Introdução à Análise de dados em FAE **(Data: 03/11/2020)**

Relatório ROOT

Professores: Sandro Fonseca, Sheila Amaral, Eliza Melo *Name:* Matheus Macedo

Macro do ROOT no C++

Foi construido dois códigos em C++ com extensões, .h e .C . O primeiro caracteriza as funções que serão usadas no .C.
Abaixo, segue primeiramente o código .h

```
#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();
virtual void    Terminate();
// Declaring pT and theta vars.
Float_t pT;
Float_t theta;
// 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;
ClassDef(Analyze,0);

};

#endif

#ifdef Analyze_cxx
void Analyze::Init(TTree *tree)
{
fReader.SetTree(tree);
}

Bool_t Analyze::Notify()
{
return kTRUE;
}


#endif // #ifdef Analyze_cxx
```

De posse dessa estrutura, foi apresentado para nós uma formação primária do código em C para ser incrementado de acordo com o que os exercícios foram propostos durante a aula.

O arquivo .h tem nome de *Analize.h* e fazemos um **include Analyze.h** no código principal .C para fazer os histogramas propostos.
Abaixo está o código que será executado no ambiente ROOT (*Analize.C*).

```
#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();

// ********** Inicializando a secao **********
chi2Hist = new TH1D("chi2", "Histogram of Chi2", 100, 0, 2.);
chi2Hist->GetXaxis()->SetTitle("chi2");
chi2Hist->GetYaxis()->SetTitle("# Eventos");

//  Cria um histograma ebeamHist
ebeamHist = new TH1D("ebeam", "Histograma do ebeam", 100, 149., 151.);
ebeamHist->GetXaxis()->SetTitle("ebeam (GeV)");
ebeamHist->GetYaxis()->SetTitle("# Eventos");

//  Cria um plot de dispersao entre o chi2 e ebeam
chi2ebeamHist = new TH2F("chi2xebeam", "Dispersao do chi2 e ebeam", 100, 0.4, 1.6,
    100, 149.4, 150.6);
chi2ebeamHist->GetXaxis()->SetTitle("chi2");
chi2ebeamHist->GetYaxis()->SetTitle("ebeam (GeV)");

//  Cria um histograma do Pt
ptHist = new TH1D("p_{T}", "Histograma do p_{T}", 100, 0, 35);
ptHist->GetXaxis()->SetTitle("pT (GeV)");
```

```
36 ptHist->GetYaxis()->SetTitle("# Eventos");
37
38 //  Cria um histograma thetaHist
39 thetaHist = new TH1D("theta", "Histograma do theta", 100, -3.15, 3.15);
40 thetaHist->GetXaxis()->SetTitle("theta");
41 thetaHist->GetYaxis()->SetTitle("# Eventos");
42
43
44 }
45
46 void Analyze::SlaveBegin(TTree * /*tree*/){}
47
48 Bool_t Analyze::Process(Long64_t entry)
49 {
50 // Do not delete this line! Without it the program will crash
51 fReader.SetLocalEntry(entry);
52
53 //**********Loop section**********
54 GetEntry(entry);
55 chi2Hist->Fill(*chi2);
56
57 ebeamHist->Fill(*ebeam);
58
59 chi2ebeamHist->Fill(*chi2,*ebeam);
60
61 // Calculo do pT e preechendoo o histograma:
62 pT = TMath::Sqrt((*px)*(*px)+(*py)*(*py));
63 ptHist->Fill(pT);
64
65 // Calculo do theta e preechendoo o histograma:
66 theta = TMath::ATan2((*py),(*px));
67 thetaHist->Fill(theta);
68
69
70 // j conta todos os eventos
71 j++;
72 // i conta todos os eventos com pz < 145.0 GeV
73 if (TMath::Abs(*pz)<145.) {
74 // Here we print the value of pz (when pz<145 GeV) on screen
75 std::cout << *pz << i << std::endl;
76 i++;
77 }
78
79 return kTRUE;
80 }
81
82 void Analyze::SlaveTerminate(){}
83
84 void Analyze::Terminate()
85 {
86 //**********Wrap-up section**********
87 // Draw chi2Hist with error bars
88 chi2Hist->Draw("E1SAME");
89
90 //Fit a gaussian to ebeam distribuition and draw ebeamHist with error bars
91 ebeamHist->Fit("gaus","V","E1SAME",149.,151.);
92
93 // Set the position of Stat Box
94 gStyle->SetStatX(0.9);
95 gStyle->SetStatY(0.9);
96
97 // Set the position of chi2ebeamHist Y axis title
98 chi2ebeamHist->GetYaxis()->SetTitleOffset(1.4);
```

```
99
100 // Plot de dispersao do chi2 e ebeam Hist
101
102 chi2ebeamHist->Draw("scat=0.8");
103
104 // Reset the stat box
105 gStyle->Reset();
106 // Set the position of ptHist Y axis title
107 ptHist->GetYaxis()->SetTitleOffset(1.4);
108 // Draw thetaHist and ptHist
109 thetaHist->SetFillColor(6);
110 thetaHist->Draw();
111
112 ptHist->SetFillColor(30);
113 ptHist->Draw();
114
115
116 // Print on screen how many events have pz<145 GeV
117 std::cout << "The number of events with pz < 145.0 is " << i << std::endl;
118
119 // Recrie um arquivo chamado "experiment-output.root" e escreva os histogramas
120
121 TFile file("experiment-output.root","recreate");
122 chi2Hist->Write();
123 ebeamHist->Write();
124 chi2ebeamHist->Write();
125 ptHist->Write();
126 thetaHist->Write();
127 file.Write();
128 file.Close();
129 }
```

Para executar o código, entra no terminal do ROOT e compila o código **Analyze.C**

```
1 root -l
2 [0] .L Analyze.C
```

Em seguida, fazer:

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