



ROOT

Part 1

Introdução

ROOT

ROOT pode ser visto como uma coleção de diversas ferramentas diferentes, como:

- ★ **Análise de dados: histogramas, gráficos, funções**
- ★ **I/O: armazenamento de qualquer objeto de C++, tipo-linha, tipo-coluna**
- ★ **Ferramentas estatísticas** (RooFit/RooStats): modelagem e inferência
- ★ **Math: funções não triviais** (e.g. Erf, Bessel) e otimizadas
- ★ **Interpretador de C++:** totalmente conforme a linguagem
- ★ **Análise Multivariável** (TMVA): e.g. Boosted decision trees (BDT), Redes Neurais
- ★ **Gráficos avançados** (2D, 3D, display de eventos)
- ★ e muito mais: servidor HTTP, visualização em JavaScript



★ Unstar

595

Fork

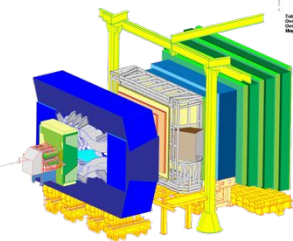
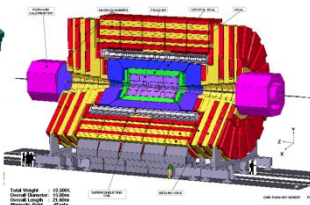
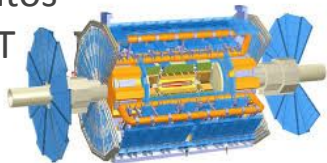
433

 176 contributors

<https://github.com/root-project/root>

ROOT - aplicações

Alguns experimentos
que utilizam ROOT



Event Filtering

Data

Offline Processing

Analysis

Reconstruction

Further
processing,
skimming

Event
Selection,
statistical
treatment ...

Raw

Reco

...

Analysis
Formats

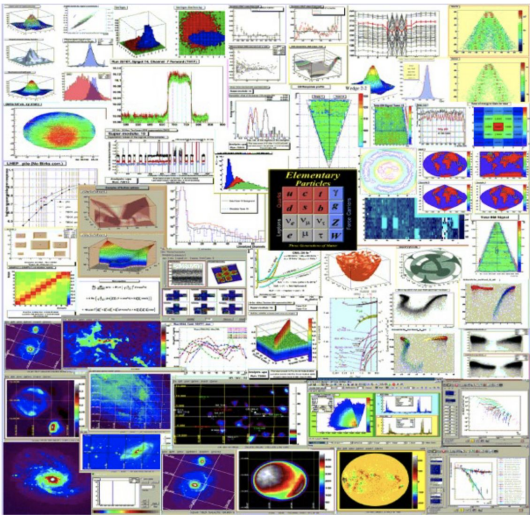
Plots

Data Storage: Local, Network

Dados no formato ROOT no LHC

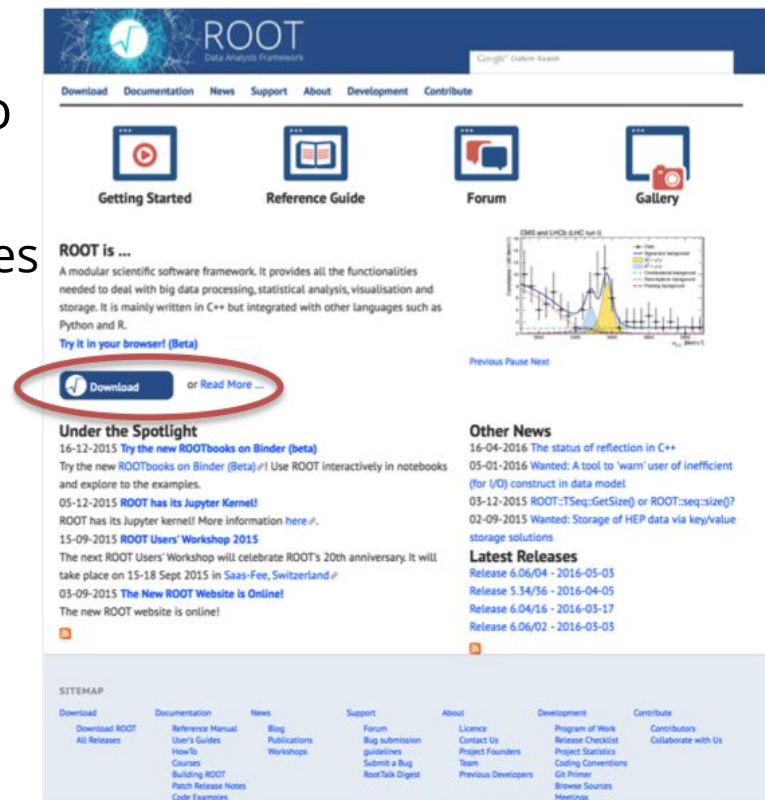
~1 EB

informação de 2019



★ ROOT web site: a principal fonte de informação e ajuda para os usuários do ROOT

- Tanto para iniciantes quanto experientes
- Downloads, instruções de instalação
- Documentação de todas as classes
- Manuals, tutoriais, apresentações
- Fórum
- ...



Principais fontes do ROOT

- ★ ROOT Website: <https://root.cern>
- ★ Treinamento: <https://github.com/root-project/training>
- ★ Mais material: <https://root.cern/ROOTPrimer>
 - um tutorial para os iniciantes: **the “ROOT Primer”**
- ★ Guia com todas as referências ao ROOT:
<https://root.cern/doc/master/index.html>
- ★ Fórum: <https://root-forum.cern.ch>

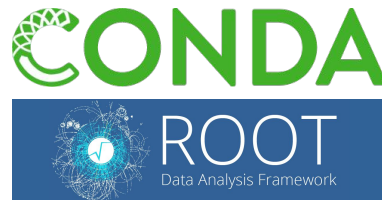
- ★ Pega o código fonte do ROOT:
 - `git clone` <http://github.com/root-project/root>
 - ou daqui <https://root.cern.ch/content/release-61600>
- ★ Criando um diretório **build** e configurando o ROOT:
 - `mkdir rootBuild; cd rootBuild`
 - `cmake ../root`
 - para mais opções de configuração: <https://root.cern.ch/building-root>
- ★ Compilando
 - `make -j`
- ★ Prepare o ambiente de trabalho:
 - `. bin/thisroot.sh`

Eu particularmente, incluo uma chamada "source \$ROOTPATH/bin/thisroot.sh" no arquivo de chamada sistema.

Instalação direto da fonte

- ★ Instala o conda - miniconda
 - Download
 - <https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html>
 - `#bash Miniconda3-latest-MacOSX-x86_64.sh`

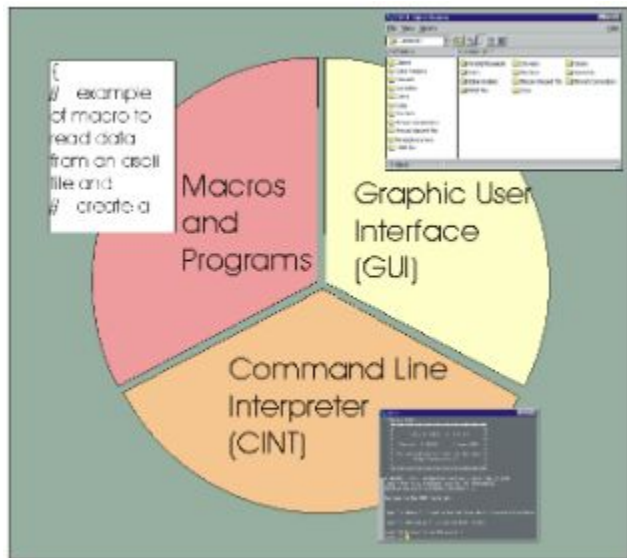
- ★ Cria um ambiente e instale o ROOT
 - `conda create -n my_root_env root -c conda-forge`
 - `conda activate my_root_env`
 - `root`



FUNCIONANDO!!!!!!

ROOT prompt e Macros

Interface de usuário



```
sheilamarass — root.exe — 80x24
sheilamarass@amaral:~$ export ROOTSYS=root/
sheilamarass@amaral:~$ export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
sheilamarass@amaral:~$ export PATH=$ROOTSYS/bin:$PATH
sheilamarass@amaral:~$ root

*****
*                                     *
*           W E L C O M E  t o  R O O T           *
*                                     *
*   Version   5.34/36           5 April 2016   *
*                                     *
* You are welcome to visit our Web site *
*           http://root.cern.ch             *
*                                     *
*****

ROOT 5.34/36 (v5-34-36@v5-34-36, Apr 05 2016, 10:25:45 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] █
```

.q	Quit - Sair
.L macro.C	Carregando uma macro
.x macro.C	Carregando e executando uma macro
.x macro.C++	Compilando e executando

The ROOT prompt

- ★ C++ é uma linguagem de programação compilada
 - Traduz o código fonte para a linguagem de máquina
- ★ ROOT fornece um **interpretador** C++
 - C++ interativo, de forma que não é necessário um compilador
 - como Python, Ruby, Haskell ...
 - o código é compilado **Just-in-Time!**
 - É inicializado com o comando:

root

- O shell interativo também é chamado de “prompt ROOT” ou “prompt interativo do ROOT”

Controlando o ROOT

- ★ Comandos especiais que não são de C++, podem ser digitados no prompt, eles começam com um “.”

```
root [1] .<command>
```

- ★ Por exemplo:
 - Para sair do ROOT: **.q**
 - Para executar um comando da shell: **.! <OS_command>**
 - Para carregar uma macro: **.L <file_name>** (mais sobre macros a seguir)
 - **.help** ou **.?** para a lista completa de comandos

ROOT como calculadora

$$\begin{aligned}\frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \dots \\ &= \sum_{n=0}^{\infty} x^n\end{aligned}$$

Dando um passo a frente

Declarando **variáveis** e usando *for*

```
root [0] double x=.5  
(double) 0.5  
root [1] int N=30  
(int) 30  
root [2] double gs=0;
```

```
root [3] for (int i=0;i<N;++i) gs += pow(x,i)  
root [4] std::abs(gs - (1/(1-x)))  
(Double_t) 1.86265e-09
```

Modo interactivo

```
root [0] #include "a.h"
root [1] A o("ThisName"); o.printName()
ThisName
root [1] dummy()
(int) 42
```

a.h


```
# include <iostream>
class A {
public:
    A(const char* n) : m_name(n) {}
    void printName() { std::cout << m_name << std::endl; }
private:
    const std::string m_name;
};

int dummy() { return 42; }
```


ROOT macros

- ★ Vimos como utilizar o ROOT de forma interativa usando o prompt
- ★ O próximo passo é escrever “ROOT Macros” – programas leves
- ★ Segue a estrutura de uma macro armazenada em um arquivo *MacroName.C*:

Função, mesmo nome do arquivo



```
void MacroName() {  
    <          ...  
    your lines of C++ code  
    ...      >  
}
```

Unnamed ROOT macros

- ★ As macros também podem ser definidas sem nome
 - Não podem ser chamadas como funções!
 - Veja os próximos slides :)

```
{  
    <          ...  
    your lines of C++ code  
          ...          >  
}
```

Running a macro

- ★ Uma macro é executada no terminal digitando:

```
> root MacroName.C
```

- ★ ou execute dentro do prompt do ROOT usando **.x**:

```
> root  
root [0] .x MacroName.C
```

- ★ ou ainda pode carregá-la primeiro dentro de uma sessão do ROOT e depois executar a função definida na macro:

```
root [0] .L MacroName.C  
root [1] MacroName();
```

Interpretação e compilação

- ▶ Vimos como o ROOT interpreta e compila o código “Just-inTime”. O ROOT também permite compilar o código “tradicionalmente”.

```
root [1] .L macro1.C+  
root [2] macro1()
```

Generate shared library
and execute function



Advanced Users

```
int main() {  
    ExampleMacro();  
    return 0;  
}
```

```
> g++ -o ExampleMacro ExampleMacro.C `root-config --cflags --libs`  
> ./ExampleMacro
```

Convenções utilizadas no ROOT

- ★ **Classes** começam com **T**
- ★ **Non-class types** terminam com **_t**
- ★ **Member functions** começam com letras maiúsculas
- ★ **Constants** começam com **k**
- ★ **Global variables** começam com a letra **g**
- ★ **Getters e setters** começam com **Get** e **Set**
- ★ Tipos pré definidos no ROOT:
 - `Int_t`, `Float_t`, `Double_t`, `Bool_t`, etc
 - mas, pode-se utilizar de tipos do C++: `int`, `double`, etc...
- ★ ROOT tem um conjunto de variáveis globais que aplicadas a cada sessão
 - Por exemplo, uma única instância do TROOT é acessível por meio do gROOT e contém informações relativas apenas à sessão atual

```
gROOT->Reset();  
gROOT->LoadMacro("ftions.cxx");  
gSystem->Load("libMyEvent.so")
```

Sintaxe

Muitos dos comandos utilizados terão esta forma geral:

“constructor”
TSomething* mything = new TSomething(stuff);

All ROOT classes start with T:
TFile
TH1
TTree
TCanvas
...

Means make a “pointer” (in the case, of type TSomething)

Por enquanto, não se preocupe com o que é um ponteiro. Não é importante para esta aula.

Name of pointer

C++ operator that allocates memory

Initializes the allocated memory with whatever “stuff” TSomething requires

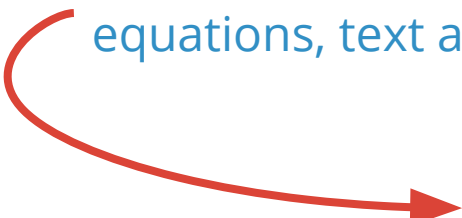
Nota: Em C ++, se você alocar memória usando o operador “novo”, você deve usar mais tarde “delete mything” para liberar a memória ... caso contrário, seu código terá um **memory leak**.

Não nos preocupemos com isso hoje, mas mantenha isso em mente quando for escrever seus códigos.

ROOTBooks

The Jupyter Notebook

A web-based interactive computing platform that combines code, equations, text and visualisations.



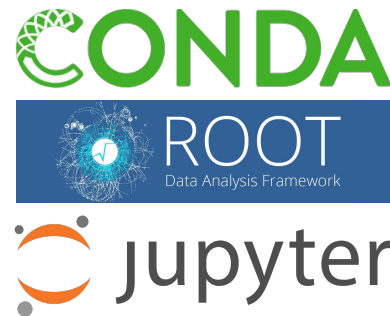
Many supported languages: C++, Python, Haskell, Julia...
One generally speaks about a “kernel” for a specific language

In a nutshell: an “interactive shell opened within the browser”



ROOT interfaces on Jupyter notebook

- ★ ROOT is well integrated with Jupyter Notebook, both for what concerns its Python and C++ interface
- ★ What is Jupyter Notebook? <https://jupyter.org/>
 - Language of choice, share notebooks, interactive output, big data integration
- ★ **How to integrate Jupyter notebook and ROOT:**
 - `conda activate my_root_env`
 - `conda config --env --add channels conda-forge`
 - `root --notebook`



Como funciona

Text

Code

Graphics

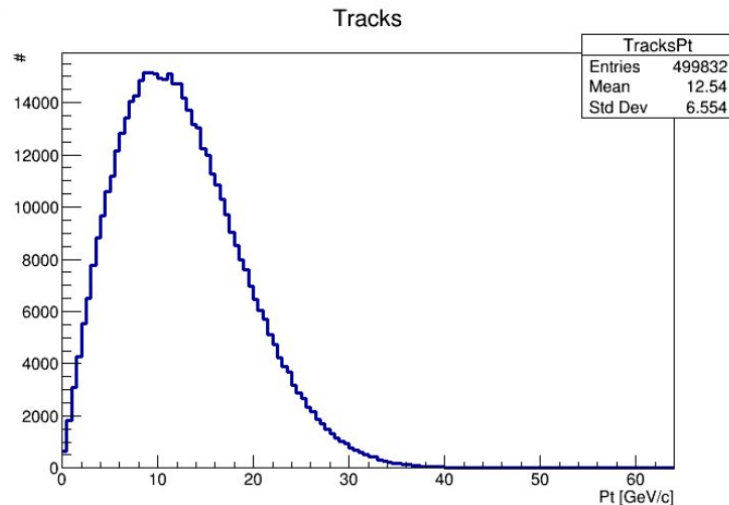
Access TTree in Python using PyROOT and fill a histogram

Loop over the TTree called "events" in a file located on the web. The tree is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
In [1]: import ROOT

f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root");
h = ROOT.TH1F("TracksPt", "Tracks;Pt [GeV/c];#", 128, 0, 64)
for event in f.events:
    for track in event.tracks:
        h.Fill(track.Pt())
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```

To execute the code,
click shift+enter



Vamos ver como é o ROOT no Jupyter Notebook

Você pode fazer um fork para a sua conta do GitHub desse repositório:

https://github.com/Analise-Dados-FAE/2021/tree/main/aula4_root

Esses ROOT notebooks são baseados no ROOT Primer

(<https://root.cern.ch/guides/primer>).

Histogramas, Gráficos e Funções

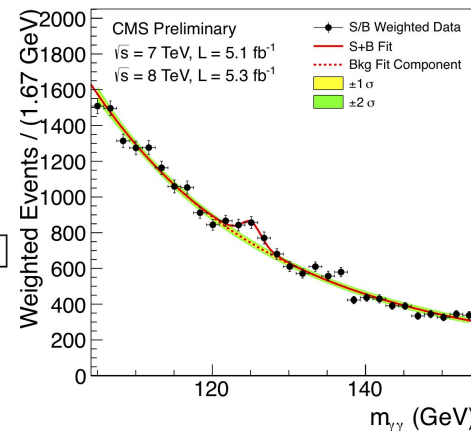
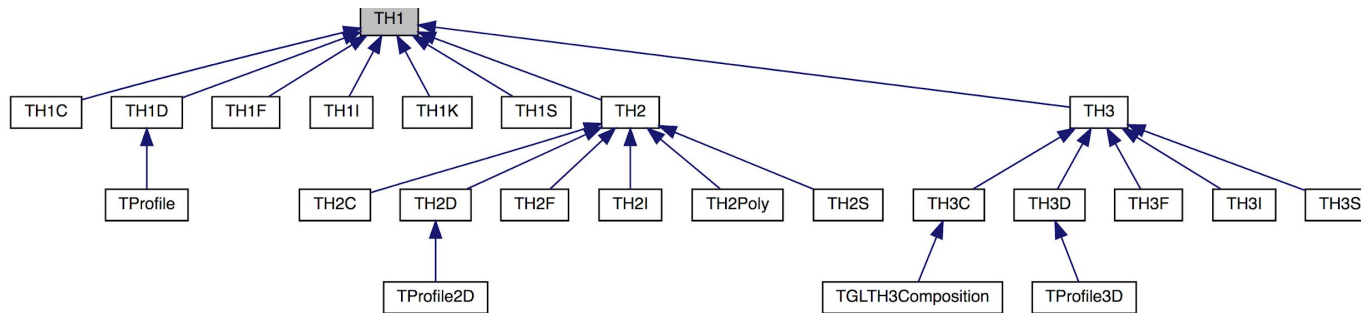
TCanvas

- ★ Canvas é janela do ROOT.
- ★ Os objetos gráficos do ROOT são vinculados ao Pad.

Command	Action
<code>c1 = new TCanvas("c1","Title, w, h)</code>	Creates a new canvas with width equal to w number of pixels and height equal to h number of pixels.
<code>c1->Divide(2,2);</code>	Divides the canvas to 4 pads.
<code>c1->cd(3)</code>	Select the 3 rd Pad
<code>c1->SetGridx(); c1->SetGridy(); c1->SetLogy();</code>	You can set grid along x and y axis. You can also set log scale plots.

Histogramas

- ★ A forma mais simples de apresentação dos dados
 - Por exemplo, bilhões de colisões podem ser mostrada em apenas alguns histogramas
 - Possível calcular o momento: média, rms, ...
 - ★ Coleta as quantidades em categorias discretas, bins. O ROOT possuem uma rica gama de tipos de histogramas
- Em cada bin do histogramas é armazenado um *float*



Histogramas 1D: ROOT

- ★ 1D histogram: `TH1F *name = new TH1F("name", "title", bins, lowest bin, highest bin);`

Example:

h1 is an instance of a TH1F class

```
root [0] TH1F *h1 = new TH1F("h1", "x distribution", 100, -4, 4);  
root [1] h1->FillRandom("gaus")  
root [2] h1->Draw()
```

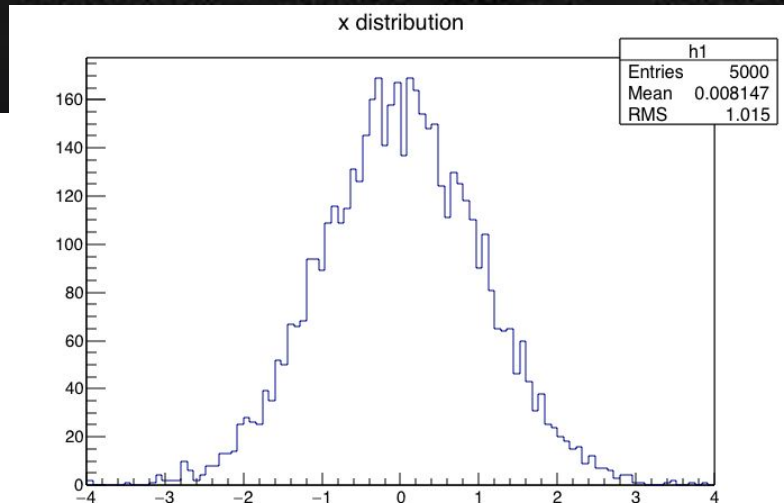
The Draw method display the histogram

1D Histograms: ROOT

- ★ 1D histogram: `TH1F *name = new TH1F("name", "title", bins, lowest bin, highest bin);`

Exemplo:

```
root [0] TH1F *h1 = new TH1F("h1", "x distribution", 100, -4, 4);  
root [1] h1->FillRandom("gaus")  
root [2] h1->Draw()
```

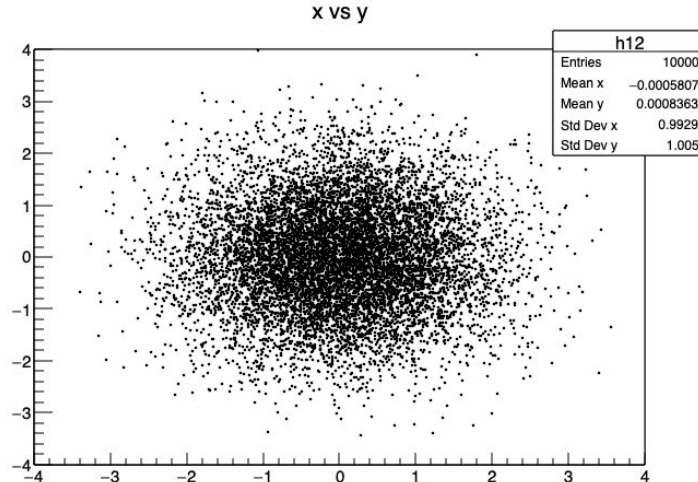


Histogramas 2D: ROOT

★ 2D histogram: `TH2F *name = new TH2F("name", "title", xbins, low xbin, up xbin, ybins, low ybin, up2 ybin);`

★ Exemplo:

```
auto h12 = new TH2F("h12", " x vs y ", 100, -4.0, 4.0, 100, -4.0, 4.0);  
for (Int_t i = 0; i < 10000; i++) h12->Fill(gRandom->Gaus(), gRandom->Gaus());  
h12->Draw();
```



3D Histograms: ROOT

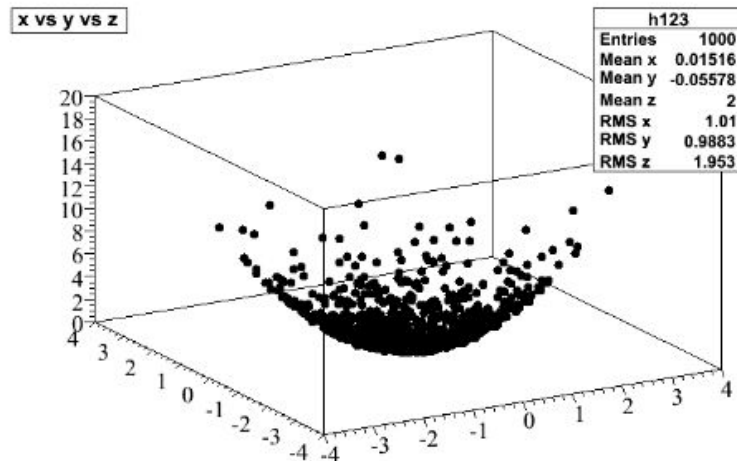
★ 3D histogram: `TH3F *name = new TH3F("name", "title", xbins, low xbin, up xbin, ybins, low ybin, up ybin, zbins, low zbin, up zbin);`

★ Exemplo:

```
auto h123 = new TH3F("h123", "x vs y vs z", 100, -4, 4, 100, -4, 4, 100, -4, 4);
```

...

```
h123->Draw();
```

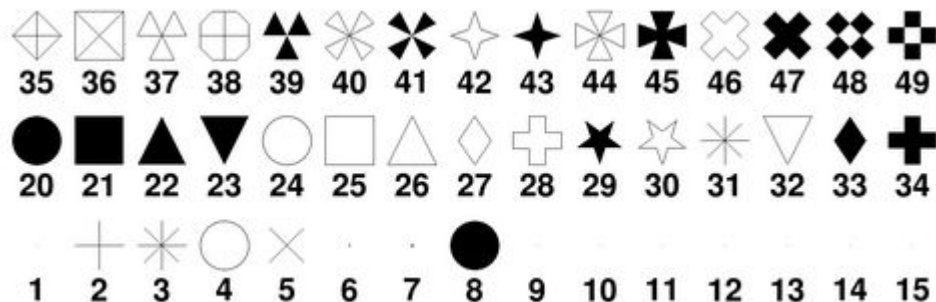


Propriedades dos histogramas

Command	Parameters
GetMean()	Mean
GetRMS()	Root of Variance
GetMaximum()	Maximum bin content
GetMaximumBin(int bin_number);	Location of maximum
GetBinCenter(int bin_number);	Center of bin
GetBinContent(int bin_number);	Content of bin

Propriedades dos histogramas - cosmética

h1.SetMarkerStyle();



h1.SetFillColor();

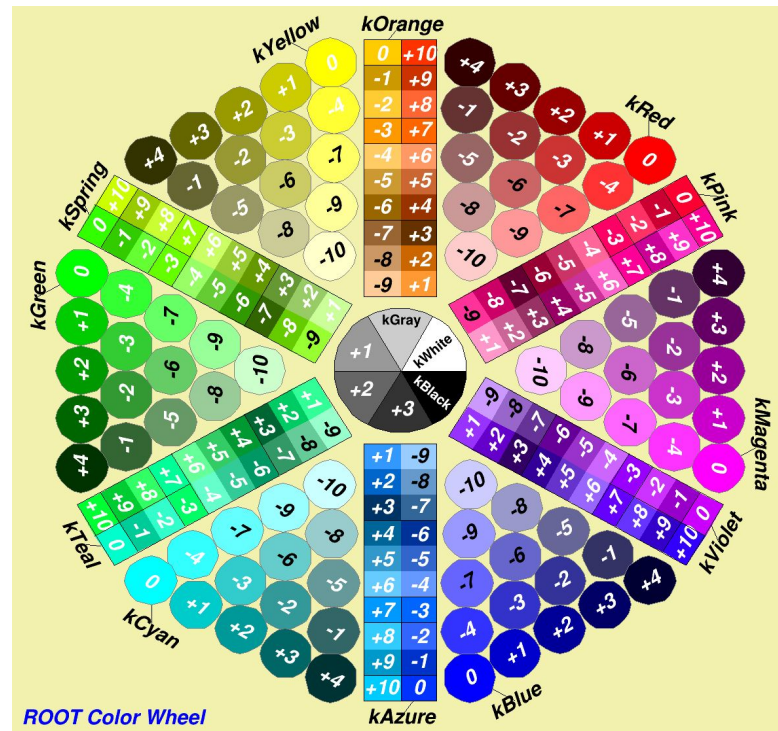
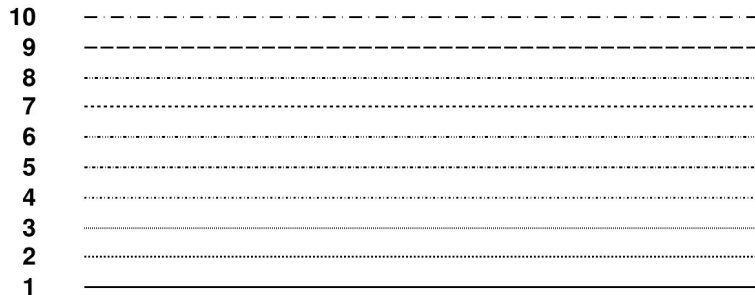


Propriedades dos histogramas - cosmética - linhas

h1.SetLineWidth();

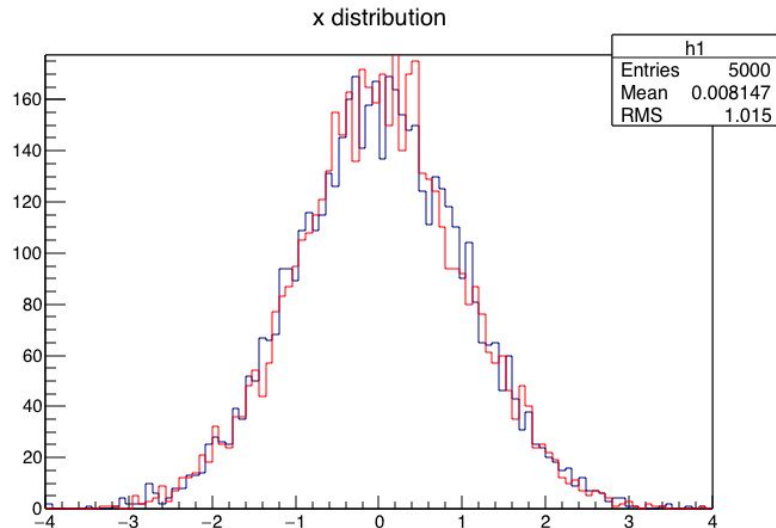


h1.SetLineStyle();



Opções dos histogramas

```
root [0] TH1F *h1 = new TH1F("h1", "x distribution", 100, -4, 4);
root [1] TH1F *h1same = new TH1F("h1same", "x distribution", 100, -4, 4);
root [2] h1->FillRandom("gaus")
root [3] h1same->FillRandom("gaus")
root [4] h1->Draw()
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [5] h1same->Draw("same")
```



" SAME": Superimpose on previous picture in the same pad.

" CYL": Use cylindrical coordinates.

" POL": Use polar coordinates.

" SPH": Use spherical coordinates.

" PSR": Use pseudo-rapidity/phi coordinates.

" LEGO": Draw a lego plot with hidden line removal.

" LEGO1": Draw a lego plot with hidden surface removal.

" LEGO2": Draw a lego plot using colors to show the cell contents.

" SURF": Draw a surface plot with hidden line removal.

" SURF1": Draw a surface plot with hidden surface removal.

" SURF2": Draw a surface plot using colors to show the cell contents.

" SURF3": Same as SURF with a contour view on the top.

" SURF4": Draw a surface plot using Gouraud shading.

" SURF5": Same as SURF3 but only the colored contour is drawn.

Note: Please check chapter 3 in user's guide to learn more about options.

Gráficos

- ★ Os gráficos são feitos a partir de n pontos (x,y)
- ★ Mostrar pontos e erros
- ★ **Fundamental para exibir tendências**

Vamos ver as classes TGraph and TGraphErrors classes

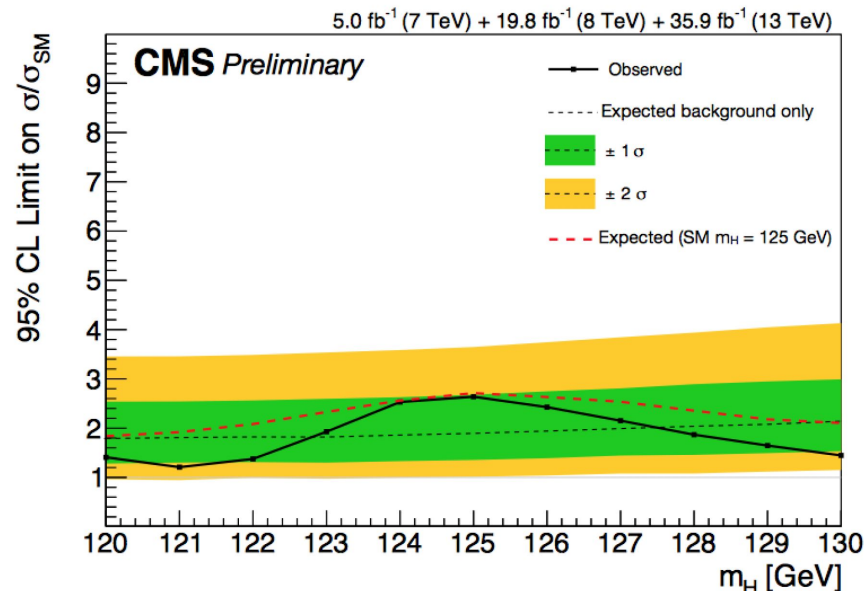
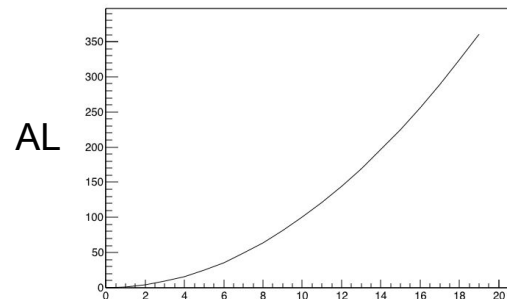
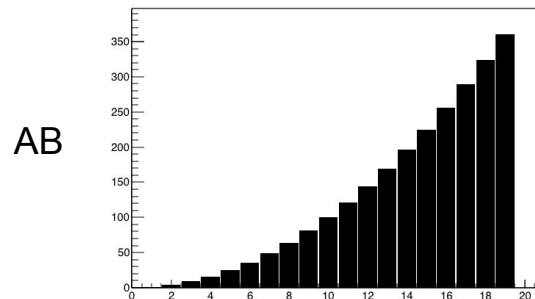
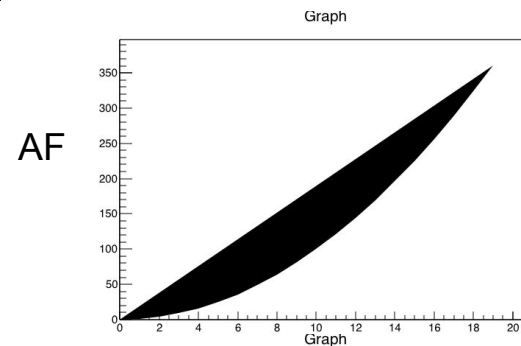
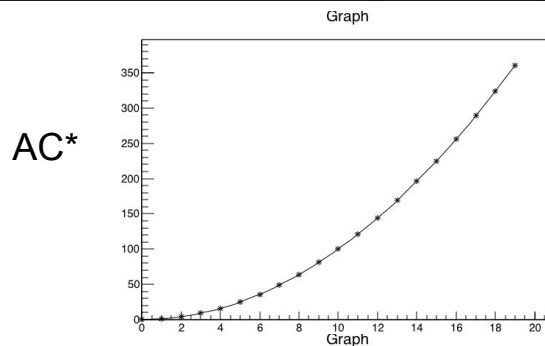


Gráfico: exemplo

```
root [0] Int_t n=20;  
root [1] Double_t x[n], y[n];  
root [2] for(Int_t i=0; i<n; i++){ x[i]=i; y[i]=i*i; }  
root [3] TGraph *gr1=new TGraph(n,x,y);  
root [4] gr1->Draw("AC*");
```



Superpondo dois gráficos

```
root [0] Int_t n=10;  
root [1] Double_t x[n], y[n], x1[n], y1[n];  
root [2] for(Int_t i=0; i<n; i++){ x[i]=i; y[i]=sin(i); x1[i]=i; y1[i]=cos(i); }  
root [3] TGraph *gr1=new TGraph(n,x,y);  
root [4] TGraph *gr2=new TGraph(n,x1,y1);  
root [5] gr1->SetLineColor(4);  
root [6] gr2->SetLineWidth(3);  
root [7] gr2->SetMarkerStyle(21);  
root [8] gr2->SetLineColor(2);  
root [9] gr1->Draw("AC*");  
Info in <TCanvas::MakeDefCanvas>: created default canvas  
root [10] gr2->Draw("CP");
```

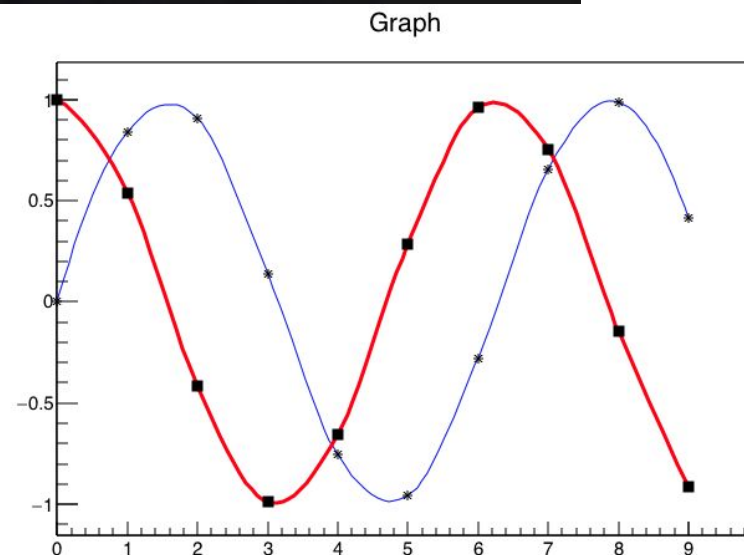
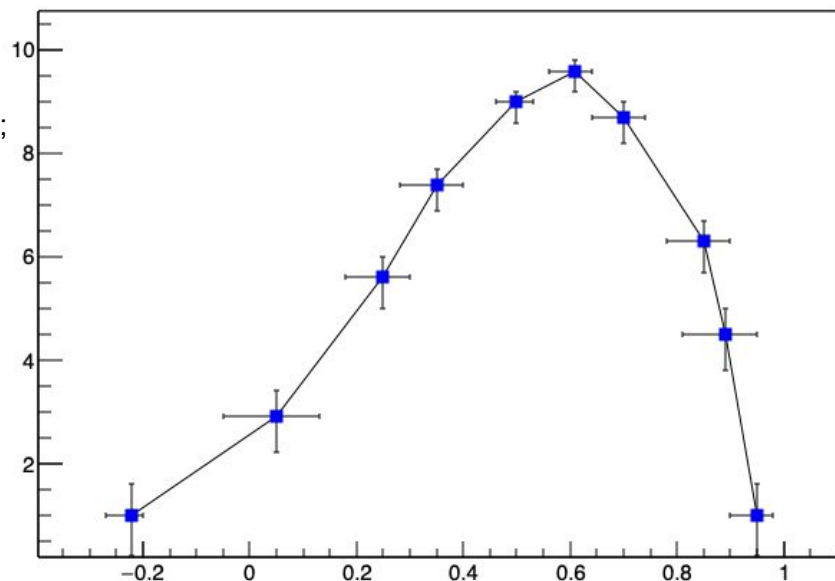


Gráfico com barras de erros

```
auto c45 = new TCanvas("c45","c45",200,10,600,400);
const Int_t n = 10;
Double_t x[n] = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61, 0.7, 0.85, 0.89, 0.95};
Double_t y[n] = {1, 2.9, 5.6, 7.4, 9.9, 9.6, 8.7, 6.3, 4.5, 1};
Double_t exl[n] = {.05, .1, .07, .07, .04, .05, .06, .07, .08, .05};
Double_t eyl[n] = {.8, .7, .6, .5, .4, .4, .5, .6, .7, .8};
Double_t exh[n] = {.02, .08, .05, .05, .03, .03, .04, .05, .06, .03};
Double_t eyh[n] = {.6, .5, .4, .3, .2, .2, .3, .4, .5, .6};
Double_t exld[n] = {.0, .0, .0, .0, .0, .0, .0, .0, .0, .0};
Double_t eyld[n] = {.0, .0, .0, .0, .0, .0, .0, .0, .0, .0};
Double_t exhd[n] = {.0, .0, .0, .0, .0, .0, .0, .0, .0, .0};
Double_t eyhd[n] = {.0, .0, .0, .0, .0, .0, .0, .0, .0, .0};
auto gr = new TGraphBentErrors(n, x, y, exl, exh, eyl, eyh, exld, exhd, eyld, eyhd);
gr->SetTitle("TGraphBentErrors Example");
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->Draw("ALP");
```

TGraphBentErrors Example



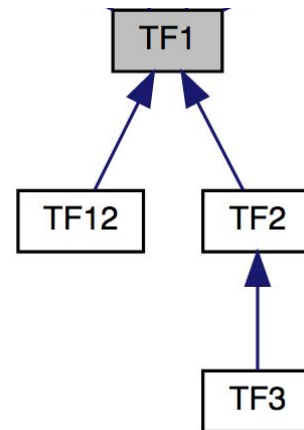
Documentação TGraphPainter:

<https://root.cern.ch/doc/master/classTGraphPainter.html>

Funções

- ★ Funções matemáticas são representadas na classe **TF1**
- ★ Ela tem nomes, fórmulas, propriedades, podem ser calculadas assim como suas integrais e derivadas
 - Por enquanto, vamos ver técnicas numéricas

option	description
"SAME"	superimpose on top of existing picture
"L"	connect all computed points with a straight line
"C"	connect all computed points with a smooth curve
"FC"	draw a fill area below a smooth curve



Funções

Pode ser representada como um função:

- ★ Fórmulas (strings)
- ★ C++ functions/functors/lambda's
 - Implemente a função mais complicada que quiser
- ★ Com e sem parâmetros
 - Crucial para ajustes e estimativa de parâmetros

ROOT como um plotador de funções

<https://root.cern.ch/doc/master/classTF1.html>

- ★ A classe TF1 representa funções unidimensionais ($f(x)$):

Declare a pointer to an object of type TF1.
The pointer's name is f1

Name of the function,
Can be nearly anything

Define the function using C-style math
x - is the evaluation variable

```
root [0] TF1 f1("f1","sin(x)/x",0.,10.); //name, formula, min, max
root [1] f1.Draw();
```

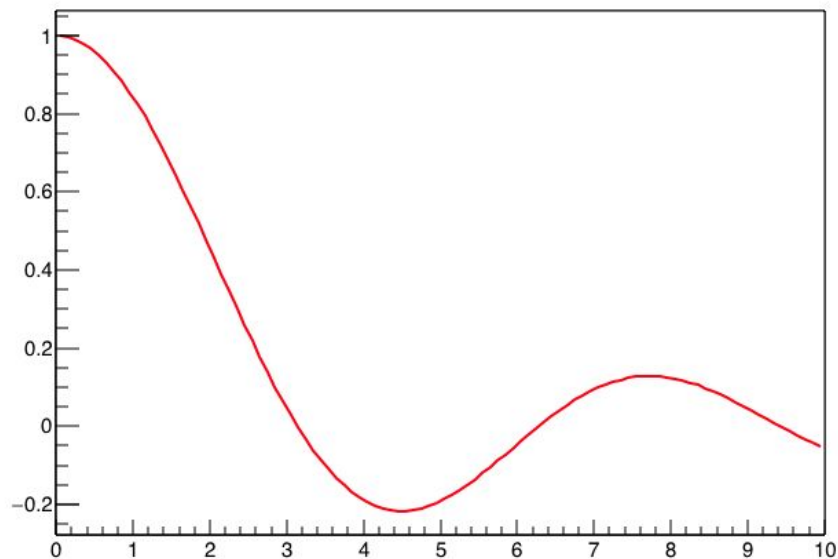
- ★ Também é possível utilizar parâmetros para construir funções definidas:

[0] and [1] - numbers in "[..]" are parameters, and can be set externally.

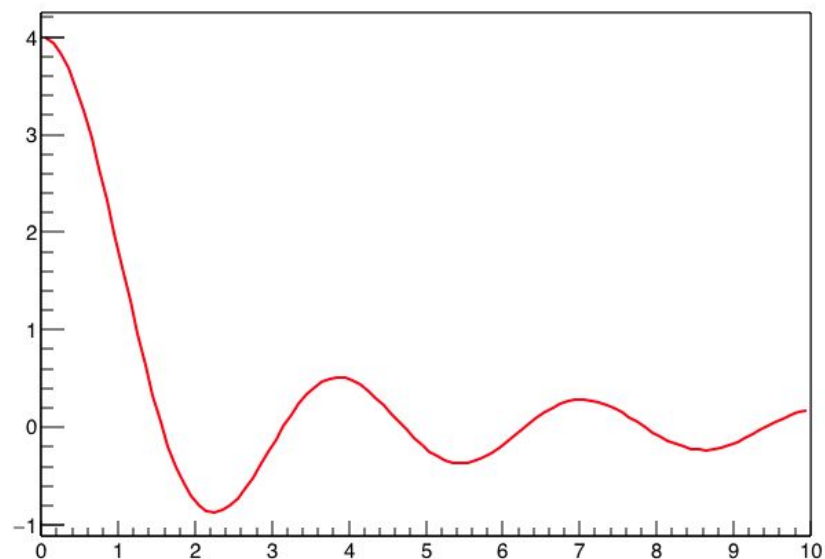
```
root [1] TF1 f2("f2","[0]*sin([1]*(x))/x",0.,10.);
root [2] f2.SetParameters(2,2);
root [3] f2.Draw();
```

ROOT as a function plotter

$\sin(x)/x$



$[0] * \sin([1] * (x)) / x$



ROOT TFile e TTree

ROOT Command Line: Some Objects

Vamos abrir um arquivo ([histograms.root](#)) e ver o que tem dentro

Declarar um ponteiro para o objeto TFile

Cria o objeto, que estará apontando para o arquivo (i.e. abre o arquivo histograms.root)

```
root [0] .ls()
root [1] TFile * input = new TFile("histograms.root");
root [2] input->ls();
```

Use o ponteiro "input" para listar o conteúdo do arquivo com o operador "->" e a função "ls()"

```
TFile**      histograms.root
TFile*       histograms.root
KEY: TH1F     GausHist1d;1    One dimensional Gaussian Distribution
KEY: TH2F     GausHist2d;1    Two dimensional Gaussian Distribution
```

Existem dois histogramas no arquivo. Ambos são objetos cujas propriedades e funções podem ser utilizadas para mostrar os dados

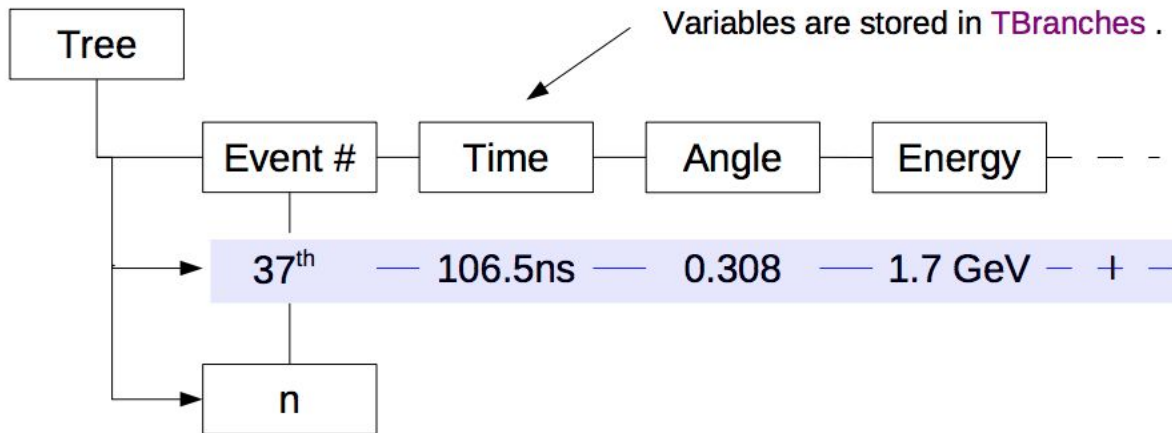
Declare um ponteiro para o objeto TH1F

Aponta para o objeto "GausHist1d" armazenado no arquivo

```
root [3] TH1F * my1Dhist = (TH1F*) input->Get("GausHist1d");
root [4] my1Dhist->Draw();
```


ROOT TTree

- ★ A TTree é uma estrutura de dados para organizar e manipular variáveis de dados de forma uníssona
- ★ Capaz de desenhar histogramas em tempo real, incluindo fazer cortes de seleção nos dados
- ★ Usa algoritmos internos de compressão para reduzir o tamanho dos dados
 - Muito útil para armazenamento de dados



ROOT Command Line: TTree Example

```
root [0] TFile *f1 = new TFile("tree.root");
root [1] f1->ls();
TFile**      tree.root
TFile*       tree.root
KEY: TTree   tree1;1 Reconstruction ntuple
root [2] TTree *mytree = (TTree *)f1->Get("tree1");
root [3] mytree->Print();
*****
*Tree       :tree1       : Reconstruction ntuple
*Entries    : 100000      : Total =      2810673 bytes  File Size =   2171135 *
*           :             : Tree compression factor =   1.30
*****
*Br   0 :event       : event/I
*Entries : 100000     : Total Size=    421248 bytes  File Size =   134514 *
*Baskets : 12        : Basket Size=    32000 bytes  Compression=   2.85 *
*****
*Br   1 :ebeam       : ebeam/F
*Entries : 100000     : Total Size=    421248 bytes  File Size =   260330 *
*Baskets : 12        : Basket Size=    32000 bytes  Compression=   1.47 *
*****
*Br   2 :px          : px/F
*Entries : 100000     : Total Size=    421194 bytes  File Size =   359238 *
*Baskets : 12        : Basket Size=    32000 bytes  Compression=   1.07 *
*****
*Br   3 :py          : py/F
*Entries : 100000     : Total Size=    421194 bytes  File Size =   359138 *
*Baskets : 12        : Basket Size=    32000 bytes  Compression=   1.07 *
*****
```

Cria um ponteiro para "tree1"

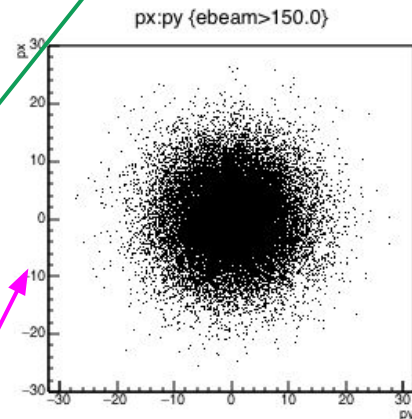
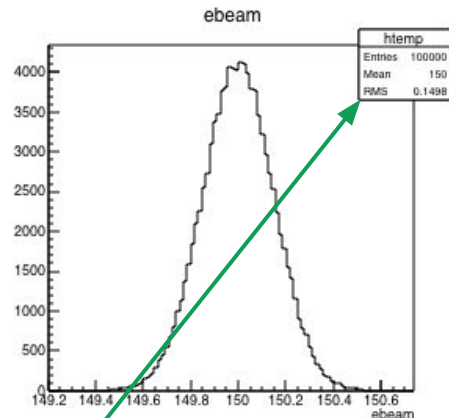
Print a estrutura da tree no prompt
Está tree contém 7 variáveis:

event, ebeam,
px, py, pz, zv,
chi2

Liga a caixa de estatística

```
root [4] TCanvas *c2 = new TCanvas("c2", "Tree canvas", 300, 600);
root [5] c2->Divide(1,2);
root [6] c2->cd(1);
root [7] gStyle->SetOptStat(1);
root [8] mytree->Draw("ebeam");
root [9] c2->cd(2);
root [10] mytree->Draw("px:py", "ebeam>150.0");
```

Draw scatter plot (py vs px) para eventos com ebeam > 150.



ROOT TTree: mais sobre argumentos

- ★ Argumentos para muitas funções de objetos do ROOT são atribuídas por objetos do tipo string
- ★ Strings são analisadas, tanto matematicamente quanto logicamente
- ★ Para trees:
 - Qualquer variável de uma tree pode ser manipulada como parte de um argumento

```
root [10] mytree->Draw("px:py", "ebeam>150.0", "lego");
```

O que plotar do evento

- Dois pontos (":") permite adicionar uma nova variável
- Pode ser usado como/em funções: e.g., "sqrt(py)"
- Pode ser combinação de variáveis: e.g. "ebeam/px : py**2"

- Opções de desenho
- Opções para histogramas n-dimensionais

Cortes de seleção: i.e. quais eventos ou entradas plotar

- Cortes múltiplos são permitidos, combinado com operadores lógico no estilo C++
- Pode ser combinação ou funções de variáveis

ROOT TTree: enchendo um histograma

- ★ Passo 1: Define um histograma com um intervalo adequado

```
root [2] TH1F * h = new TH1F("hBeamEnergy", "Beam Energy", 200, 148.0, 152.0);
```

- ★ Passo 2: Projeta o conteúdo da TTree no histograma

```
root [3] mytree->Project("hBeamEnergy", "ebeam", "px>10.0");
```

Projeta para o histograma **NOME**,
não é um ponteiro

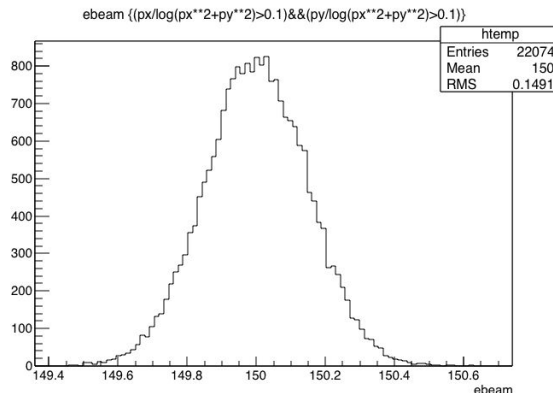
Cortes opcionais

Variável utilizada para preencher o histograma
"projetado".

ROOT TTree: cortes complicados

- ★ Considere encapsular os cortes em objetos TCut
- ★ Objetos TCut podem combinar operadores comumente usados em C
- ★ Eles podem ser combinados com outros cortes do tipo string

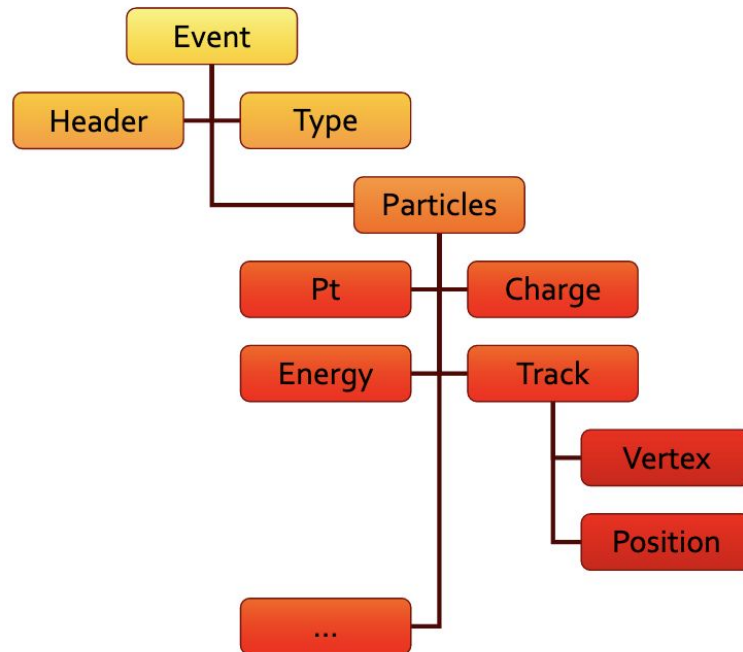
```
root [14] TCut * px_plane = new TCut("px / log(px*2 + py**2) > 0.10");  
root [15] TCut * py_plane = new TCut("py / log(px*2 + py**2) > 0.10");  
root [16] mytree->Draw("ebeam", *px_plane && *py_plane);
```



TNtuple

x	y	z
-1.10228	-1.79939	4.452822
1.867178	-0.59662	3.842313
-0.52418	1.868521	3.766139
-0.38061	0.969128	1.084074
0.552454	-0.21231	0.350281
-0.18495	1.187305	1.443902
0.205643	-0.77015	0.635417
1.079222	-0.32739	1.271904
-0.27492	-1.72143	3.038699
2.047779	-0.06268	4.197329
-0.45868	-1.44322	2.293266
0.304731	-0.88464	0.875442
-0.71234	-0.22239	0.556881
-0.27187	1.181767	1.470484
0.886202	-0.65411	1.213209
-2.03555	0.527648	4.421883
-1.45905	-0.464	2.344113
1.230661	-0.00565	1.514559
		3.562347

TTree



Exercício

1. Crie uma função $\sin(x)/x$ do tipo TF1 e plote-a. Crie uma função $p0 * \sin(p1 * x) / x$, com parâmetros $p0$ e $p1$, e também plote-a para diferentes valores. Defina a cor da função paramétrica para azul. Depois de desenhar a função, calcule para os valores dos parâmetros ($p0 = 1$, $p1 = 2$):
 - a. valor da função para $x = 1$ (Dica: Use o método Eval)
 - b. derivada de função para $x = 1$
 - c. integral da função entre 0 e 3
2. Utilizando o conjunto de pontos presente no arquivo [graphdata.txt](#).
 - a. Plote esses pontos usando a classe TGraph. Use como marcador para cada ponto uma caixa verde.
 - b. Consultando as opções de desenho para o TGraph no [TGraphPainter](#), plote uma linha conectando os pontos.
3. Crie um histograma com 50 bins entre 0 a 10 e preencha-o com 10.000 números aleatórios distribuídos segundo uma distribuição gaussiana com média 5 e sigma 2. Plote o histograma e, olhando a documentação [THistPainter](#), mostre na caixa de estatísticas o número de entradas, a média, o RMS, a integral do histograma.
4. Usando a TTree [tree.root](#), faça a distribuição do momentum total de todas as entradas cuja energia do feixe esteja fora do valor médio em mais de 0,2. Use objetos do tipo TCut para a seleção de eventos. Projete essa distribuição em um histograma, desenhe e salve em um arquivo.

A solução deve ser adicionada no GitHub e o link para eles no AVA!

Referências adicionais

★ http://webhome.phy.duke.edu/~raw22/public/root.tutorial/basic_root_20100701.pdf

★ PyROOT

- https://docs.google.com/presentation/d/1nNFRdh483KSYnoaA6q7x0nVeDPbhY7gjWyaGmr0ZdTA/edit#slide=id.g2a0483ea55_3_300

★ Link dos notebooks utilizados na aula de hoje

- https://root.cern/doc/master/group_tutorial_pyroot.html
- <https://github.com/Analise-Dados-FAE/2021>