mu}{\sigma}\right)^{-n}, & \text{se } \frac{x-\mu}{\sigma} \leq -\alpha \end{cases}$$

onde: - μ é a média da distribuição, - σ é o desvio padrão, - α controla a posição da cauda, - n é um parâmetro que define o comportamento da cauda.

Código em C++ para Implementação

Para realizar o exercício, criamos um código em `C++` que faz uso das classes do `RooFit` para definir e ajustar a função Crystal Ball.

```
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooPlot.h>
#include <RooCBShape.h>
#include <RooFit.h>
#include <TCanvas.h>
#include <TPaveStats.h>
#include <iostream>

using namespace RooFit;

void ExercicioCrystalBall() {

    RooRealVar x("x", "x", -10, 10);

    RooRealVar mean("mean", "Média", 0, -10, 10);
    RooRealVar sigma("sigma", "Desvio padrão", 1, 0.1, 5);
    RooRealVar alpha("alpha", "Cauda", 1, 0.1, 10);
    RooRealVar n("n", "Expoente da cauda", 2, 0.1, 10);

    RooDataSet* data = crystalBall.generate(x, 10000); // Amostra com 10.000 pontos

    crystalBall.fitTo(*data);

    RooPlot* xframe = x.frame(Title("Ajuste da p.d.f. Crystal Ball"));
    data->plotOn(xframe);
    crystalBall.plotOn(xframe);

    TCanvas* c1 = new TCanvas("c1", "Exercicio Crystal Ball", 800, 600);
    xframe->Draw();

    TPaveStats *stats = (TPaveStats*)xframe->findObject("stats");
    if (stats) {
        stats->SetX1NDC(0.7); // posição da caixa (ajustável)
        stats->SetY1NDC(0.7); // posição da caixa (ajustável)
    }
    c1->Update();

    c1->SaveAs("ajuste_crystal_ball.png");

    std::cout << "Valores ajustados para os parâmetros:" << std::endl;
    std::cout << "Média (mean): " << mean.getVal() << " ± " << mean.getError() << std::endl;
```

```
    std::cout << "Desvio padrão (sigma): " << sigma.getVal() << " ± " << sigma.getError() <<
    std::cout << "Cauda (alpha): " << alpha.getVal() << " ± " << alpha.getError() << std::en
    std::cout << "Expoente da cauda (n): " << n.getVal() << " ± " << n.getError() << std::en
}
```

Explicação do Código

- Definimos a variável observável x no intervalo $[-10, 10]$, que representa a variável aleatória da nossa p.d.f.
- Definimos os parâmetros da Crystal Ball: média (`mean`), desvio padrão (`sigma`), cauda (`alpha`) e expoente da cauda (`n`).
- Criamos a p.d.f. Crystal Ball com a classe `RooCBShape`.
- Geramos uma amostra de dados com 10.000 pontos usando a função de densidade Crystal Ball.
- Realizamos o ajuste dos dados gerados com o método `fitTo`.
- Exibimos os dados e a função ajustada em um gráfico, adicionando uma caixa de informação estatística com `TPaveStats`.

Resultados Esperados

O código produz os seguintes resultados:

- Um gráfico, salvo como `ajuste_crystal_ball.png`, com a distribuição dos dados e o ajuste da função Crystal Ball.
- No console, são exibidos os valores ajustados dos parâmetros com seus respectivos erros:

Exercício 2

Enunciado: Ajuste uma função exponencial para um conjunto de dados e analise os resultados.

Resolução

Neste exercício, utilizamos o `ROOT` para ajustar uma função exponencial a um conjunto de dados. O ajuste exponencial foi realizado utilizando o método dos mínimos quadrados.

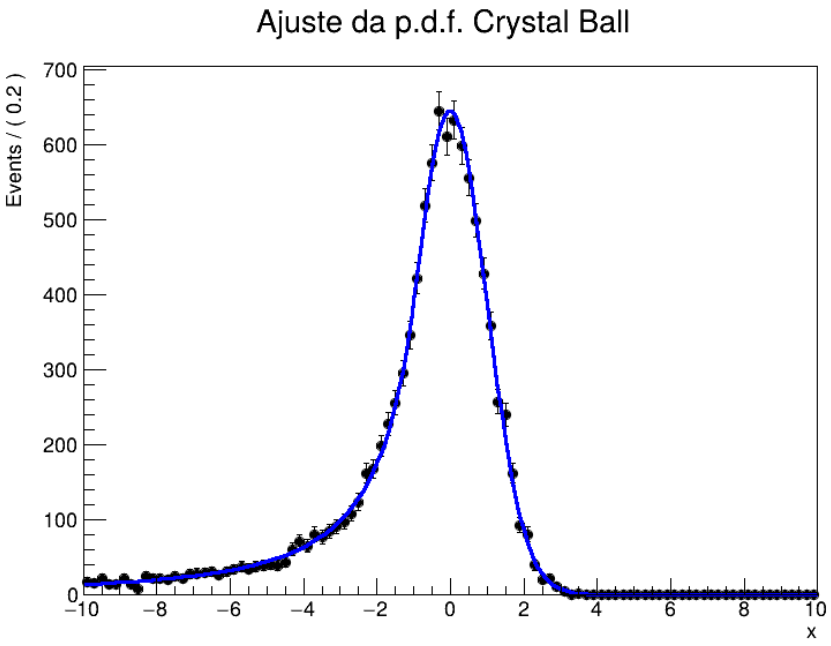

Figure 1: Gráfico Cristal Ball

```
Valores ajustados para os parâmetros:
Média (mean): -0.00512439 ± 0.0183233
Desvio padrão (sigma): 0.997925 ± 0.0135816
Cauda (alpha): 0.996408 ± 0.0419262
Expoente da cauda (n): 1.98769 ± 0.15415
```

Figure 2: Valores dos Parâmetros

Código em C++ para Implementação

```
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooExponential.h>
#include <RooFit.h>
#include <RooPlot.h>
#include <TCanvas.h>
#include <iostream>

using namespace RooFit;

void ExercicioExponencial() {

    RooRealVar x("x", "x", 0, 10);

    RooRealVar lambda("lambda", "Lambda", -1, -10, 0);

    RooExponential expo("expo", "Exponencial", x, lambda);

    RooDataSet* data = expo.generate(x, 1500);

    expo.fitTo(*data);

    RooPlot* xframe = x.frame(Title("Ajuste da função Exponencial"));
    data->plotOn(xframe);
    expo.plotOn(xframe);

    TCanvas* c1 = new TCanvas("c1", "Exercicio Exponencial", 800, 600);
    xframe->Draw();
    c1->Update();

    c1->SaveAs("ajuste_exponencial.png");

    std::cout << "Lambda ajustado: " << lambda.getVal() << " ± " << lambda.getError() << std
}
```

Resultados do Ajuste

Os resultados obtidos para o ajuste exponencial foram:

- **Lambda ajustado:** -0.971077 $\pm$ 0.0251421
- **Total de eventos ajustados:** 1500 $\pm$ 38.7256

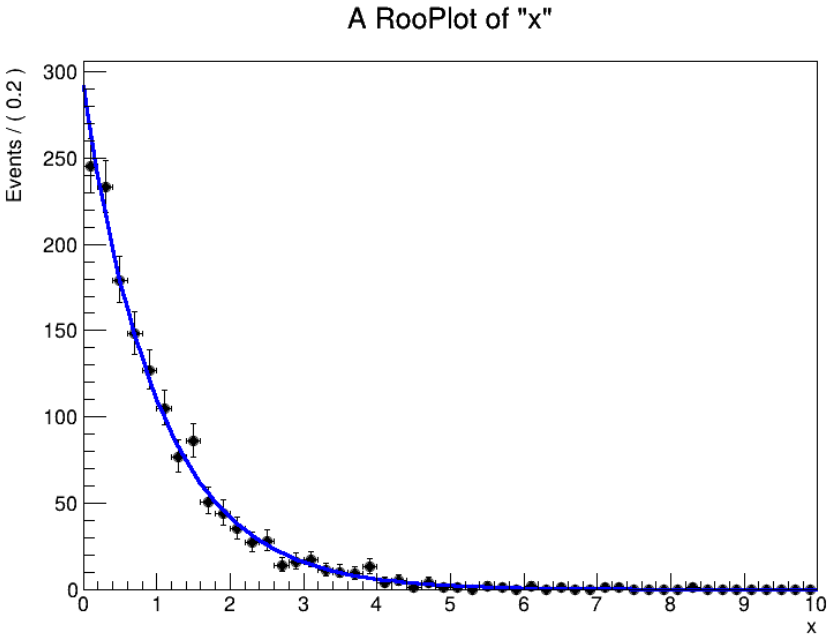

Figure 3: Enter Caption

Valores Esperados

- **Lambda inicial:** 1
- **Total de eventos gerados:** 1500