**Introdução à Análise de dados em FAE** **(Data: 03/11/20)**

# Relatório ROOT e PYROOT.

*Professores:* Sandro Fonseca, Sheila Amaral, Eliza Melo. *Name:* João Pedro Gomes Pinheiro (`jgomespi`).

## 1º Parte: Exercício baseado no Tutorial sobre o ROOT.

Foi produzido um "esqueleto" chamado `Analyze` designado a fazer um *loop* sobre uma *tree* chamada `tree1` presente no arquivo `experiment.root`. A este "esqueleto" foram acrescentados alguns comandos pelos professores. Abaixo podemos ver uma cópia do arquivo `Analyze.cc` da forma como foi entregue aos alunos do curso. Ao decorrer do relatório, vamos mostrando quais comandos foram acrescentados, bem como seus objetivos. Ao fim, será mostrado o resultado final do arquivo `Analyze.cc`.

```
1  #define Analyze_cxx
2  #include "Analyze.h"
3  #include <TH2.h>
4  #include <TStyle.h>
5
6  //**********Definition section**********
7  TH1* chi2Hist = NULL;
8
9  void Analyze::Begin(TTree * /*tree*/)
10 {
11 TString option = GetOption();
12
13 //**********Initialization section**********
14 chi2Hist = new TH1D("chi2", "Histogram of Chi2", 100, 0, 20);
15 chi2Hist->GetXaxis()->SetTitle("chi2");
16 chi2Hist->GetYaxis()->SetTitle("number of events");
17 }
18
19 void Analyze::SlaveBegin(TTree * /*tree*/){}
20
21 Bool_t Analyze::Process(Long64_t entry)
22 {
23 // Don'‘t delete this line! Without it the program will crash
24 fReader.SetLocalEntry(entry);
25
26 //**********Loop section**********
27 GetEntry(entry);
28 chi2Hist->Fill(*chi2);
29
30 return kTRUE;
31 }
32
33 void Analyze::SlaveTerminate(){}
34
35 void Analyze::Terminate()
36 {
37 //**********Wrap-up section**********
38 chi2Hist->Draw();
39 }
```

O código acima cria um histograma chamado `chi2Hist` no método `Analyze::Begin`, e acrescenta o nome dos eixos $x$ e $y$. No método `Analyze::Process`, o histograma `chi2Hist` é preenchido com a variável `chi2`, presente em `tree1`. No método `Analyze:Terminate` utiliza-se o comando `Draw` para desenhar o histograma na tela.

Podemos acrescentar barras de erro no histograma `chi2Hist`, basta acrescentar a opção "`E1`" dentro do comando `Draw`, ou seja, a linha 38 do programa acima fica:

```
1  // Draw chi2Hist with error bars
```

```
chi2Hist->Draw("E1");
```

Além disso, o *range* do histograma foi alterado para de 0 a 2 de forma a facilitar a visualização das barras de erro.

Vamos criar um histograma para plotar a variável `ebeam` e um *scatterplot* com as variáveis `ebeam` e `chi2`. No método `Analyze::Begin`, acrescentamos os comandos que criam um histograma `TH1` chamado `ebeamHist`, com 100 bins, que vai de 149,0 a 151,0. Além disso, criamos um histograma `TH2` chamado `chi2ebeamHist`, que receberá as variáveis `ebeam` e `chi2`, este será nosso *scatterplot*. Observe que tanto o *range* quando a *binnagem* são iguais às dos respectivos histogramas `TH1` de cada variável. Também inserimos títulos nos eixos $x$ e $y$ de ambos os histogramas, como pode ser visto abaixo:

```
// Create a ebeamHist histogram, to Fill ebeam var, and set its axis titles.
ebeamHist = new TH1D("ebeam", "Histogram of ebeam", 100, 149., 151.);
ebeamHist->GetXaxis()->SetTitle("ebeam (GeV)");
ebeamHist->GetYaxis()->SetTitle("number of events");

// Create a scatterplot for ebeam and chi2 vars and set its axis titles.
chi2ebeamHist = new TH2D("chi2ebeam", "ScatterPlot of chi2 and ebeam", 100, 0, 2.,
    100, 149., 151.);
chi2ebeamHist->GetXaxis()->SetTitle("chi2");
chi2ebeamHist->GetYaxis()->SetTitle("ebeam (GeV)");
```

No método `Analyze::Process` preenchemos os histogramas `ebeamHist` e `chi2ebeamHist`, como mostrado abaixo:

```
// Fill ebeamHist with ebeam var
ebeamHist->Fill(*ebeam);
// Fill chi2ebeamHist with ebeam and chi2 vars
chi2ebeamHist->Fill(*chi2,*ebeam);
```

No método `Analyze::Terminate`, desenhamos os histogramas da tela. Vamos ajustar uma gaussiana na distribuição de `ebeam` e desenhá-la com barras de erro com o seguinte comando:

```
//Fit a gaussian to ebeam distribuition and draw ebeamHist with error bars
ebeamHist->Fit("gaus","V","E1",149.,151.);
```

Vamos desenhar o *scatterplot* com a palete de cores, e para isso acrescentamos a opção `COLZ` no comando `Draw`. Além disso, ajustamos a posição no título do eixo y, que acaba ficando muito próximo do eixo. Também utilizamos comandos para ajustar a posição da caixa de estatística, que estava sobrepondo o palete de cores.

```
// Set the position of Stat Box
gStyle->SetStatX(0.9);
gStyle->SetStatY(0.9);

// Set the position of chi2ebeamHist Y axis title
chi2ebeamHist->GetYaxis()->SetTitleOffset(1.4);
// Draw chi2ebeamHist scatterplot with "COLZ" option, which include the palette
    colors
chi2ebeamHist->Draw("COLZ");
```

Agora, vamos calcular o $p_T$ e o valor de $\theta$, a partir dos valores de $p_x$ e $p_y$ presentes em `tree1`. As equações 0.1 e 0.2 definem estas variáveis.

$$p_T = \sqrt{p_x^2 + p_y^2} \tag{0.1}$$

$$\theta = \arctan\left(\frac{p_T}{p_z}\right) \tag{0.2}$$

Primeiramente, no método `Analyze::Begin` definimos os respectivos histogramas `TH1` que serão preenchidos com estas variáveis (`ptHist` e `thetaHist`) e escolhemos seu *range* e sua *binnagem*.

```
// Create a ptHist histogram to Fill pT var and set its axis titles.
ptHist = new TH1D("pt", "Histogram of pT", 100, 0, 35);
ptHist->GetXaxis()->SetTitle("pT (GeV)");
ptHist->GetYaxis()->SetTitle("number of events");

```

```
// Create a thetaHist histogram to Fill theta var and set its axis titles.
thetaHist = new TH1D("theta", "Histogram of theta", 100, -3.15, 3.15);
thetaHist->GetXaxis()->SetTitle("theta");
thetaHist->GetYaxis()->SetTitle("number of events");
```

No arquivo `Analyze.h` acrescentamos as linhas abaixo para declarar as variáveis do tipo `Float_t` (`pT` e `theta`). Estas linhas devem ser acrescentadas dentro da classe `Analyze`, na parte **public**.

```
// Declaring pT and theta vars.
Float_t pT;
Float_t theta;
```

No método `Analyze::Process`, utilizando a biblioteca `TMath`, calculamos $p_T$ e $\theta$ com base nas Equações 0.1 e 0.2. Por fim, preenchemos os respectivos histogramas.

```
// Calc pT:
pT = TMath::Sqrt((*px)*(*px)+(*py)*(*py));
ptHist->Fill(pT);

// Calc theta:
theta = TMath::ATan2((pT),(*pz));
thetaHist->Fill(theta);
```

No método `Analyze::Terminate`, restauramos a posição da caixa de estatística com o comando `Reset` e ajustamos a posição do título do eixo $y$ do histograma `ptHist`. Por fim, desenhamos os histogramas `thetaHist` e `ptHist`.

```
// Reset the stat box
gStyle->Reset();
// Set the position of ptHist Y axis title
ptHist->GetYaxis()->SetTitleOffset(1.4);
// Draw thetaHist and ptHist
thetaHist->Draw();
ptHist->Draw();
```

Para contar a quantidade de eventos com $p_z < 145,0$ GeV, definimos dois contadores ($i$ e $j$) no arquivo `Analyze.h`, logo abaixo da definição de `pT` e `theta`, e atribuímos valor zero a eles:

```
// Declaring i and j to count the number of eventos above some cut defined on .C file
Int_t i=0;
Int_t j=0;
```

Agora sim vamos realizar a contagem dos eventos que possuem $p_z < 145,0$ GeV. No arquivo `Analyze.C`, no método `Analyze::Process`, acrescentamos os seguintes comandos:

```
// j will count all events
j++;
// i will count events with pz < 145.0 GeV
if (TMath::Abs(*pz)<145.) {
        // Here we print the value of pz (when pz<145 GeV) on screen
        std::cout << *pz << i << std::endl;
        i++;
}
```

Para descobrirmos quantos eventos possuem $p_z < 145,0$ GeV, inserimos o comando abaixo no método `Analyze::Terminate`:

```
// Print on screen how many events have pz<145 GeV
std::cout << "The number of events with pz<145.0 GeV is " << i << std::endl;
```

Para criar o arquivo `.root` (`experiment-output.root`) e escrever os histogramas nele, devemos inserir os comandos abaixo no método `Analyze::Terminate`:

```
// Recreate a file named "experiment-output.root" and write the histrograms
TFile f("experiment-output.root","recreate");
chi2Hist->Write();
ebeamHist->Write();
chi2ebeamHist->Write();
ptHist->Write();
```

```
thetaHist->Write();
f.Write();
f.Close();
```

Os programas e arquivos necessários para rodar esta análise estão num repositório `GitHub` que pode ser executado desta forma:

```
git clone git@github.com:jgomespi/ROOT-AnaliseDados.git
cd ROOT-AnaliseDados/
```

Entre no `root` e compile o programa:

```
root -l
[0] .L Analyze.C
```

Para rodar o programa, basta executar o comando abaixo numa sessão `root`:

```
[1] TFile *f = new TFile("experiment.root"); tree1->Process("Analyze.C")
```

Os outputs serão os valores de $p_z$ (para eventos onde este é menor que 145,0 GeV), os parâmetros do ajuste do histograma *ebeamHist* e, por fim, o número de eventos com $p_z < 145,0$ GeV. Além disso, serão desenhados os histogramas na tela e o arquivo `experiment-output.root` é criado com todos os histogramas citados aqui.

Abaixo deixamos uma cópia da versão final dos arquivos *Analyze.C* e *Analyze.h*, que também podem ser encontrados no repositório `GitHub`.

```
#define Analyze_cxx
#include "Analyze.h"
#include <TH2.h>
#include <TStyle.h>

//**********Definition section**********
TH1* chi2Hist = NULL;
TH1* ebeamHist = NULL;
TH2* chi2ebeamHist = NULL;
TH1* thetaHist = NULL;
TH1* ptHist = NULL;

void Analyze::Begin(TTree * /*tree*/)
{
TString option = GetOption();

//**********Initialization section**********
chi2Hist = new TH1D("chi2", "Histogram of Chi2", 100, 0, 2.);
chi2Hist->GetXaxis()->SetTitle("chi2");
chi2Hist->GetYaxis()->SetTitle("number of events");

// Create a ebeamHist histogram, to Fill ebeam var, and set its axis titles.
ebeamHist = new TH1D("ebeam", "Histogram of ebeam", 100, 149., 151.);
ebeamHist->GetXaxis()->SetTitle("ebeam (GeV)");
ebeamHist->GetYaxis()->SetTitle("number of events");

// Create a scatterplot for ebeam and chi2 vars and set its axis titles.
chi2ebeamHist = new TH2D("chi2ebeam", "ScatterPlot of chi2 and ebeam", 100, 0, 2.,
    100, 149., 151.);
chi2ebeamHist->GetXaxis()->SetTitle("chi2");
chi2ebeamHist->GetYaxis()->SetTitle("ebeam (GeV)");

// Create a ptHist histogram to Fill pT var (Float_t type, also created here) and set
     its axis titles.
ptHist = new TH1D("pt", "Histogram of pT", 100, 0, 35);
ptHist->GetXaxis()->SetTitle("pT (GeV)");
ptHist->GetYaxis()->SetTitle("number of events");

// Create a thetaHist histogram to Fill theta var (Float_t type, also created here)
    and set its axis titles.
thetaHist = new TH1D("theta", "Histogram of theta", 100, -3.15, 3.15);
thetaHist->GetXaxis()->SetTitle("theta");
```

```
40 thetaHist->GetYaxis()->SetTitle("number of events");
41
42
43 }
44
45 void Analyze::SlaveBegin(TTree * /*tree*/){}
46
47 Bool_t Analyze::Process(Long64_t entry)
48 {
49 // Dont delete this line! Without it the program will crash
50 fReader.SetLocalEntry(entry);
51
52 //**********Loop section**********
53 GetEntry(entry);
54 chi2Hist->Fill(*chi2);
55 // Fill ebeamHist with ebeam var
56 ebeamHist->Fill(*ebeam);
57 // Fill chi2ebeamHist with ebeam and chi2 vars
58 chi2ebeamHist->Fill(*chi2,*ebeam);
59
60 // Calc pT:
61 pT = TMath::Sqrt((*px)*(*px)+(*py)*(*py));
62 ptHist->Fill(pT);
63
64 // Calc theta:
65 theta = TMath::ATan2((pT),(*pz));
66 thetaHist->Fill(theta);
67
68 // j will count all events
69 j++;
70 // i will count events with pz < 145.0 GeV
71 if (TMath::Abs(*pz)<145.) {
72 // Here we print the value of pz (when pz<145 GeV) on screen
73 std::cout << *pz << i << std::endl;
74 i++;
75 }
76
77 return kTRUE;
78 }
79
80 void Analyze::SlaveTerminate(){}
81
82 void Analyze::Terminate()
83 {
84 //**********Wrap-up section**********
85 // Draw chi2Hist with error bars
86 chi2Hist->Draw("E1");
87
88 //Fit a gaussian to ebeam distribuition and draw ebeamHist with error bars
89 ebeamHist->Fit("gaus","V","E1",149.,151.);
90
91 // Set the position of Stat Box
92 gStyle->SetStatX(0.9);
93 gStyle->SetStatY(0.9);
94
95 // Set the position of chi2ebeamHist Y axis title
96 chi2ebeamHist->GetYaxis()->SetTitleOffset(1.4);
97 // Draw chi2ebeamHist scatterplot with "COLZ" option, which include the palette
    colors
98 chi2ebeamHist->Draw("COLZ");
99
100 // Reset the stat box
101 gStyle->Reset();
```

```
// Set the position of ptHist Y axis title
ptHist->GetYaxis()->SetTitleOffset(1.4);
// Draw thetaHist and ptHist
thetaHist->Draw();
ptHist->Draw();

// Print on screen how many events have pz<145 GeV
std::cout << "The number of events with pz<145.0 GeV is " << i << std::endl;

// Recreate a file named "experiment-output.root" and write the histrograms
TFile f("experiment-output.root","recreate");
chi2Hist->Write();
ebeamHist->Write();
chi2ebeamHist->Write();
ptHist->Write();
thetaHist->Write();
f.Write();
f.Close();
}
```

```
#ifndef Analyze_h
#define Analyze_h

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
#include <TSelector.h>
#include <TTreeReader.h>
#include <TTreeReaderValue.h>
#include <TTreeReaderArray.h>

// Headers needed by this particular selector


class Analyze : public TSelector {
public :
TTreeReader     fReader;  //!the tree reader
TTree          *fChain = 0;   //!pointer to the analyzed TTree or TChain

// Readers to access the data (delete the ones you do not need).
TTreeReaderValue<Int_t> event = {fReader, "event"};
TTreeReaderValue<Float_t> ebeam = {fReader, "ebeam"};
TTreeReaderValue<Float_t> px = {fReader, "px"};
TTreeReaderValue<Float_t> py = {fReader, "py"};
TTreeReaderValue<Float_t> pz = {fReader, "pz"};
TTreeReaderValue<Float_t> zv = {fReader, "zv"};
TTreeReaderValue<Float_t> chi2 = {fReader, "chi2"};


Analyze(TTree * /*tree*/ =0) { }
virtual ~Analyze() { }
virtual Int_t   Version() const { return 2; }
virtual void    Begin(TTree *tree);
virtual void    SlaveBegin(TTree *tree);
virtual void    Init(TTree *tree);
virtual Bool_t  Notify();
virtual Bool_t  Process(Long64_t entry);
virtual Int_t   GetEntry(Long64_t entry, Int_t getall = 0) { return fChain ? fChain->
    GetTree()->GetEntry(entry, getall) : 0; }
virtual void    SetOption(const char *option) { fOption = option; }
virtual void    SetObject(TObject *obj) { fObject = obj; }
virtual void    SetInputList(TList *input) { fInput = input; }
virtual TList  *GetOutputList() const { return fOutput; }
virtual void    SlaveTerminate();
```

```
44 virtual void    Terminate();
45 // Declaring pT and theta vars.
46 Float_t pT;
47 Float_t theta;
48 // Declaring i and j to count the number of eventos above some cut defined on .C file
49 Int_t i=0;
50 Int_t j=0;
51 ClassDef(Analyze,0);
52
53 };
54
55 #endif
56
57 #ifdef Analyze_cxx
58 void Analyze::Init(TTree *tree)
59 {
60 fReader.SetTree(tree);
61 }
62
63 Bool_t Analyze::Notify()
64 {
65 return kTRUE;
66 }
67
68
69 #endif // #ifdef Analyze_cxx
```