División de Rayos Cósmicos (DRC)

Introducción a ROOT

Reunión anual de Rayos Cósmicos

Eduardo Moreno Barbosa

FCFM BUAP

emoreno@fcfm.buap.mx

Ligas de root

- ROOT es un software de código abierto
  - https://cds.cern.ch/record/491486/files/p11.pdf
- Pagina principal: https://root.cern.ch/
- Guía de referencia:
  - https://root.cern.ch/doc/master/index.html
  - https://root.cern.ch/root/htmldoc/guides/user-guide/ROOTUsersGuide.html#getting-started
- Manual: https://root.cern/get_started/
- Tutoriales: https://root.cern/get_started/courses/

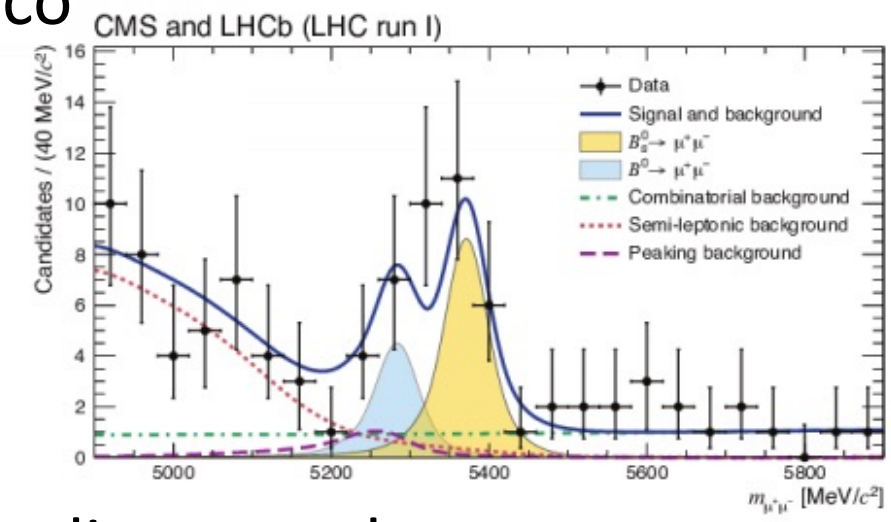
Como instalar root

- Hoy en día hay diversas maneras de realizar la instalación
  - https://root.cern/install/
- Checar las dependencias acorde al sistema operativo y distribución de linux.
  - https://root.cern/install/dependencies/
- Recomiendo realizar la instalación construyendo ROOT desde la fuente.
  - https://root.cern/install/#build-from-source

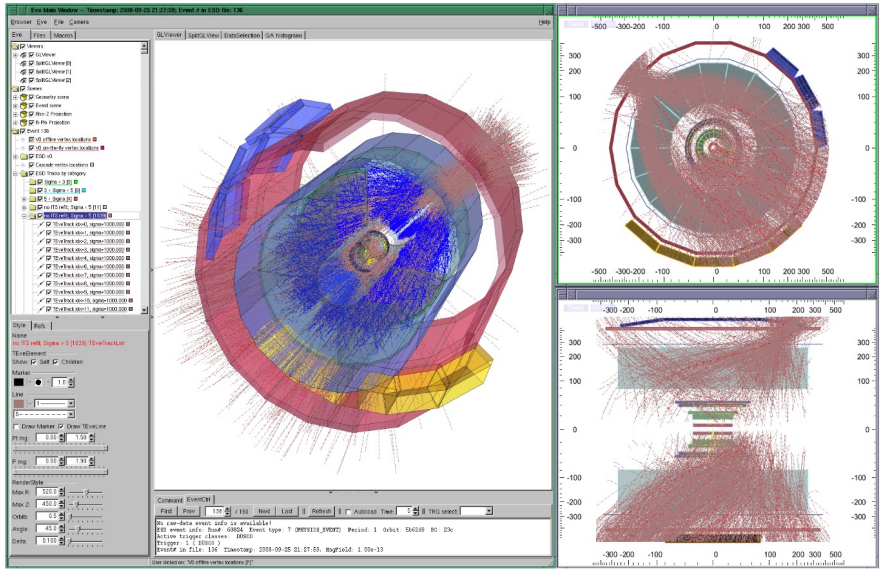
```
$ git clone --branch latest-stable --depth=1 https://github.com/root-project/root.git root_src
$ mkdir root_build root_install && cd root_build
$ cmake -DCMAKE_INSTALL_PREFIX=../root_install ../root_src # && check cmake configuration outpu
$ cmake --build . -- install -j4 # if you have 4 cores available for compilation
$ source ../root_install/bin/thisroot.sh # or thisroot.{fish,csh}
```

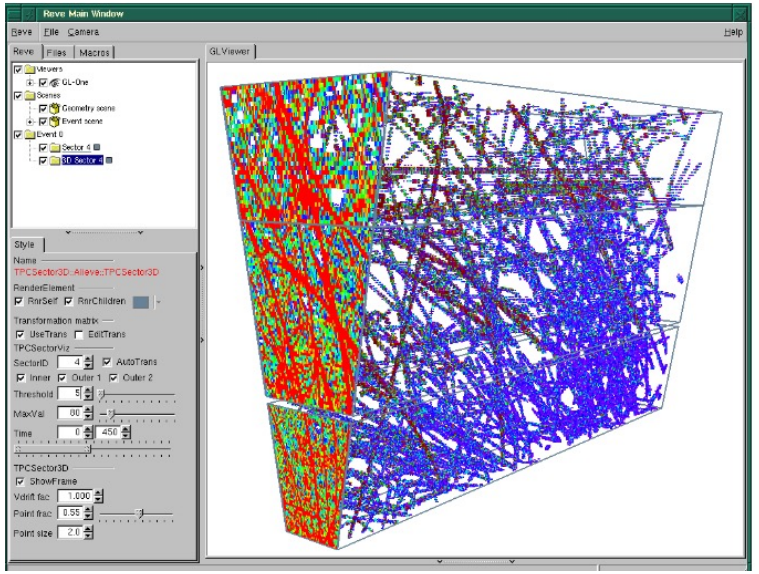
¿Qué es ROOT?

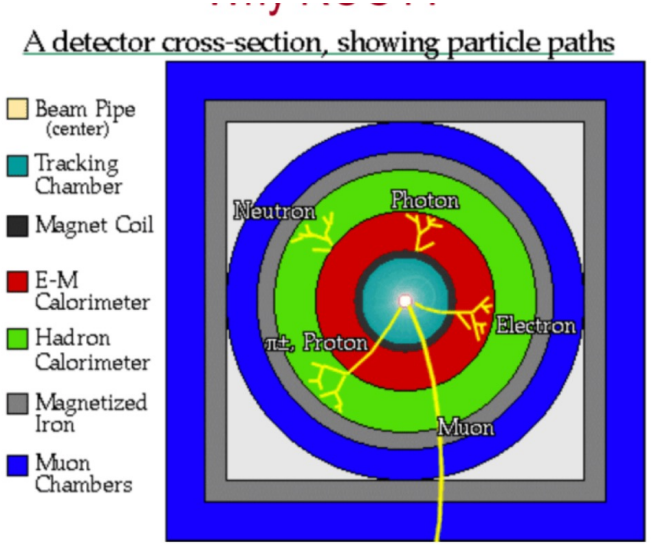
- Es un framework modular de software científico
- Su uso más común es en análisis de datos en:
  - Altas energías
  - Física nuclear
  - Física medica
  - Detectores

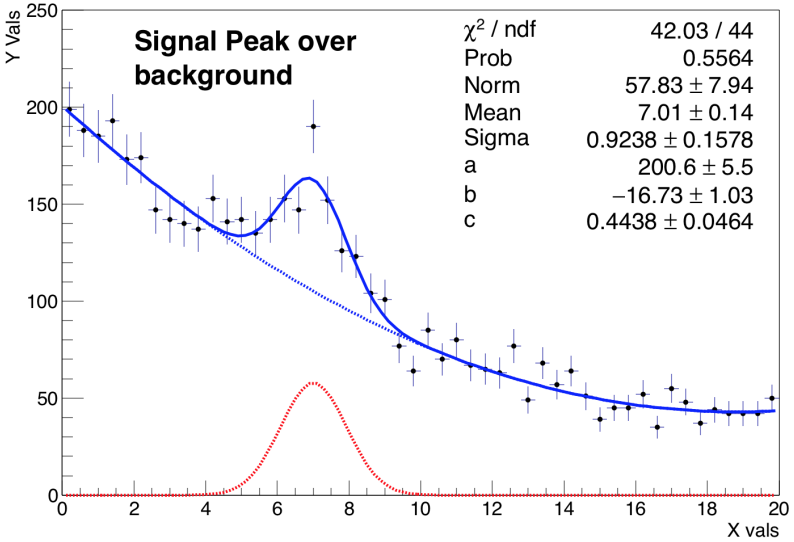


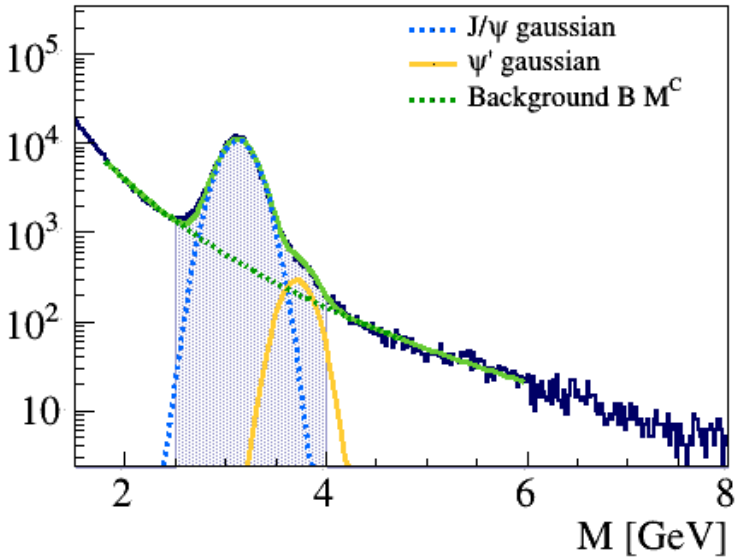
- Es una herramienta dinámica y que permite analizar grandes cantidades de datos
- Permite la representación gráfica de datos, métodos de análisis, ajustes, etc.

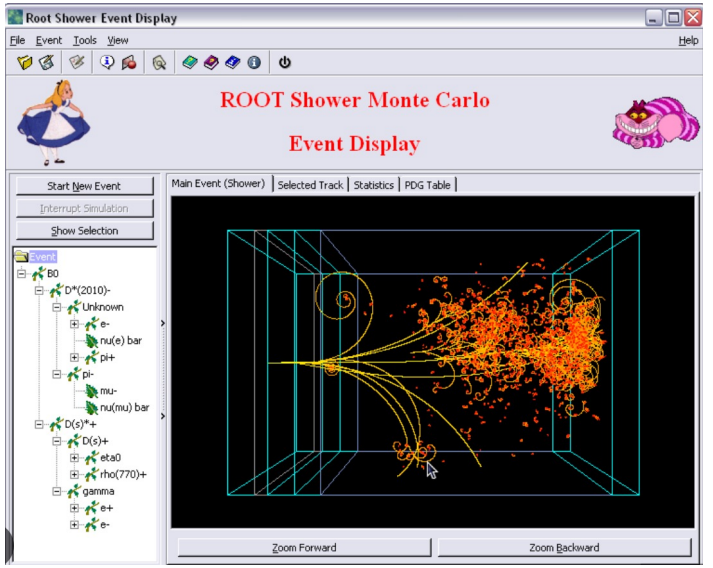





A detector cross-section, showing particle paths
Beam Pipe (center)
Tracking Chamber
Magnet Coil
E-M Calorimeter
Hadron Calorimeter
Magnetized Iron
Muon Chambers
Neutron
Photon
Electron
π±, Proton
Muon


Signal Peak over background
χ² / ndf 42.03 / 44
Prob 0.5564
Norm 57.83 ± 7.94
Mean 7.01 ± 0.14
Sigma 0.9238 ± 0.1578
a 200.6 ± 5.5
b −16.73 ± 1.03
c 0.4438 ± 0.0464
Y Vals
X vals


J/ψ gaussian
ψ' gaussian
Background B M^C
M [GeV]


Root Shower Event Display
ROOT Shower Monte Carlo
Event Display
Start New Event
Interrupt Simulation
Show Selection
Main Event (Shower)
Selected Track
Statistics
PDG Table
Zoom Forward
Zoom Backward

Como usar ROOT

```
(base) barbosa@barbosa tree % root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.26/10                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosx64 on Mar 26 2023, 09:40:00                      |
  | From heads/latest-stable@4dddea35                                |
  | With Apple clang version 12.0.5 (clang-1205.0.22.9)              |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] Float_t valor=4.5
(float) 4.50000f
root [1] cout << "valor de calculo: " << valor << endl;
valor de calculo: 4.5
root [2] .q
(base) barbosa@barbosa tree %
```

- Mediante la ejecución de comandos en el ambiente de root y programas en c++
- root tiene un interprete de c++: CINT (root 5) CLING (root 6)
- En una terminal se puede usar desde línea de comandos y el interprete ejecuta línea por línea.
- Ventaja - El resultado se visualiza de manera inmediata
- Desventaja - Una interpretación de comando siempre es mas lenta que ejecutar código compilado

Como usar ROOT

```
(base) barbosa@barbosa tmp % more macro.C

{
Float_t valor=4.5;
cout << "valor de calculo: " << valor << endl;
}
(base) barbosa@barbosa tmp % root macro.C -l -q

Processing macro.C...
valor de calculo: 4.5
(base) barbosa@barbosa tmp %
```

```
(base) barbosa@barbosa tmp % more macro.C

void macro(int n){
Float_t valor=4.5;
cout << "valor de calculo: " << valor*n << endl;
}
(base) barbosa@barbosa tmp % root -l
root [0] .L macro.C
root [1] macro(2)
valor de calculo: 9
root [2]
```

- Mediante el uso de archivos de texto que contienen el conjunto de instrucciones a ejecutar denominados macros.
- El macro puede tener la extensión C, cxx o cc
- El interprete CINT/CLING ejecuta línea por línea del macro.
- Dentro del macro cada línea se termina con ; (código C++).
- El macro se puede ejecutar sin pasarle parámetros
- El macro se puede ejecutar pasándole parámetros

Como usar ROOT

```
(base) barbosa@barbosa curso % cat funcion3.C
#include <iostream>

void funcion3(int n){
Float_t valor=2.5;
cout << "valor multiplicado por una constante: " << valor*n << endl;
}
(base) barbosa@barbosa curso % root -l
root [0] .L funcion3.C++
Info in <TMacOSXSystem::ACLiC>: creating shared library /Users/barbos
ld: warning: dylib (/usr/local/Cellar/zstd/1.5.4/lib/libzstd.1.dylib)
ld: warning: dylib (/usr/local/Cellar/lz4/1.9.4/lib/liblz4.1.dylib) w
ld: warning: dylib (/usr/local/Cellar/lz4/1.9.4/lib/liblz4.1.9.4.dyli
ld: warning: dylib (/usr/local/Cellar/zstd/1.5.4/lib/libzstd.1.5.4.dy
ld: warning: dylib (/usr/local/lib/liblzma.5.dylib) was built for new
ld: warning: dylib (/usr/local/Cellar/zstd/1.5.4/lib/libzstd.1.dylib)
ld: warning: dylib (/usr/local/Cellar/lz4/1.9.4/lib/liblz4.1.dylib) w
ld: warning: dylib (/usr/local/Cellar/lz4/1.9.4/lib/liblz4.1.9.4.dyli
ld: warning: dylib (/usr/local/Cellar/zstd/1.5.4/lib/libzstd.1.5.4.dy
root [1] funcion3(1)
valor multiplicado por una constante: 2.5
```

- Macros compilados
- Los códigos compilados son más rápidos
- En programas más grandes o lentos archivos .C o .cc pueden compilarse
- La compilación se realiza por medio de ACLiC (The Automatic Compiler of Libraries for CINT)
- .L macro.C++ //Carga y compila
- macro(10) // llamada al macro
- .x macro.C(10)++ //carga, compila y ejecuta

Como usar ROOT

```
(base) barbosa@barbosa ejecutables % more primer.c++
#include <iostream>
#include "TH1.h"
#include "TH1D.h"
#include "TFile.h"

int main(){
  TH1D *hRawMet=new TH1D("Met","Met",50,0.,200.);
  hRawMet->Fill(50.);
  TFile f("output.root","recreate");
  hRawMet->Write();
  f.Close();
  return 0;
}
```

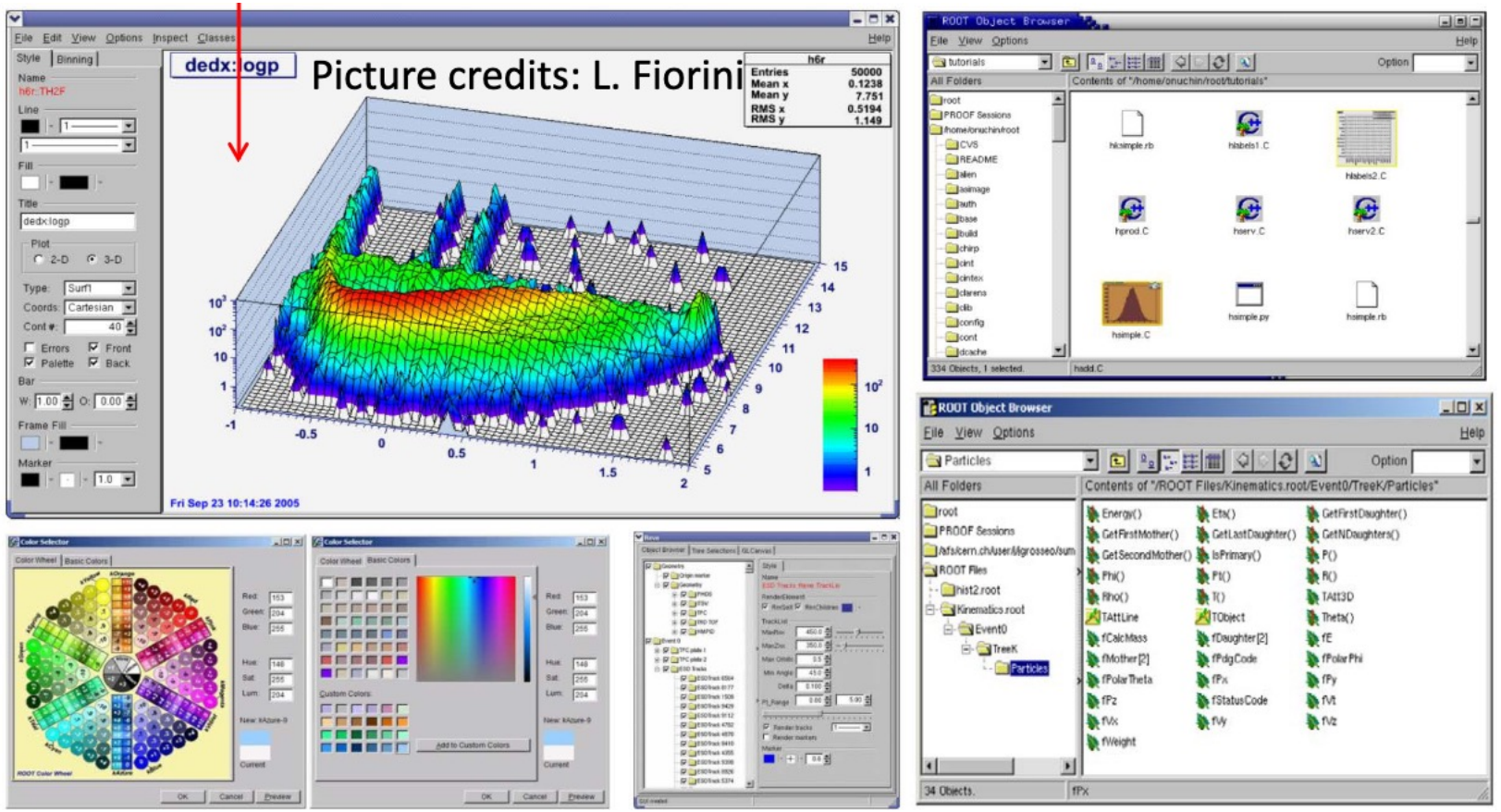
```
g++ -o primero primer.c++ `root-config --cflags --glibs`
./primero
```

```
void datos(){
  comandos;
  ..
  return;
}
#Ifndef__CINT_

_
Int main(){
Datos();
Return 0;
}
#endif
```

- Aplicación Stand-alone
- Es una forma elegante de programar
- Como parte de semántica de c++ se debe incluir una función main()
- El programa principal de compilación es gcc o g++ y no CINT
- Los programas son independientes de la versión de root una vez credo el archivo ejecutable
- Compilación
- g++ -o salida salida.cc `root –config –cflags --glibs`

Como usar ROOT

- Interfaz graficas
- ROOT object browser (Tbrowser)
- Graphical use-interface (GUI)
- Editor
- Fit Panel



Como usar ROOT

- Jupyter notebook
- Uso de jupyter para ejecutar comandos de root usando pyroot
- https://root.cern.ch/notebooks/HowTos/HowTo_ROOT-Notebooks.html

In [1]:
```
import ROOT
```
```
Welcome to JupyROOT 6.18/00
```

Now we are ready to use PyROOT. For example, we create a histogram.

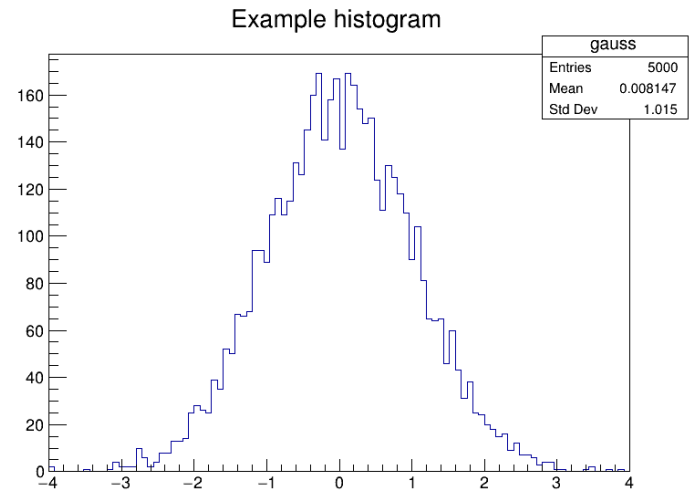
In [2]:
```
h = ROOT.TH1F("gauss","Example histogram",100,-4,4)
h.FillRandom("gaus")
```

Next we create a canvas, the entity which holds graphics primitives in ROOT.

In [3]:
```
c = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
h.Draw()
```

For the histogram to be displayed in the notebook, we need to draw the canvas.

In [4]:
```
c.Draw()
```



In [6]:
```
%%cpp
cout << "This is a C++ cell" << endl;
```
```
This is a C++ cell
```

Not bad. On the other hand, ROOT offers much more than this. Thanks to its type system, entities such as functions, classes and variables, created in a C++ cell, can be accessed from within Python.

In [7]:
```
%%cpp
class A{
    public:
    A(){cout << "Constructor of A!" << endl;}
};
```

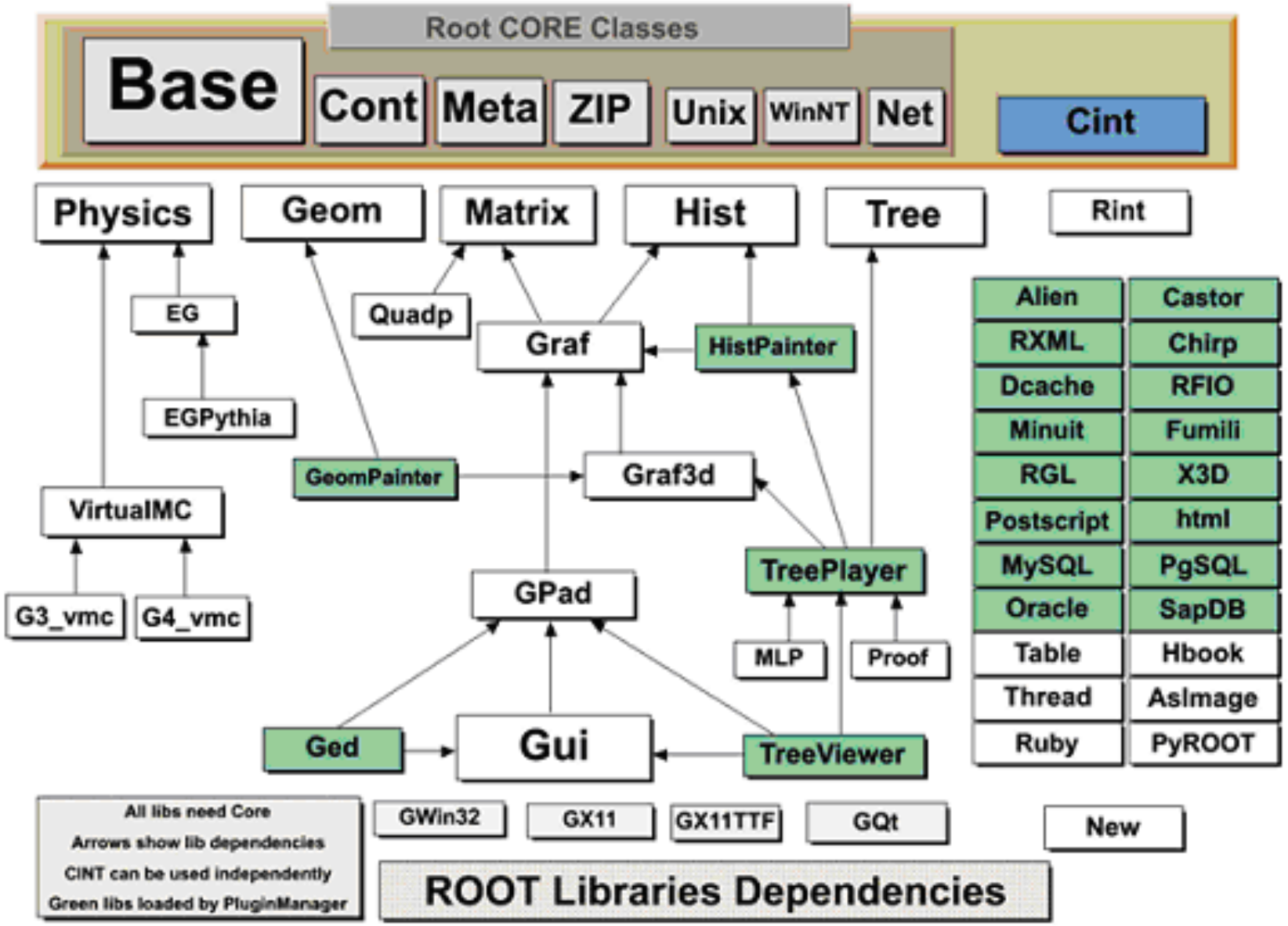
In [8]:
```
a = ROOT.A()
```
```
Constructor of A!
```

The Python and C++ worlds are so entangled that we can find back in C++ the entities created in Python. To illustrate this, from within a C++ cell, we are going to fit a function in the `gauss` histogram displayed above and then re-draw the canvas.

In [9]:
```
%%cpp
gauss->Fit("gaus", "S");
myCanvasName->Draw();
```
```
 FCN=65.0502 FROM MIGRAD    STATUS=CONVERGED      54 CALLS          55 TOTAL
                     EDM=9.21369e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     1.57425e+02   2.76819e+00   8.89596e-03  -5.39870e-04
   2  Mean         1.38321e-02   1.43799e-02   5.69927e-05   8.22299e-04
   3  Sigma        1.00178e+00   1.04857e-02   1.11224e-05  -4.31183e-01
```

Librerias

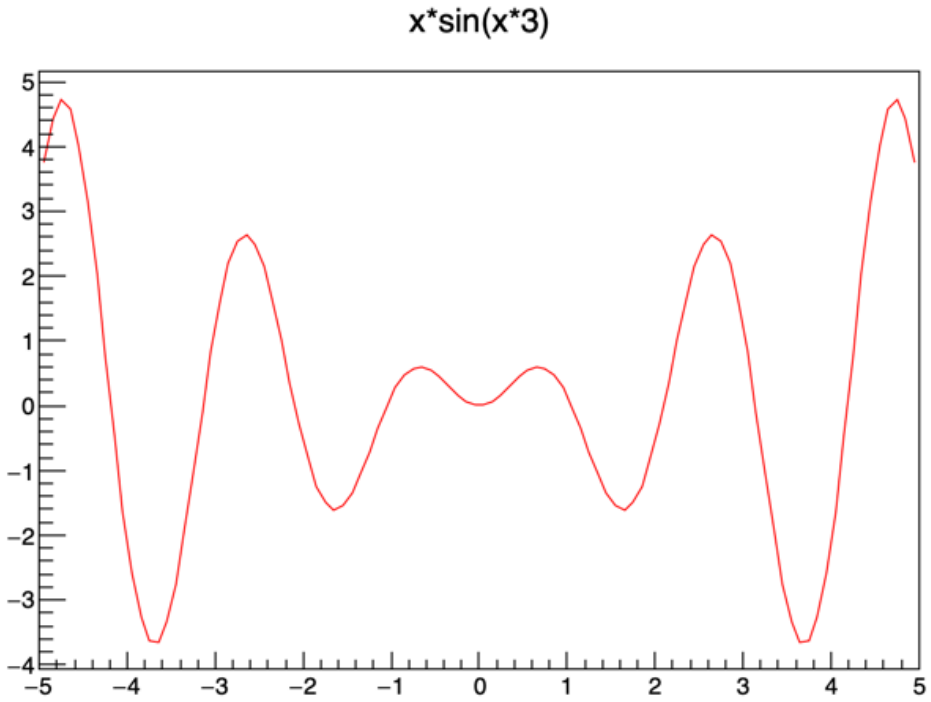
- Este curso se centra en los siguientes objetos
- Funciones (TF)
- Histogramas (TH)
- Ajustes (Fit)
- Arboles (TTree)
- PyROOT



Funciones

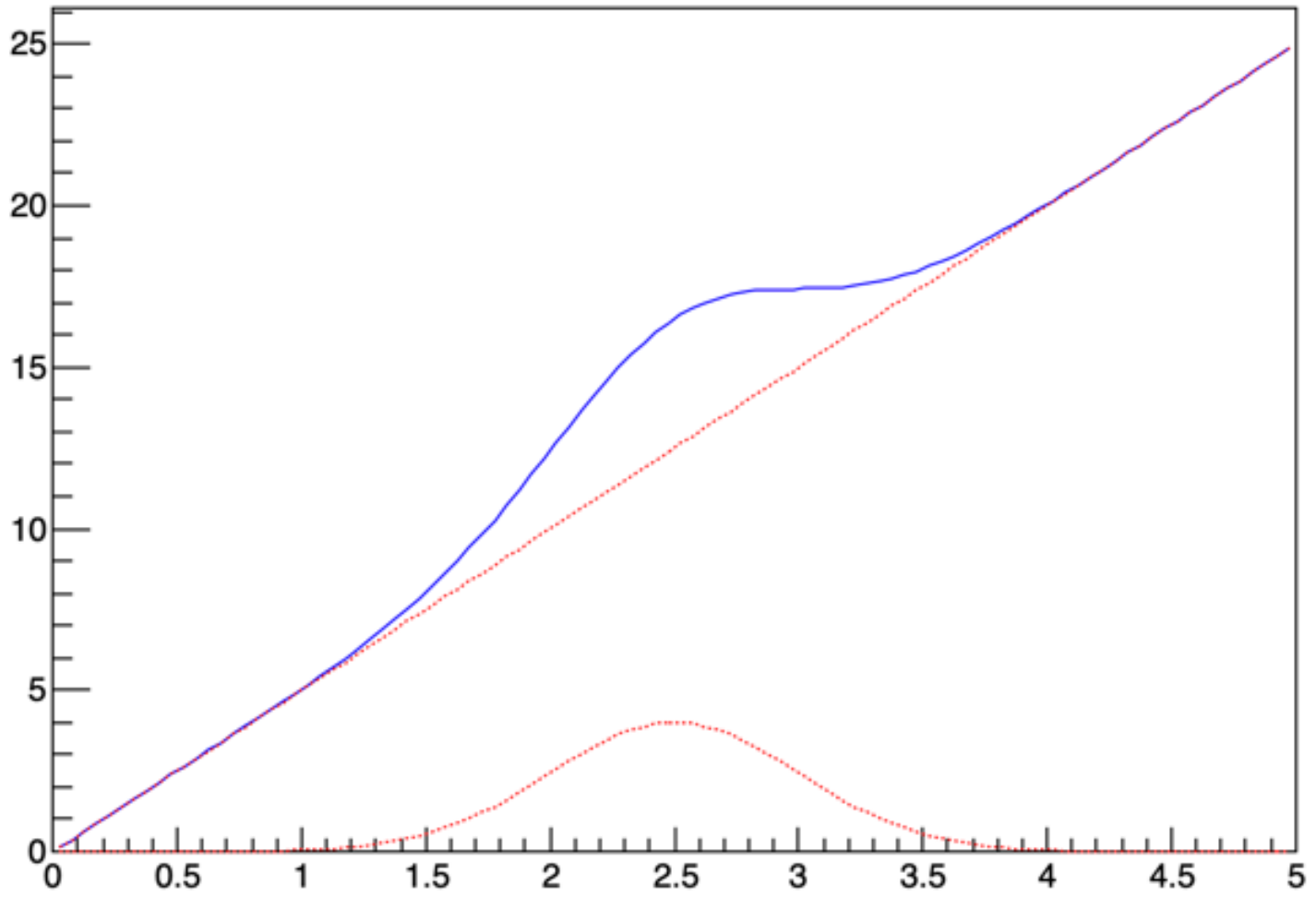
- Regla de correspondencia o asociación, en donde a cada valor de x le asocia una valor y
- Se utiliza la clase TF https://root.cern.ch/doc/master/group__Functions.html
- TFn n={1,2,3} para crear funciones en 1D, 2D y 3D

- root [ ] TF1 f1("f1",x*sin(x*3),-5,5)
- root [ ] TF1 * f1 = new TF1("f1",x*sin(x),-5,5)
  - "f1" nombre de la función
  - "x*sin(x*3)" formula
  - -5,5 rango de la función
- root[ ] f1.Draw() o f1->Draw()
- Root [] f1.SetlineColor(kRed)



Funciones

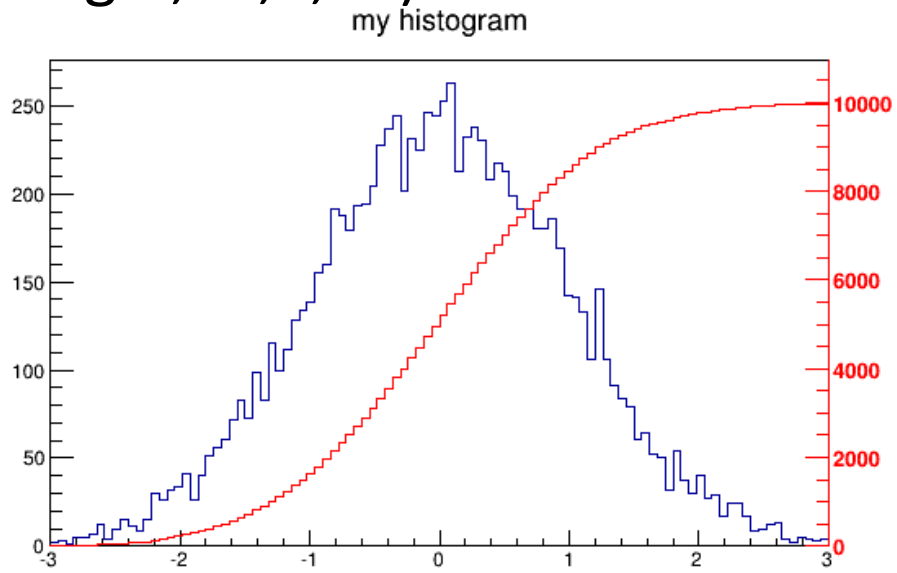
```
void funciones2(){
TF1 *f1 = new TF1("f1","5*x",0,5);
TF1 *f2 = new TF1("f2","sin(x)",0,5);
TF1 *f3 = new TF1("f3","gaus",0,5);

f3->SetParameter(0,4); //Amplitud
f3->SetParameter(1,2.5); //media
f3->SetParameter(2,0.5); //sigma

TF1 *f4 = new TF1("f4","f3+f1",0,5);
TF1 *f5 = new TF1("f5","f3(f1)",0,5);

f4->Draw();
f1->Draw("SAME");
f3->Draw("SAME");
}
```



Histogramas

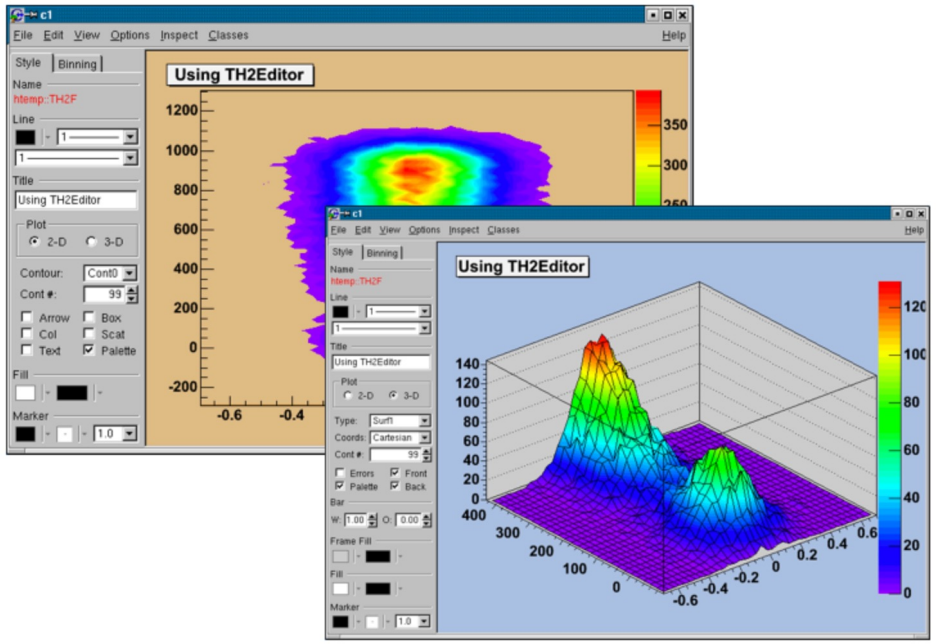
- Representación gráfica de un conjunto de datos organizado en rangos específicos.
- Se utiliza la clase TH https://root.cern.ch/doc/master/group__Histograms.html
- THnt
  - n={1,2,3} para crear funciones en 1D, 2D y 3D
  - t={F,C,D,I,S} tipo de datos (flotante, byte, doble, entero, short) por subintervalo (bin)
- root [ ] TH1F h1("h1","Histograma de carga",20,5,10)
- root [ ] TH1F * h1 = new TH1F("h1","Histograma de carga",20,5,10)
  - "f1" - nombre del histograma
  - "Histograma de carga" - Título del histograma
  - 20 - número de subintervalos (bin)
  - 5,10 - Rango de datos
- root[ ] h1.Draw() o h1->Draw()
- Root [] h1.SetlineColor(kBlue)

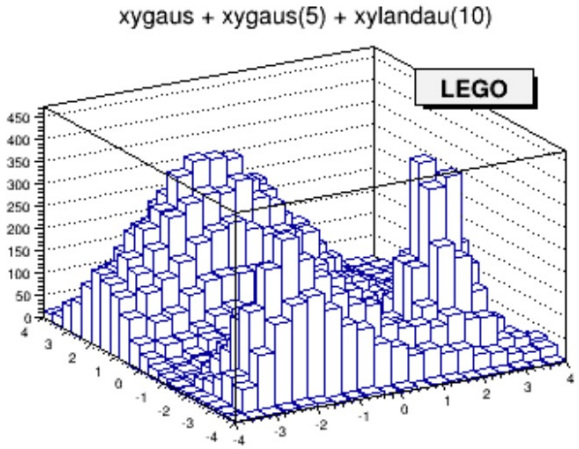
https://doi.org/10.1016/j.edurev.2019.100291

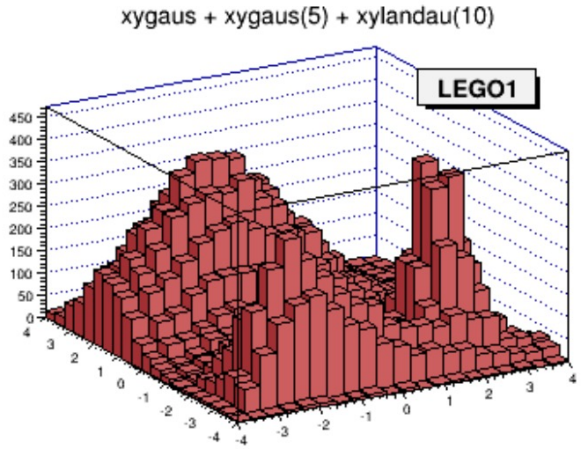


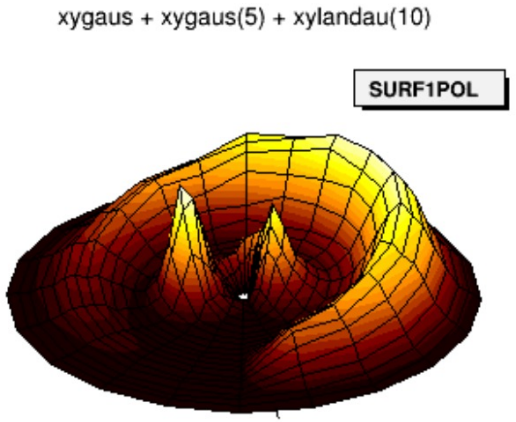
Histogramas 2D

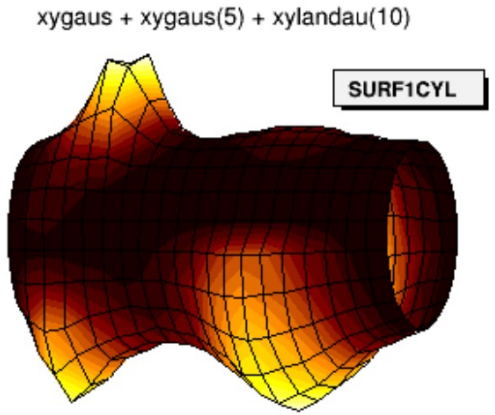
- `TH2D h2d("h2d","Histograma bidimensional",10,-1,1,10,-1,1)`
- `gStyle()->SetPalette(1)`
- `H2d->Draw("colz")`
- `H2d-Draw("lego")`





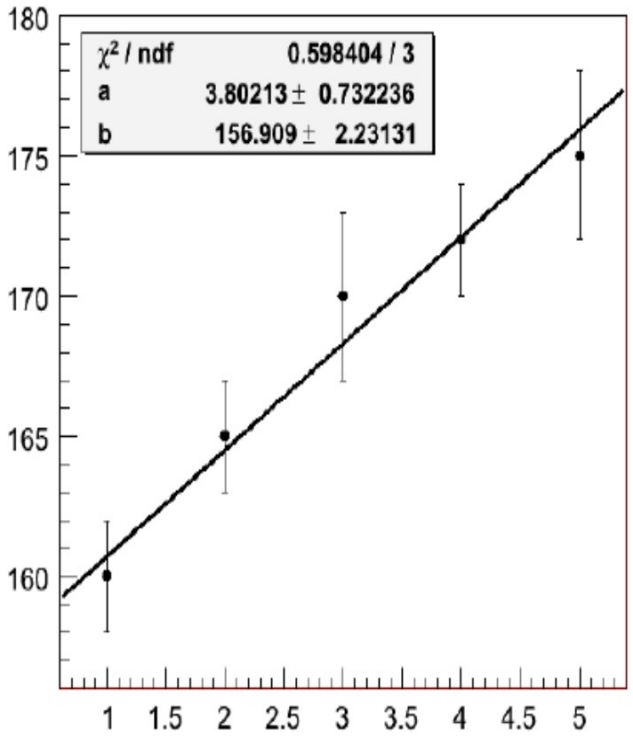




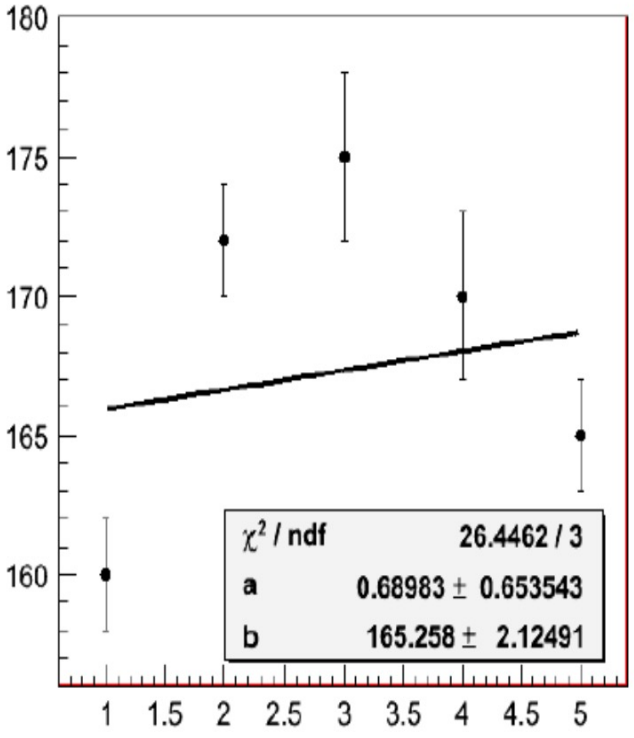


Ajustes (Fit)

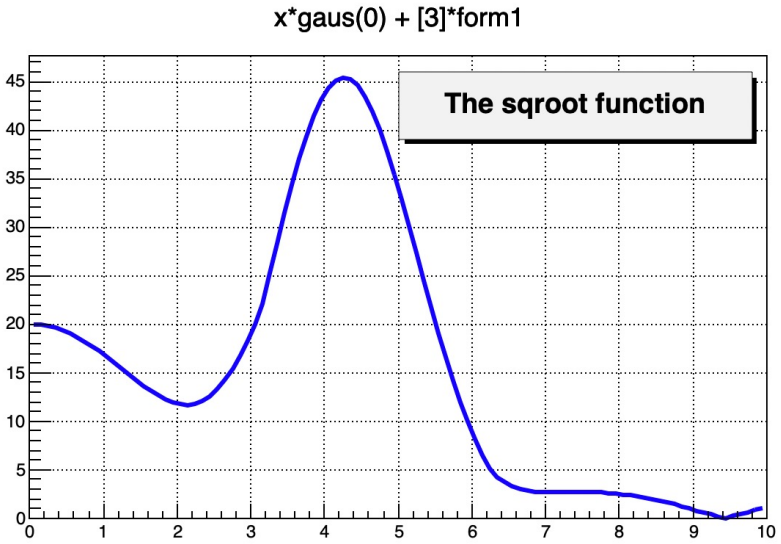
- Es procedimeinto para comparar un conjunto de datos respecto a una función y determina cuales son los mejores parámetros..
- Se utiliza la clase FiT https://root.cern.ch/root/htmldoc/guides/users-guide/FittingHistograms.html
- TFit
  - n={1,2,3} para crear ajustes en 1D, 2D y 3D
- El valor /ndf es un indicador de la calidad del ajuste
- **ndf** es el número de puntos menos el número de parámetros e la función y se denomina número de grados d libertad

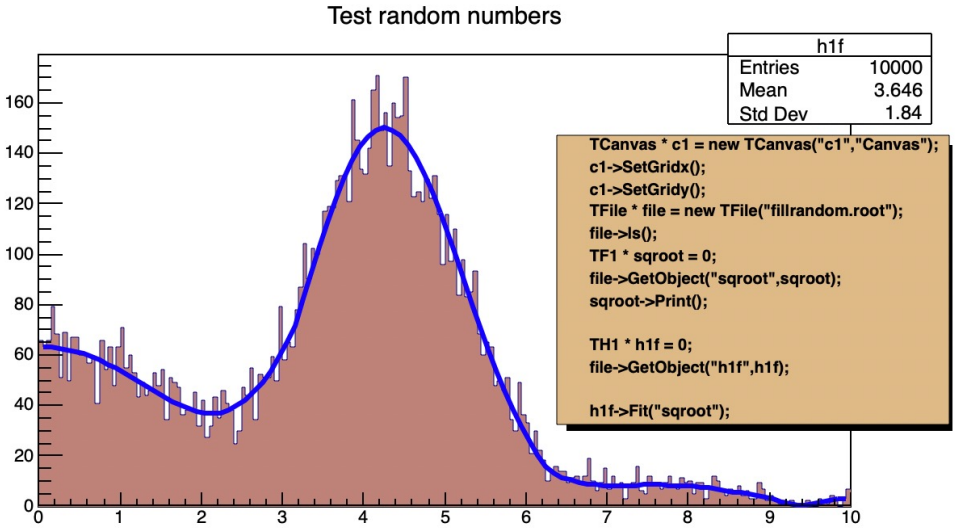


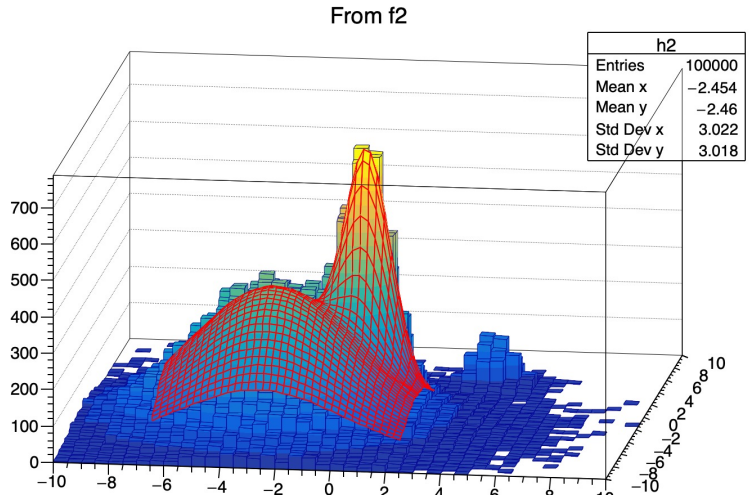
$$\chi^2 = \sum_i \frac{(y_i - f(x_i))^2}{(\Delta y_i)^2}$$



Ajustes

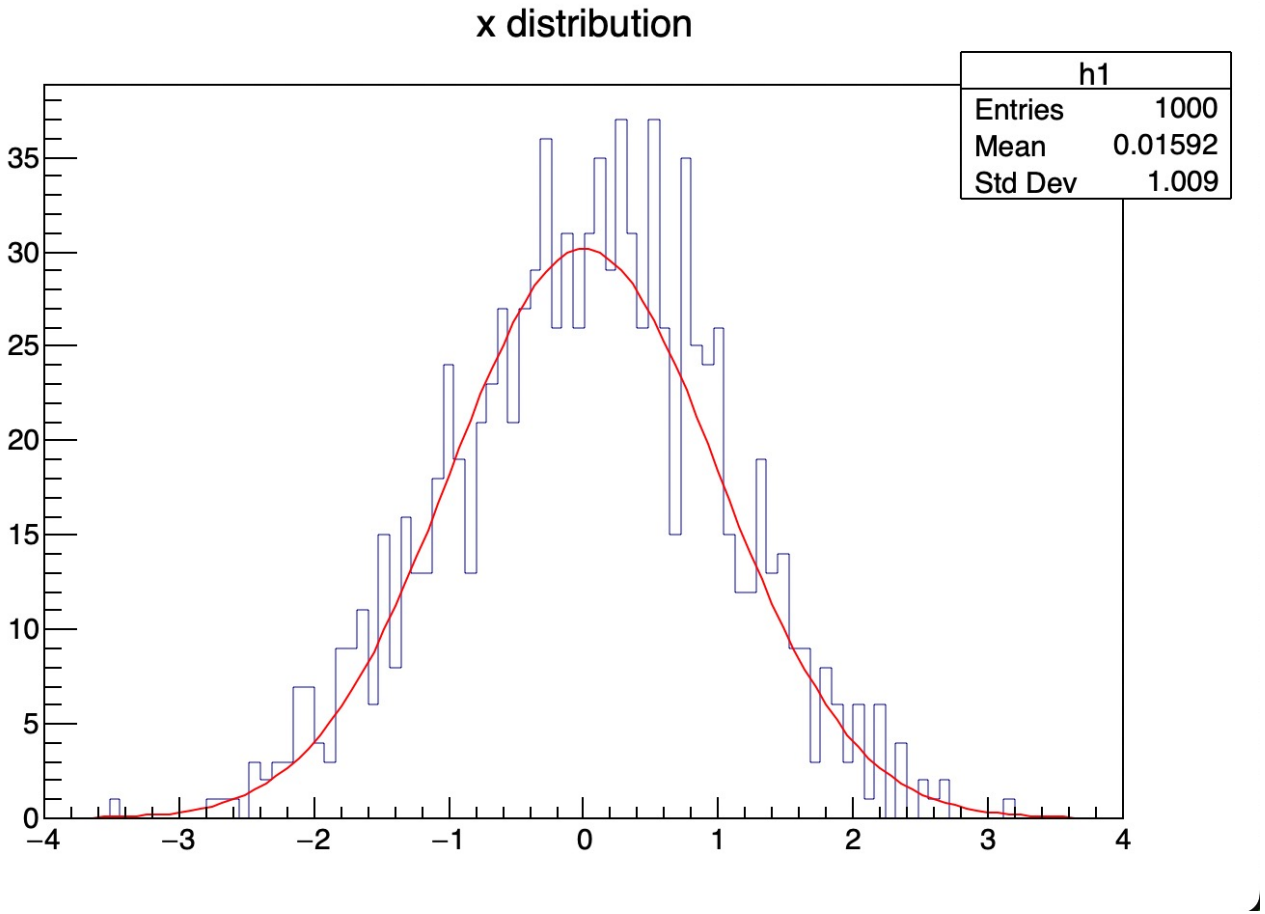

x*gaus(0) + [3]*form1
The sqroot function

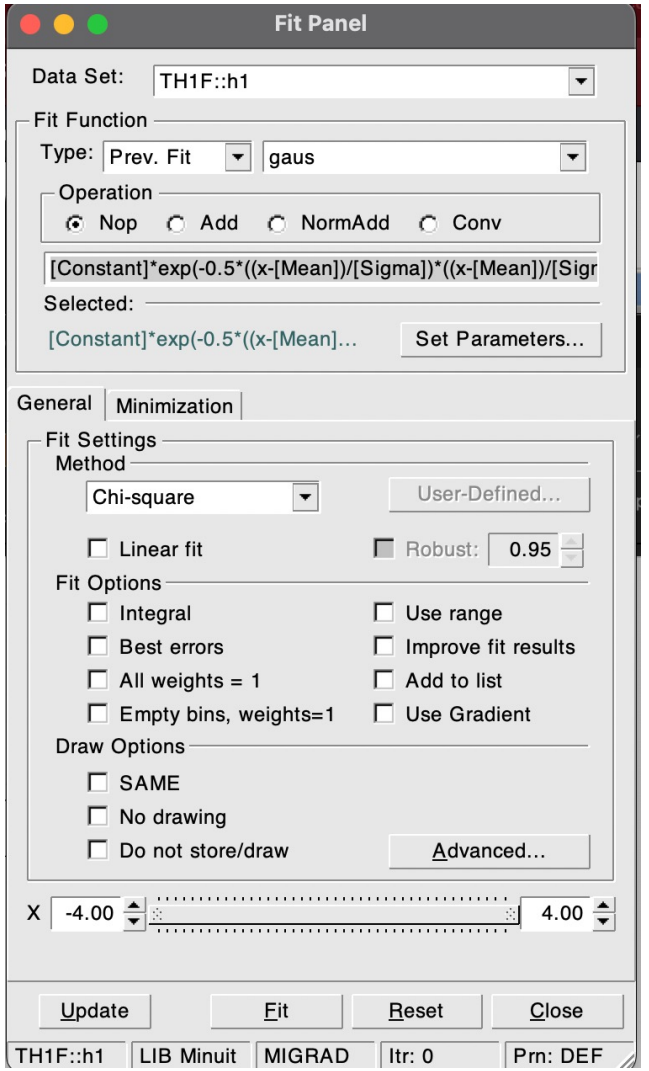

Test random numbers
h1f
Entries 10000
Mean 3.646
Std Dev 1.84
TCanvas * c1 = new TCanvas("c1","Canvas");
c1->SetGridx();
c1->SetGridy();
TFile * file = new TFile("fillrandom.root");
file->ls();
TF1 * sqroot = 0;
file->GetObject("sqroot",sqroot);
sqroot->Print();
TH1 * h1f = 0;
file->GetObject("h1f",h1f);
h1f->Fit("sqroot");


From f2
h2
Entries 100000
Mean x −2.454
Mean y −2.46
Std Dev x 3.022
Std Dev y 3.018

Ajustes

- Además del uso de comandos en el ambiente de root y macros, se puede utilizar el FitPanel una herramienta interactiva
- Existen funciones predefinidas pero es posible utilizar funciones definidas por el usuario

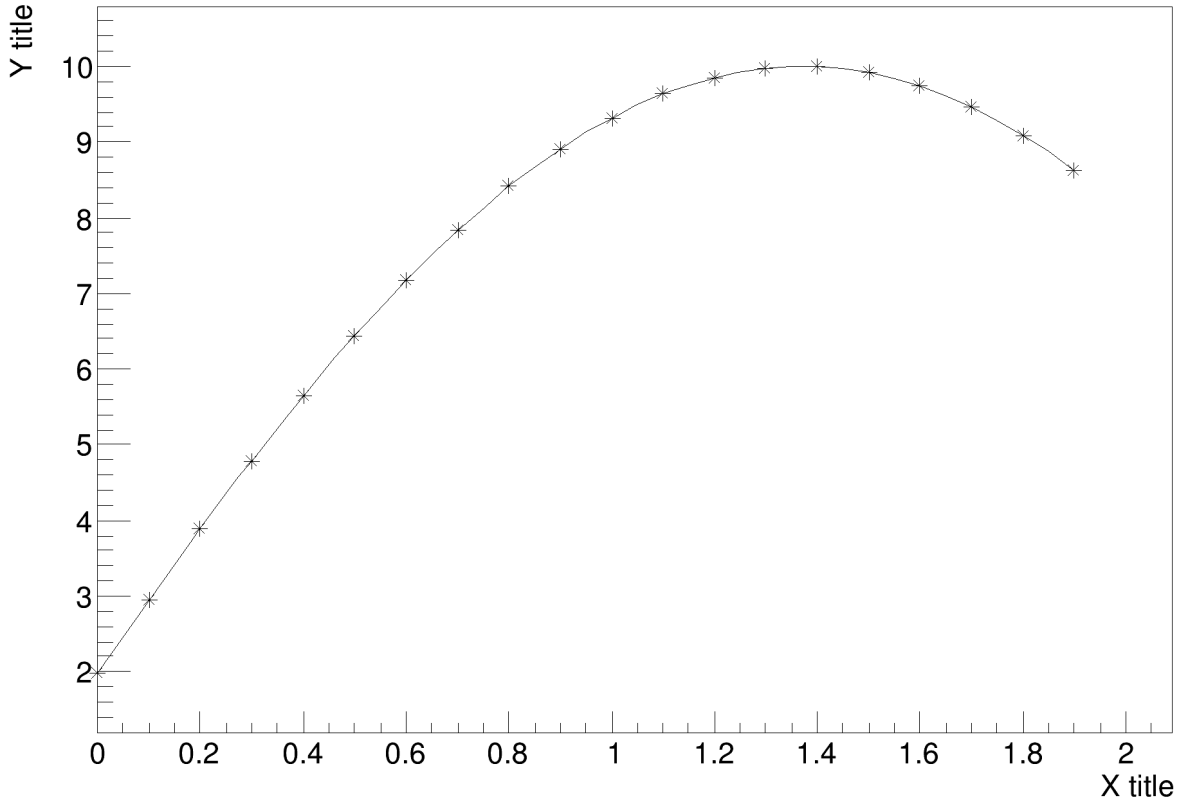




Graficos

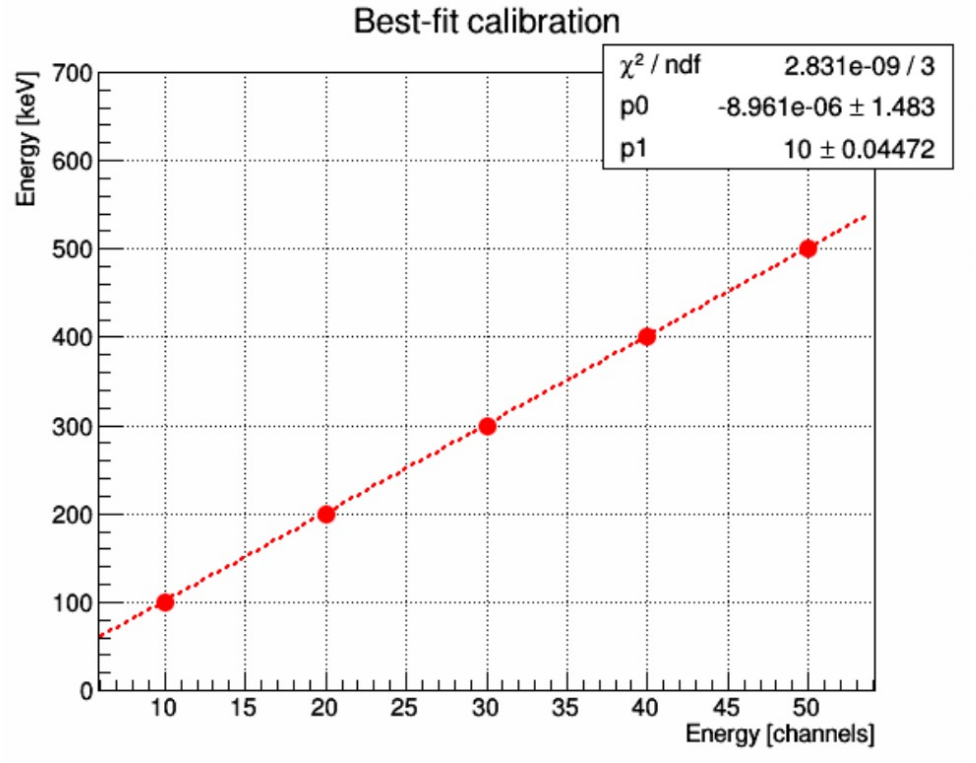
- Es una representación de valores de una variable en función de otra variable.
- Se utiliza la clase Tgraph https://root.cern.ch/doc/master/classTGraph.html

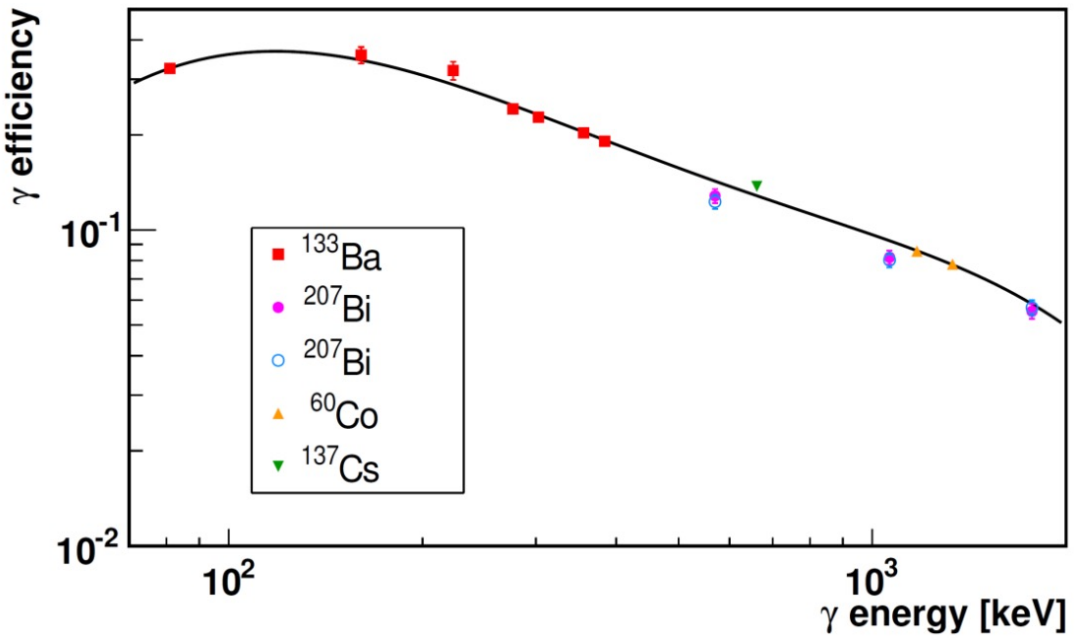
```
double x[100], y[100];
int n = 20;
for (int i=0;i<n;i++) {
x[i] = i*0.1;
y[i] = 10*sin(x[i]+0.2);
}
auto g = new TGraph(n,x,y);
g->SetTitle("Graph title;X title;Y
title");
g->Draw("AC*");
```



Ajuste de graficos

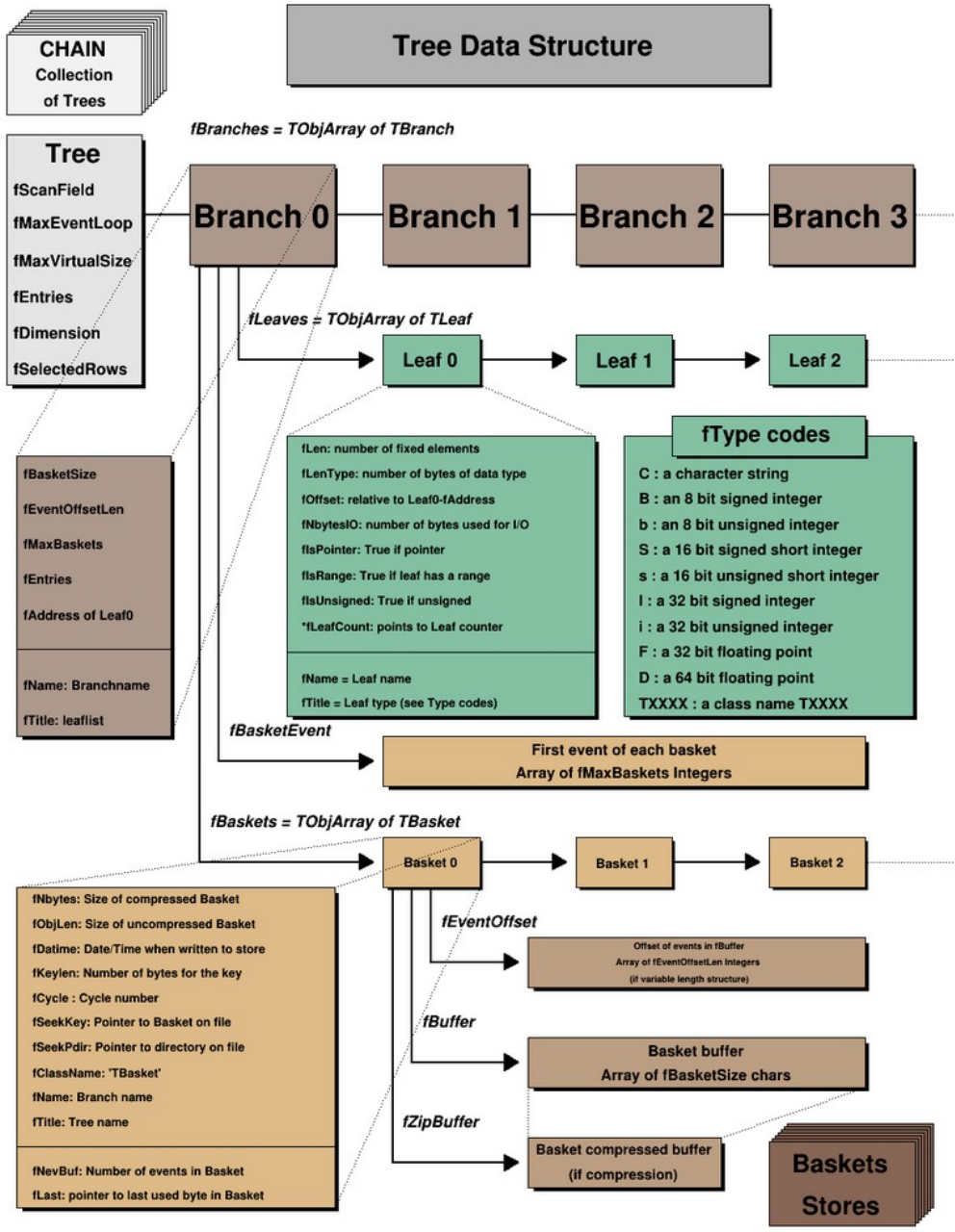
```
TF1 *bestfit = new TF1("bestfit","pol1",x1,x2)
GraphErrors->Fit("bestfit","R")
//Getting χ² and ndf
root [ ] bestfit->GetChisquare()
root [ ] bestfit->GetNDF()
```





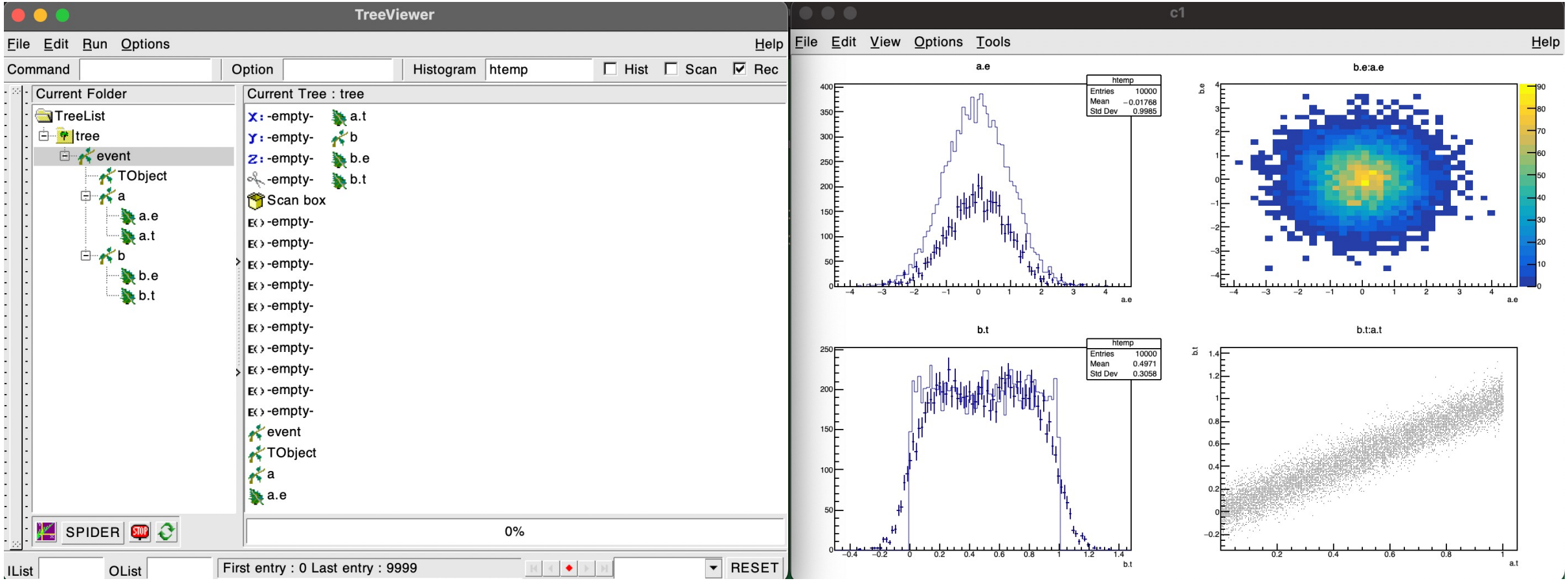
Arboles (Tree)

- Se utiliza la clase TTree
  - https://root.cern.ch/doc/master/classTTree.html
- Los arboles pueden soportar una gran cantidad de colección de objetos.
- La información en un árbol se puede obtener de manera directa o aleatoria
- Esta clase esta optimizada para reducir espacio en disco y optimizada para velocidad de acceso.



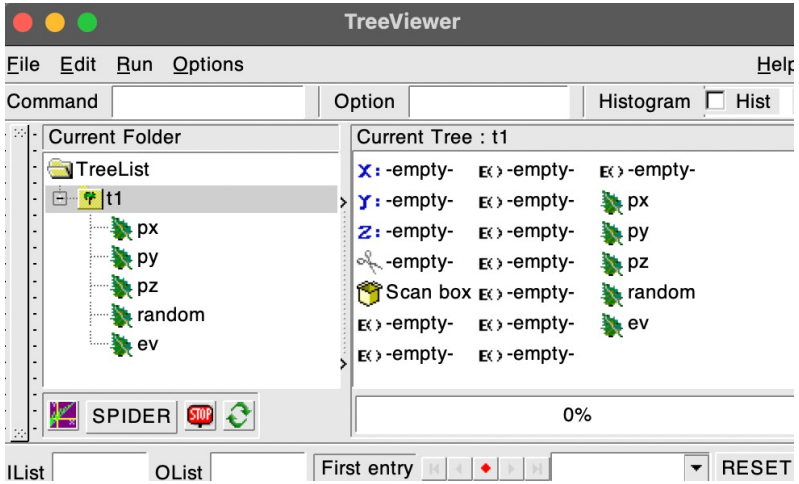
Arboles (Tree)

- Ejemplo tree0.C



Arboles (Tree)

- Ejemplo tree1.C




```
root [1] t1->Show(1)
======> EVENT:1
 px               = 0.32719
 py               = 0.0378782
 pz               = 0.108488
 random           = 0.484974
 ev               = 1
```

```
root [3] t1->Scan("*")
************************************************************************
*    Row   *     px.px *     py.py *     pz.pz * random.ra *     ev.ev *
************************************************************************
*        0 * 0.0194094 * 0.0118254 * 0.0005165 * 0.2826178 *         0 *
*        1 * 0.3271895 * 0.0378782 * 0.1084877 * 0.4849736 *         1 *
*        2 * -0.294611 * -0.010545 * 0.0869072 * 0.5400436 *         2 *
*        3 * -0.774589 * 0.0484584 * 0.6023373 * 0.6586366 *         3 *
```

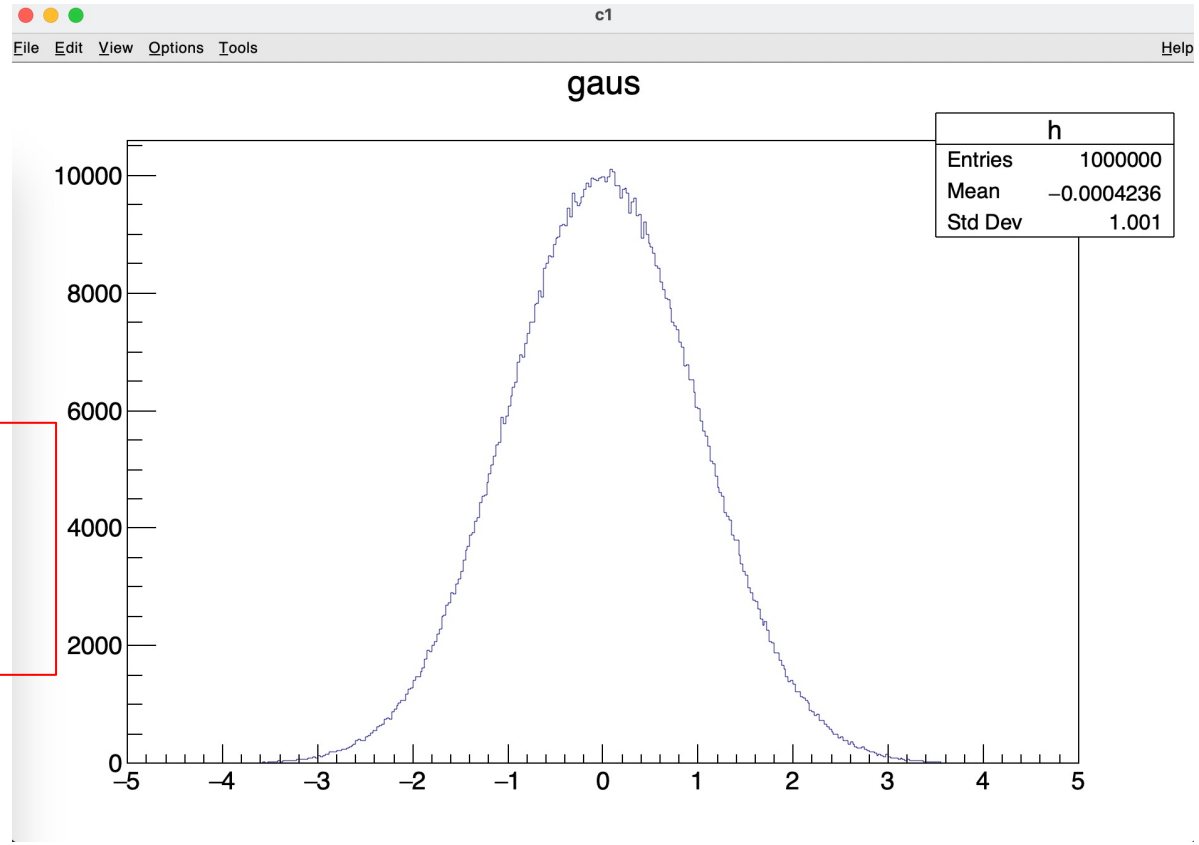
```
root [15] t1->Print()
******************************************************************************
*Tree    :t1        : a simple Tree with simple variables                     *
*Entries :    10000 : Total =          243611 bytes  File  Size =     180826 *
*        :          : Tree compression factor =   1.34                       *
******************************************************************************
*Br    0 :px        : px/F                                                    *
*Entries :    10000 : Total  Size=      40615 bytes  File Size  =      37333 *
*Baskets :        2 : Basket Size=      32000 bytes  Compression=   1.08      *
*.............................................................................*
*Br    1 :py        : py/F                                                    *
*Entries :    10000 : Total  Size=      40615 bytes  File Size  =      37310 *
*Baskets :        2 : Basket Size=      32000 bytes  Compression=   1.08      *
*.............................................................................*
*Br    2 :pz        : pz/F                                                    *
*Entries :    10000 : Total  Size=      40615 bytes  File Size  =      36663 *
*Baskets :        2 : Basket Size=      32000 bytes  Compression=   1.09      *
*.............................................................................*
*Br    3 :random    : random/D                                                *
*Entries :    10000 : Total  Size=      80722 bytes  File Size  =      54555 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   1.47      *
*.............................................................................*
*Br    4 :ev        : ev/I                                                    *
*Entries :    10000 : Total  Size=      40615 bytes  File Size  =      14155 *
*Baskets :        2 : Basket Size=      32000 bytes  Compression=   2.84      *
*.............................................................................*
```

```
root [11] t1->Scan("*","py.py>3.3")
************************************************************************
*    Row   *     px.px *     py.py *     pz.pz * random.ra *     ev.ev *
************************************************************************
*      605 * 3.3974299 * 3.3683254 * 22.888145 * 0.9418406 *       605 *
*     3103 * 0.7493053 * 3.3094620 * 11.513998 * 0.7039981 *      3103 *
*     6968 * -0.391461 * 3.5674879 * 12.880212 * 0.9972652 *      6968 *
*     7556 * 1.3714664 * 3.3104612 * 12.840074 * 0.1000605 *      7556 *
*     9512 * -1.344522 * 3.3791277 * 13.226246 * 0.2540522 *      9512 *
*     9818 * -0.007705 * 3.9492180 * 15.596382 * 0.0831928 *      9818 *
************************************************************************
==> 6 selected entries
(long long) 6
```

PyROOT

- Interfaz con el lenguaje de programación Python

```
>>>import ROOT
>>>h = ROOT.THF1("h","histograma gausiano",300,-5,5)
>>>r = ROOT.Trandom()
>>>for i in range(1000000):
...       h.Fill(r.Gaus())
>>>h.Draw()
```



Python es un lenguaje que interpreta comandos
ejecuta línea a línea o
se crea un macro con extensión py
dentro de la consola de Python (v3) se ejecuta con la instrucción

```
(base) barbosa@barbosa curso % python
Python 3.8.5 (default, Sep  4 2020, 02:22:02)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('histo.py').read())
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
>>>
```

Muchas gracias

Este tutorial se basa en los tutoriales que se encuentran en la base de datos de CERN