Universidade do Estado do Rio de Janeiro

Jully Moura Alves

Atividade 4

Rio de Janeiro

2023

Exercício 1

Para criar uma p.d.f Crystall Ball utilizando o root será feito um código em C que será visto a seguir:

Neste código, começamos “chamando” as bibliotecas que vamos usar utilizando o comando *#include <>* .

```
#include <iostream>
#include <RooRealVar.h>
#include <RooPlot.h>
#include <RooFitResult.h>
#include <RooFit.h>
#include <RooRandom.h>
#include <RooDataSet.h>
#include <TCanvas.h>
#include <RooArgSet.h>
#include <RooCrystalBall.h>
gSystem->Load("libRooFit.so");
gSystem->Load("libRooFitCore.so");
#include <TPaveText.h>
#include <TLegend.h>
```

Em seguida começamos a nossa void nomeada de “atv1” declarando as variáveis/parâmetros que serão usados para criar a p.d.f. com seus valores ajustados.

```
void atv1() {
    //Declarando as variáveis/parâmetros
    RooRealVar x("x", "x", -25, 25);
    RooRealVar media("media", "media", 2, -5, 2);
    RooRealVar sigma("sigma", "sigma", 1, 0.5, 2);
    RooRealVar alfa("alfa", "alfa", 1, 1, 1);
    RooRealVar n("n", "n", 1, 3, 1);
```

O próximo passo no código é criar o modelo da p.d.f utilizando as variáveis/parâmetros que foram definidos.

```
    RooCrystalBall crystallball("crystallball", "CrystallBall", x, media, sigma, alfa, n);

```

Com o modelo da p.d.f criado, é gerado um conjunto de dados com 1000 itens utilizando o *RooDataSet* e feito um ajuste utilizando o *RooFitResult* , note que os comandos chamam a “crystallball”, isto é porque os dados estão sendo gerados e ajustados a partir da p.d.f criada.

```
    //Gerando dados
    RooDataSet* dados = crystallball.generate(RooArgSet(x), 1000);

    //Fazendo ajuste
    RooFitResult* resultado = crystallball.fitTo(*dados, RooFit::Save());
```

Criou-se um frame associado a x com o Rooplot e foi feito o plote dos dados e do ajuste no frame.

```
    RooPlot* frame = x.frame();
    dados->plotOn(frame);
    crystallball.plotOn(frame);
```

Para visualizar o gráfico foi criado um canvas e usado o comando *Draw()* para “desenhar” o nosso frame dentro desse canvas.

```
    // Criação do canvas
    TCanvas* canvas = new TCanvas("canvas", "Eventos gerados com CrystallBall", 800, 600);
    frame->Draw();

```

E a seguir, adiciona-se a legenda.

```
    // Adicionando a legenda
    TLegend* legend = new TLegend(0.1, 0.7, 0.3, 0.9);
    legend->SetHeader("Informacoes Estatisticas");

    //RooRealVar para usar getVal()
    RooRealVar* mediaPtr = dynamic_cast<RooRealVar*>(resultado->floatParsFinal().find("media"));
    RooRealVar* sigmaPtr = dynamic_cast<RooRealVar*>(resultado->floatParsFinal().find("sigma"));

    if (mediaPtr) {
        legend->AddEntry("", Form("Media: %.2f", mediaPtr->getVal()), "");
    }
    if (sigmaPtr) {
        legend->AddEntry("", Form("Desvio Padrao: %.2f", sigmaPtr->getVal()), "");
    }
    legend->AddEntry("", Form("Alfa: %d", (int)dados->numEntries()), "");

```

Usa-se o *Draw* novamente, desta vez para adicionar a legenda, e *SaveAs* para salvar a imagem gerada:

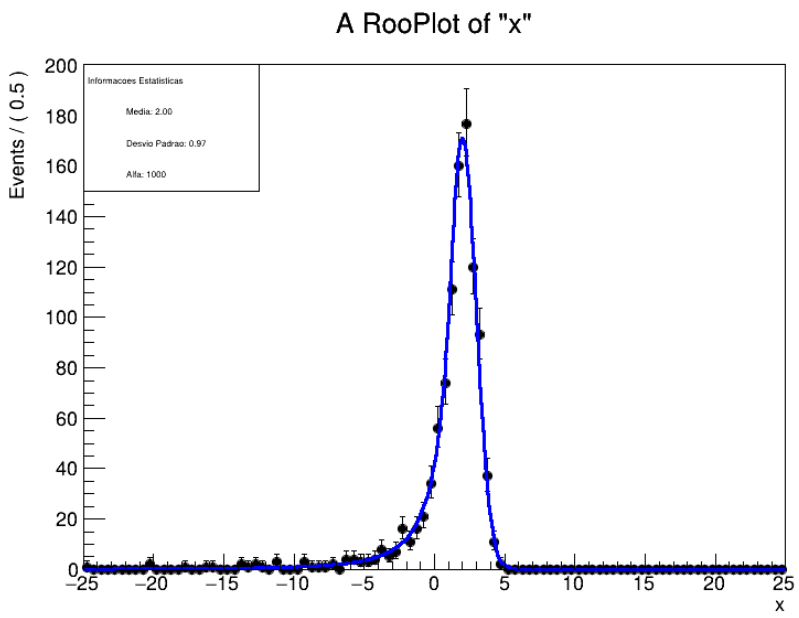
```
    // Desenha a legenda no canvas
    legend->Draw();

    // Salva o canvas
    canvas->SaveAs("CrystallballEventos.png");
}
```

Por fim, executamos a void “atv1” criada.

```
int main() {
    // Executando a função
    atv1();
    return 0;
}
```

Imagem Gerada



Exercício 2

Começamos novamente “chamando” as bibliotecas que serão utilizadas através do *#include* <>.

```
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooExponential.h>
#include <RooFitResult.h>
#include <RooPlot.h>
#include <RooRandom.h>
#include <TCanvas.h>
#include <TLatex.h>
#include <cstdio>
#include <TPaveText.h>
#include <TLegend.h>
```

Para este exercício, a void foi nomeada como “atv2”, e foi iniciada com a definição de variáveis/parâmetros:

```
void atv2() {

    //Definindo variáveis e parâmetros
    RooRealVar x("x", "x", 0, 10);
    RooRealVar lambda("lambda", "Taxa de decaimento", 1, 0.1, 2);
    RooRealVar Eventos("Eventos", "Número de Eventos", 1500, 100, 5000);
```

Para criar a função exponencial que será utilizada no ajuste utilizou-se o comando *RooExponential* .

```
    // Criando a função exponencial decrescente
    RooExponential expDecay("expDecay", "Decaimento Exponencial", x, lambda);
```

Gerou-se os dados novamente utilizando o R*ooDataSet**.

```
    // Gerando eventos
    RooDataSet* dados = expDecay.generate(RooArgSet(x), 1500);
```

E foi feito o ajuste com *RooFitResult**

```
    // Fazendo ajuste
    RooFitResult* fitResult = expDecay.fitTo(*dados, RooFit::Save(), RooFit::Extended(kTRUE));
```

Para conseguir gerar a imagem do gráfico, criou-se um frame.

```
    // Criando um frame
    RooPlot* frame = x.frame();
    dados->plotOn(frame);
    expDecay.plotOn(frame);
    frame->GetXaxis()->SetTitle("x");
    frame->GetYaxis()->SetTitle("Frequencia");
```

E, logo em seguida, criou-se um canvas e usou-se o comando Draw() pra “desenhar” o frame no canvas.

```
    // Criando um canvas
    TCanvas* canvas = new TCanvas("canvas", "Ajuste Exponencial", 800, 600);
    frame->Draw();
```

Para adicionar legenda no gráfico, desta vez utilizou-se o *TText** pois o método utilizado na atividade 1 estava apresentando erro quando aplicado neste código. *->SetNDC()* e *->SetTextSize()* permitem definir o fomato do texto.

```
41      // Criando texto para mostrar resultado
42      double adjustedLambda = lambda.getVal();
43      double adjustedEventos = Eventos.getVal();
44
45      TText* texto1 = new TText(0.2, 0.8, Form("Ajuste de \\lambda: %.3f", adjustedLambda));
46      TText* texto2 = new TText(0.2, 0.75, Form("Total de Eventos Ajustados: %.0f", adjustedEventos));
47
48      texto1->SetNDC();
49      texto1->SetTextSize(0.03);
50      texto2->SetNDC();
51      texto2->SetTextSize(0.03);
52
53      texto1->Draw();
54      texto2->Draw();
55
```

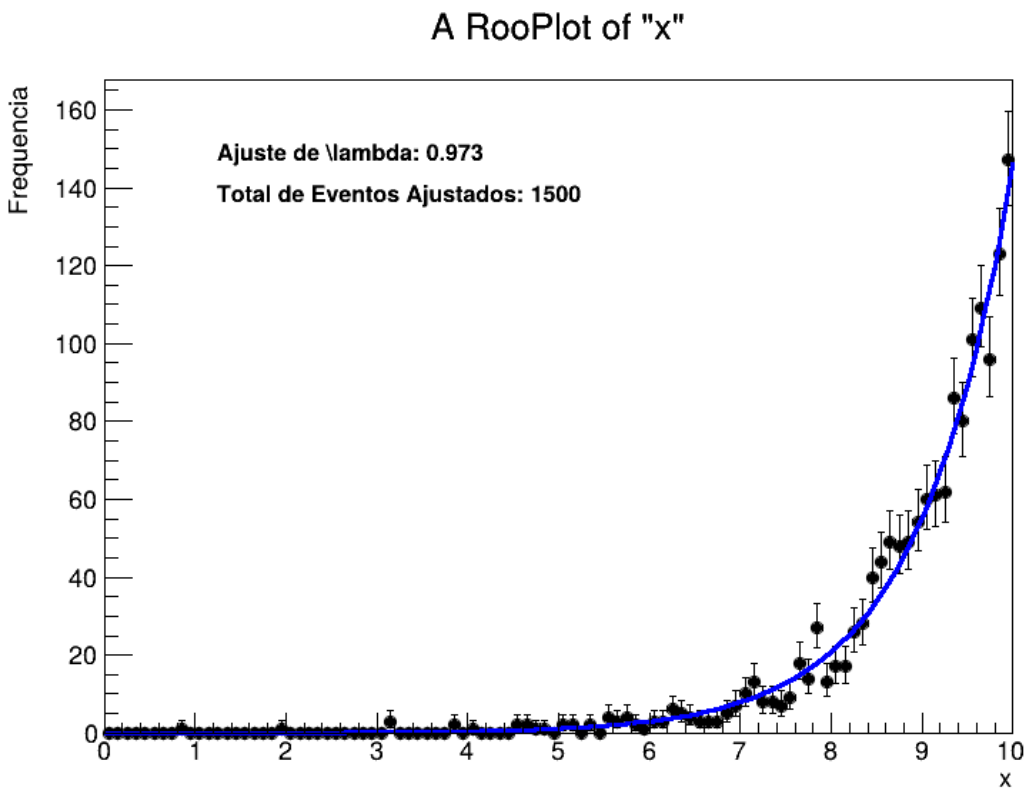
A imagem com todos os elementos foi salva com o comando *SaveAs*

```
57      // Salva o canvas
58      canvas->SaveAs("fitExponential.png");
59
```

Por último, chamamos a nossa função atv2 para ser executada;

```
63  int main() {
64      atv2();
65      return 0;
66  }
```

Imagem Gerada



λ= 0.973

Eventos ajustados = 1500

Como 0.973 <1, está dentro da expectativa.