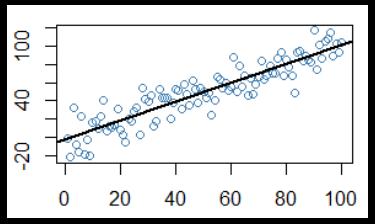
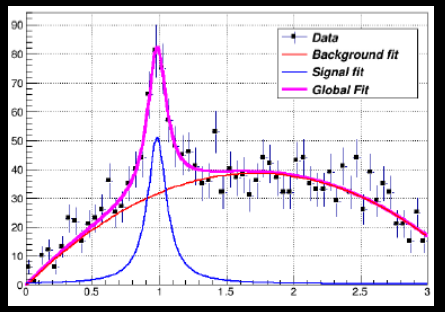
Temas avanzados en física computacional
Análisis de datos





Semestre

2016-I

Clase-3

José Bazo

jbazo@pucp.edu.pe

Contenidos del curso

1. ✓ Introducción al análisis de datos y data science
2. ✓ Lenguaje de programación R
3. ROOT Data Analysis Framework
4. Manipulación y visualización de datos
5. Modelamiento estadístico
6. Machine Learning
7. TMVA (Toolkit for Multivariate Data Analysis)

3. ROOT Data Analysis Framework

Referencias

D. Piparo, O. Couet Summer Students Course 2015. CERN PH-SFT Video

A ROOT Guide For Beginners

ROOT Users Guide

Class index

Tutorial

ROOT

https://root.cern.ch/

Release 6.06/02 - 2016-03-03

Release 5.34/36 - 2016-04-05

ROOT 6: compiler needs c++11 for building

Sistemas operativos:

- Linux x86-64 with gcc >= 4.8
- MacOS X >= 10.8 with clang

- Virtual machine

Sistemas operativos:

- Linux x86-64 with gcc >= 4.8
- MacOS X >= 10.8 with clang
- Win64 with cygwin/gcc with gcc >= 4.8

En .bashrc :
source /Applications/root_v6.04.14/bin/thisroot.sh

Carácterísticas de ROOT

- Maneja n-tuplas, histogramas, cuadrivectores, geometría de detectores, diagramas de Feynman, algebra lineal, ajuste de funciones, análisis multivariado, etc.

- Puede manejar grandes cantidades de datos (millones de eventos físicos, archivos de Gb - Tb)

- Multi-plataforma: Windows, Mac, Linux de datos

- Gratis

- Hay que conocer C++ para usarlo correctamente.
(código antiguo de comunidad HEP en FORTRAN a C++)

- Pero existe pyroot

- PROOF: Parallel ROOT Facility

- Procesamiento
- Análisis
- Visualización
- Almacenamiento

} de datos

Interpretador

- Interpretador CLING de C++ (versión anterior CINT)

Reglas relajadas de sintaxis y manejo de memoria automático pero no impecable.

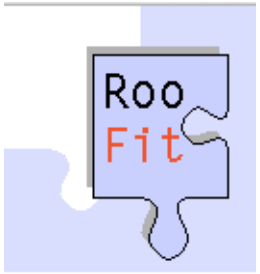
- Shell interactivo en C++
- Puede interpretar (leer) “macros” (programas no compilados)

- Es posible usar también interpretación de python incluyendo: import ROOT

Librerías

ROOT también tiene librerías y herramientas para análisis estadísticos sofisticados de datos

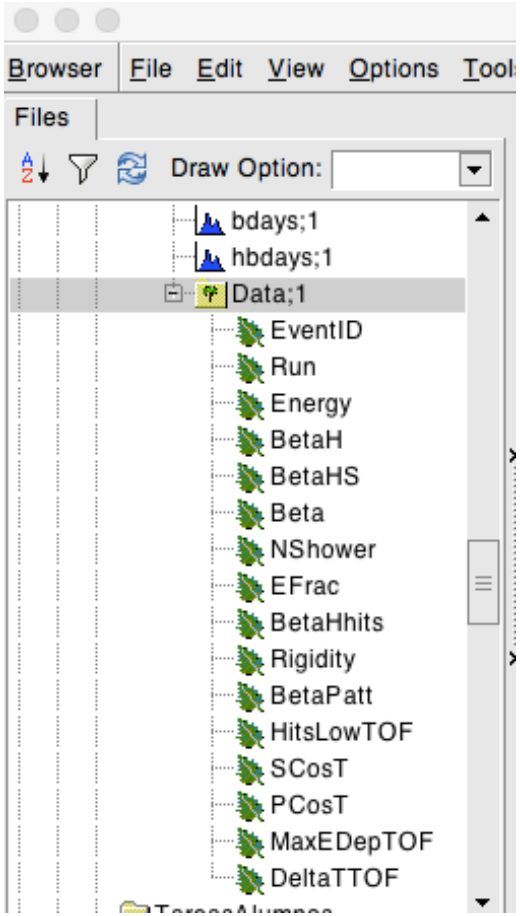
RooFit / RooStats: Toolkit for Data Modeling with ROOT



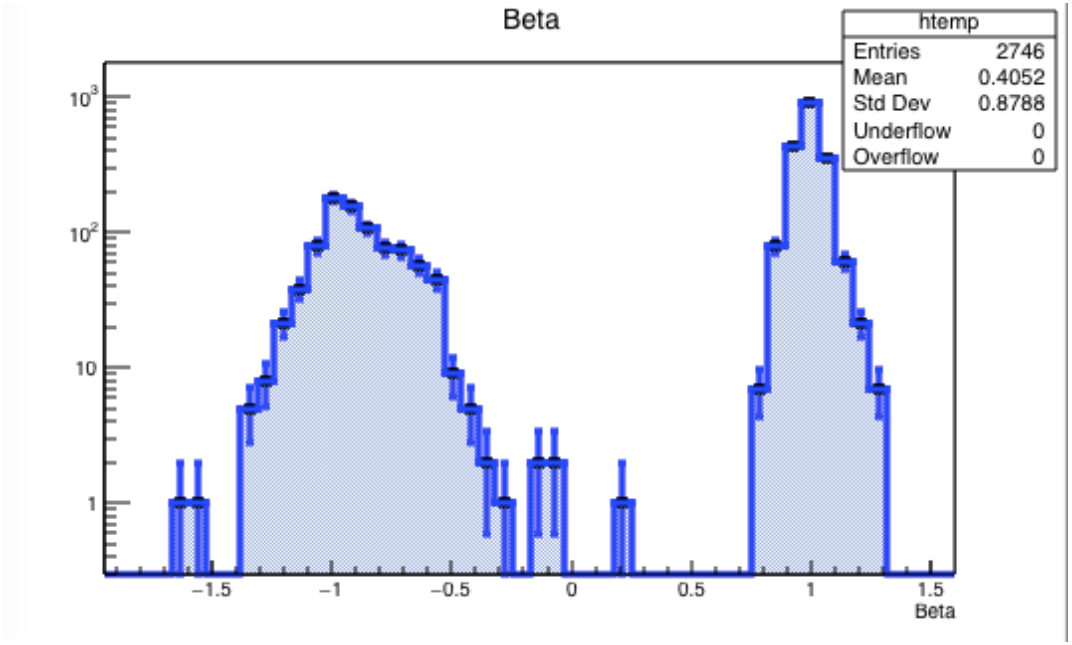
TMVA: Toolkit for Multivariate Data Analysis with ROOT

Introducción

Usar valores guardados en una n-tupla para realizar cálculos y graficar histogramas.



Histograma



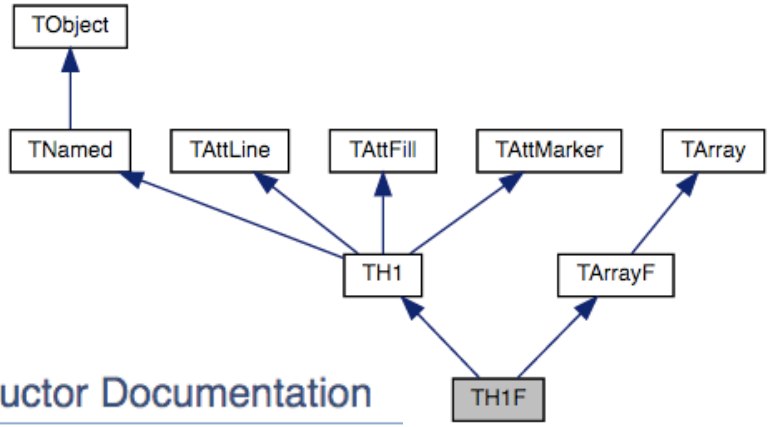
TH1F : un float por canal (precisión 7 dígitos)

TH1F (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup)

TTree: lista ordenada de objetos de C++

Ejemplo de clase

Class:
Data members (“variables of the class”)
Class methods (“functions of the class”)
Object (instance of a class) created by constructor



TH1F Class Reference

Histogram Library

Public Member Functions

TH1F ()

TH1F (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup)
Create a 1-Dim histogram with fix bins of type float (see TH1::TH1 for explanation of parameters) More...

TH1F (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins)
Create a 1-Dim histogram with variable bins of type float (see TH1::TH1 for explanation of parameters) More...

TH1F (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins)
Create a 1-Dim histogram with variable bins of type float (see TH1::TH1 for explanation of parameters) More...

Static Public Member Functions

static void SetDefaultSumw2 (Bool_t sumw2=kTRUE)
When this static function is called with sumw2=kTRUE, all new histograms will automatically activate the storage of the sum of squares of errors, ie TH1::Sumw2 is automatically called. More...

Constructor & Destructor Documentation

TH1F::TH1F ( )

Acceder a métodos y miembros

- Para acceder a métodos y miembros de objetos se usa "**.**"
- Para acceder a punteros de objetos se usa "**->**"

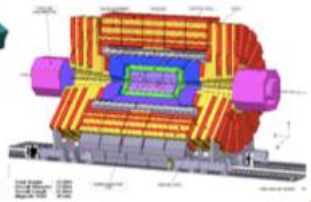
Ejemplo:

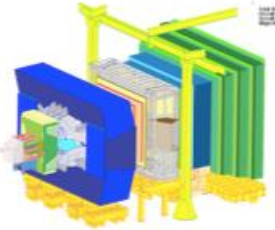
```
MyClass myClassInstance("myName");
myClassInstance.GetName();

auto myClassInstancePtr = new MyClass ("myName");
myClassInstancePtr->GetName();
```

Introducción

ROOT Application Domains

A selection of the experiments adopting ROOT





Event Filtering

Data

Analysis

Offline Processing

Event Selection, statistical treatment ...

Further processing, skimming

Reconstruction

Raw

Reco

...

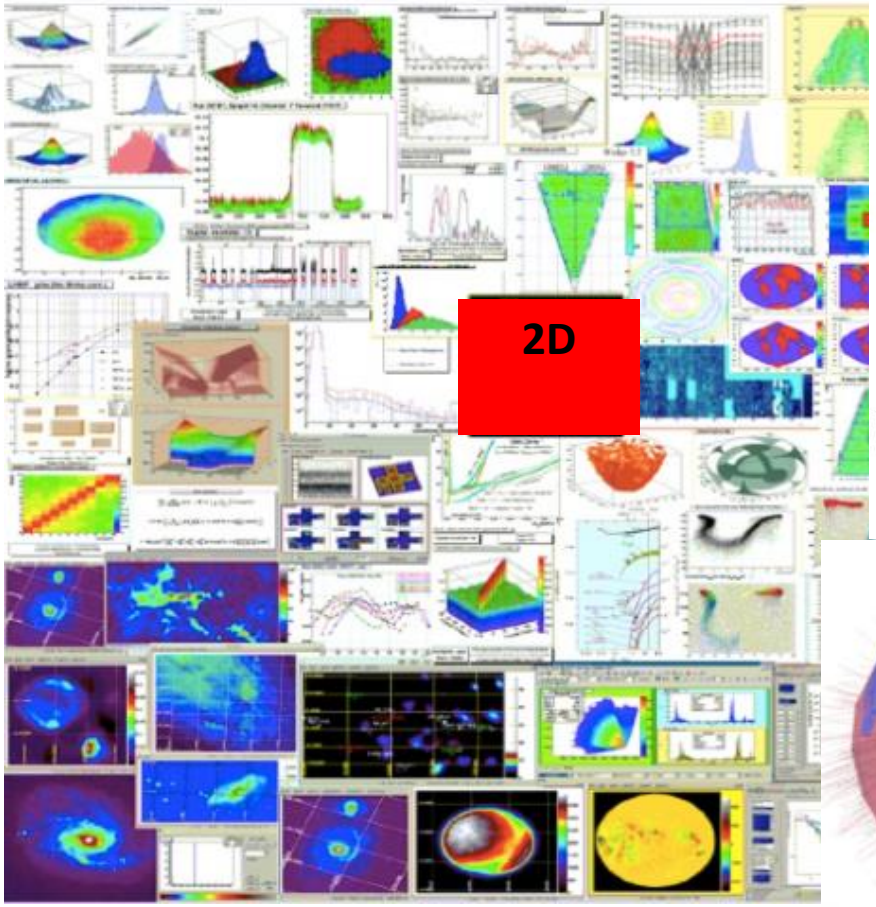
Analysis Formats

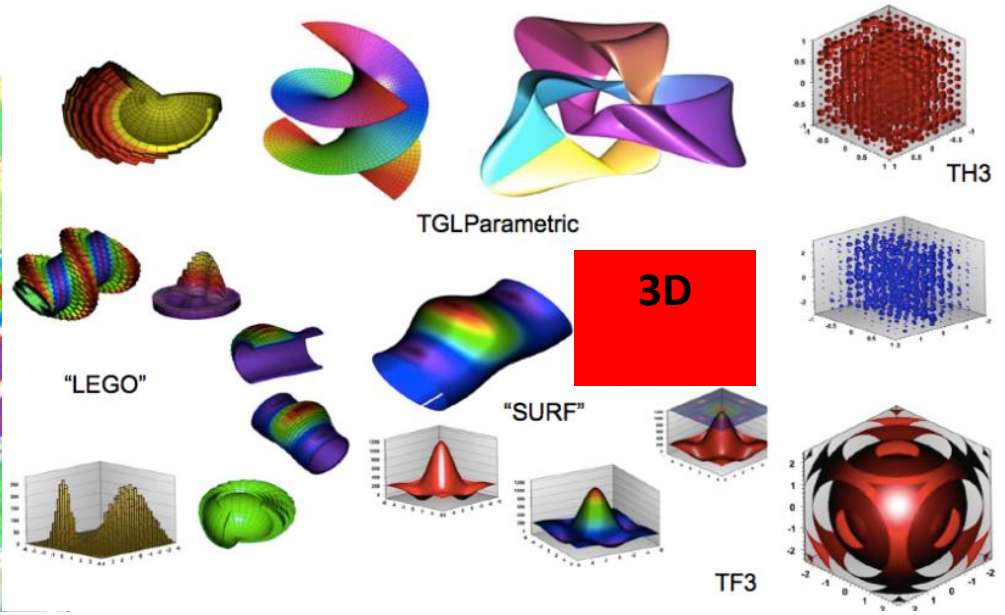
Images

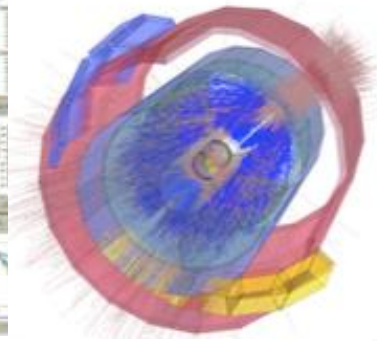
Data Storage: Local, Network

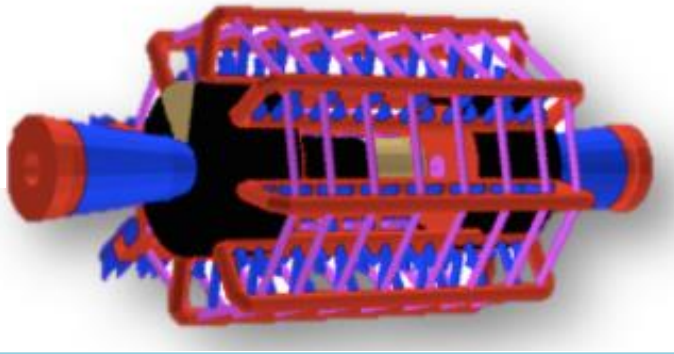
ROOT Summer Student Tutorial 2015 8

Gráficos









ROOT interactive prompt

$ root ← Shell interactivo: “ROOT prompt”

```
 ------------------------------------------------------------
| Welcome to ROOT 6.04/14                 http://root.cern.ch |
|                                (c) 1995-2014, The ROOT Team |
| Built for macosx64                                          |
| From tag v6-04-14, 3 February 2016                          |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
 ------------------------------------------------------------

root [0] .q
```

Se puede usar como calculadora:

```
root [0] 1+1
(int)2
root [1] 2*(4+2)/12.
(double) 1.000000e+00
root [2] sqrt(3.)
(double) 1.732051e+00
root [3] 1 > 2
(bool) false
```

TMath:: (namespace reference)

```
root [4] TMath::Pi()
(Double_t) 3.141593e+00
root [5] TMath::Erf(.2)
(Double_t) 2.227026e-01
```

ROOT interactive prompt

Comandos especiales empiezan con “.”

- .q : salir de root
- .!<OS_command> : ejecutar un comando de shell gSystem->cd(” ”)
- .L library | filename.cxx : cargar una librería o filename.cxx (macro)
- .x filename.cxx : carga filename y llama a void filename(), si está definido
- .I path : adds an include path
- .help :lista completa de comandos

Definir variables y loops

Se deben declarar tipos de variables:

```
root [6] double x=.5
(double) 5.000000e-01
root [7] int N=30
(int) 30
root [8] double gs=0
(double) 0.000000e+00
```

Así se usan los loops como *for*

```
root [9] for (int i=0;i<N;++i) gs += TMath::Power(x,i)
```

```
for(auto v:{1,4,5}) cout <<v<<endl;
```

Funciones

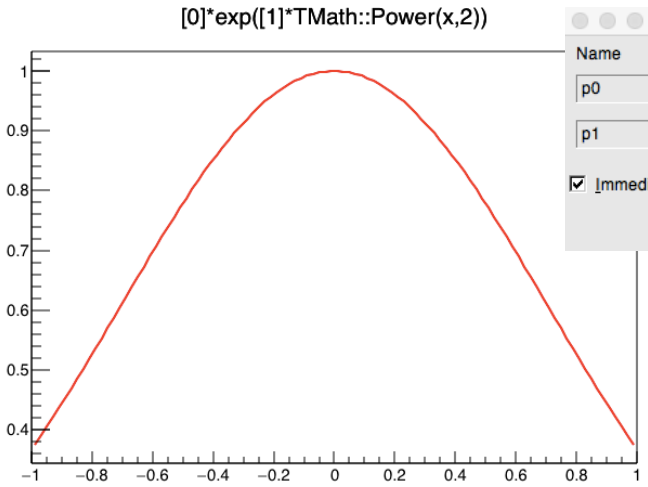
Funciones matemáticas en una dimensión: clase TF1

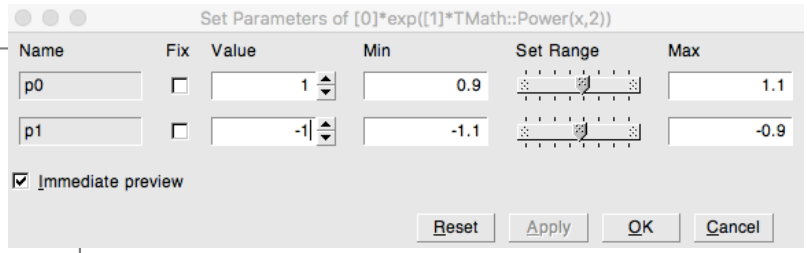
```
root [0] TF1 f1("f1","sin(x)/x",0.,10.);//name,formula, min, max
root [1] f1.Draw();
```

Se pueden definir también parámetros

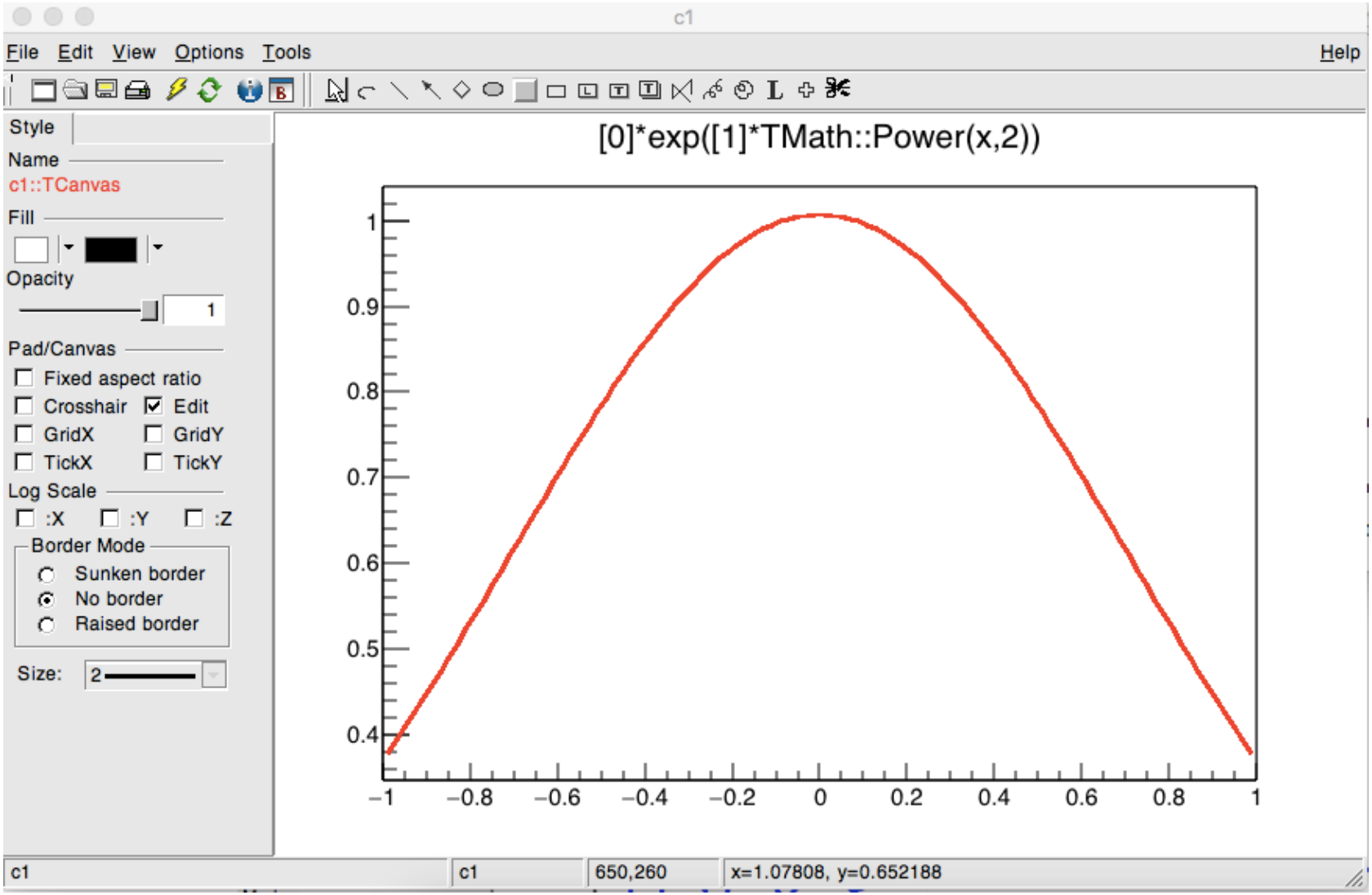
```
root [2] TF1 f2("f2","[0]*sin([1]*x)/x",0.,10.);
root [3] f2.SetParameters(1,1);
root [4] f2.Draw();
```

TF1 *f0=new **TF1**("f0","[0]*exp([1]***TMath**::Power(x,2))",-1,1);
f0->SetParameters(1,-1)
f0->Draw();





Gráficos


c1
File Edit View Options Tools Help
Style
Name
c1::TCanvas
Fill
Opacity
1
Pad/Canvas
Fixed aspect ratio
Crosshair
Edit
GridX
GridY
TickX
TickY
Log Scale
:X
:Y
:Z
Border Mode
Sunken border
No border
Raised border
Size:
2
[0]*exp([1]*TMath::Power(x,2))
1
0.9
0.8
0.7
0.6
0.5
0.4
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
c1
c1
650,260
x=1.07808, y=0.652188

ROOT Macro

Programa simple guardado como Macro.C
Sin función main(), para la función usar el mismo nombre del archivo:

```
void Macro() {  código en C++  }
```

Formas de ejecutarlo:

Desde la terminal: root Macro.C

Dentro de ROOT: .x Macro.C;
o en dos pasos: .L Macro.C;
Macro();

Compilación

Dentro de ROOT un programa se puede compilar:
.L macro.C+
y ejecutar: macro()

Genera librerías compartidas y ejecuta función

O ambos a la vez: .x macro.C+

Y para producir aplicaciones standalone:

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

```
#include <iostream>
#include <TFile.h>
using namespace std;

int macro(){
cout <<"test"<<endl;
return 0; }
```

Histogramas

TH1 class (unidimensional), TH2 (bidimensional)

Según el tipo de número salvado se agrega una letra:
TH1F (float)
TH1D (double)
También C (byte), S(short) e I(int)

Crear histograma con bins de ancho fijo

TH1F (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup)
Create a 1-Dim histogram with fix bins of type float (see **TH1::TH1** for explanation of parameters) More...

Crear histograma con bins de ancho variables

TH1F (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins)
Create a 1-Dim histogram with variable bins of type float (see **TH1::TH1** for explanation of parameters)

```
const int nbins=5;
float bins_array[nbins+1]={1,2,5,10,20,100};
TH1F *H1=new TH1F("H1","",nbins,bins_array);
```

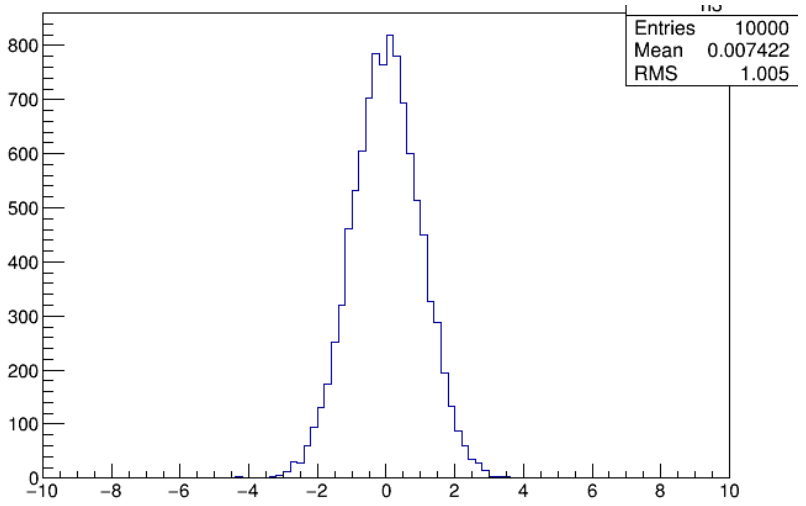
Histogramas

Para rellenar el histograma creado se usa:

Histo->**Fill**(valor, peso) or simplemente Histo->Fill(valor)

```
TH1 *h=new TH1("h","",100,-10,10)
TF1 *Gaus=new TF1("Gaus","gaus",-10.,10.);
Gaus.SetParameters(1,0,1);
for (int i=0;i<10000;i++) h->Fill(Gaus.GetRandom())
h->Draw()
```

```
[0]*exp(-0.5*((x-[1])/[2])**2)
```



Histogramas

Otras funciones útiles:

Hist->**FillRandom**("gaus",10000);

virtual void **FillRandom** (const char *fname, **Int_t** ntimes=5000)
Fill histogram following distribution in function fname.

virtual void **FillRandom** (**TH1** *h, **Int_t** ntimes=5000)
Fill histogram following distribution in histogram h.

Hist->**Sumw2**() Método para que barras de error luego de repesaje se calculen correctamente.

Hist->**Add**(Hist2)

virtual **Bool_t** **Add** (const **TH1** *h1, **Double_t** c1=1)
Performs the operation: `this = this + c1*h1`

Hist->**Divide**(Hist2)

virtual **Bool_t** **Divide** (**TF1** *f1, **Double_t** c1=1)
Performs the operation: `this = this/(c1*f1)`

virtual **Bool_t** **Divide** (const **TH1** *h1, const **TH1** *h2, **Double_t** c1=1, **Double_t** c2=1, **Option_t** *option="")

`this = c1*h1/(c2*h2)`

Histogramas

Como no perder un histograma de un file que se cierra

```
TFile *f = new TFile( "test.root" );
TH1F *h = (TH1F *) gDirectory->Get("histo");
h->SetDirectory(0); // "detach" the histogram from the file
f->Close();
h->Draw();
```

Sacar valores de un histograma:

```
virtual Double_t   GetBinCenter (Int_t bin) const
                   Return bin center for 1D histogram.

virtual Double_t   GetBinWidth (Int_t bin) const
                   Return bin width for 1D histogram.

virtual Double_t   GetBinError (Int_t bin) const
```

y set para definirlos:

```
virtual void   SetBinContent (Int_t bin, Double_t content)

virtual void   SetBinError (Int_t bin, Double_t error)
```

Histograma de la función cumulativa:

```
TH1 *   GetCumulative (Bool_t forward=kTRUE, const char *suffix="_cumulative") const
        Return a pointer to an histogram containing the cumulative The cumulative can be computed both in the
        forward (default) or backward direction; the name of the new histogram is constructed from the name of
        this histogram with the suffix suffix appended. More...
```

Histogramas

Número de entradas

virtual **Double_t** **GetEntries** () const
Return the current number of entries.

Máximos y mínimos

virtual **Double_t** **GetMaximum** (**Double_t** maxval=FLT_MAX) const

virtual **Int_t** **GetMaximumBin** () const

virtual **Double_t** **GetMinimum** (**Double_t** minval=-FLT_MAX) const

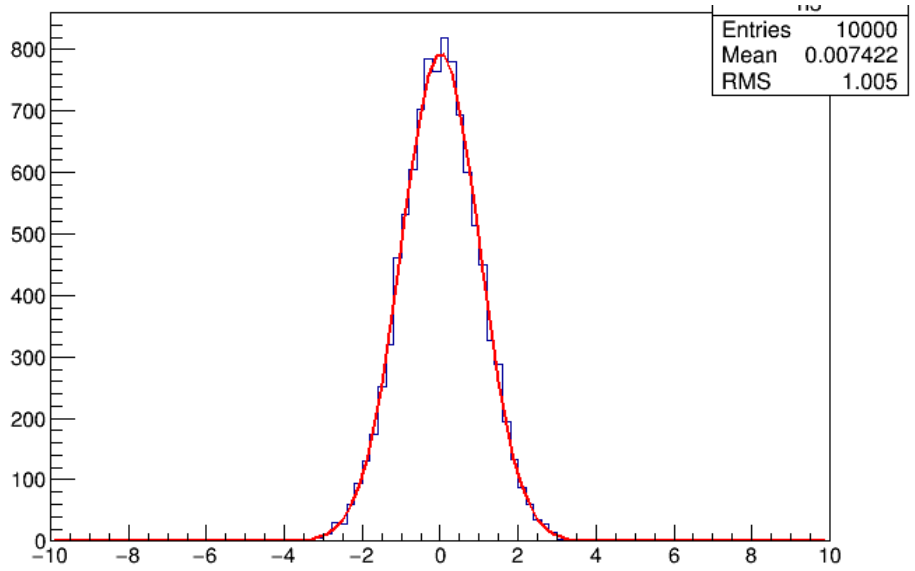
Integral

virtual **Double_t** **Integral** (**Int_t** binx1, **Int_t** binx2, **Option_t** *option="") const
Return integral of bin contents in range [binx1,binx2]. More...

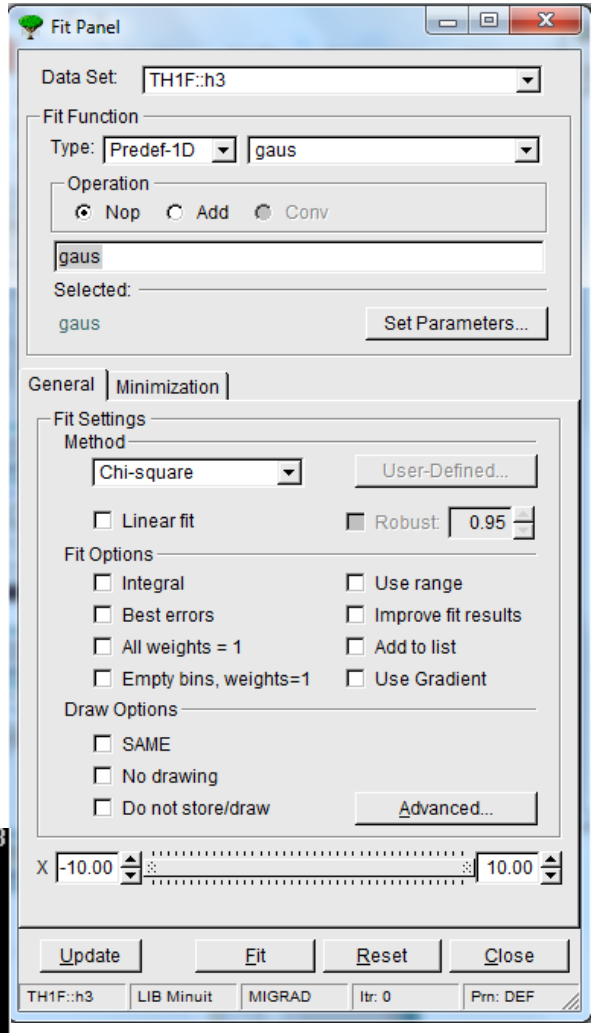
Interpolar

virtual **Double_t** **Interpolate** (**Double_t x**)
Given a point x, approximates the value via linear interpolation based on the two nearest bin centers.

Ajuste



Panel de ajuste:
Click derecho sobre la curva y elegir “Fit Panel”

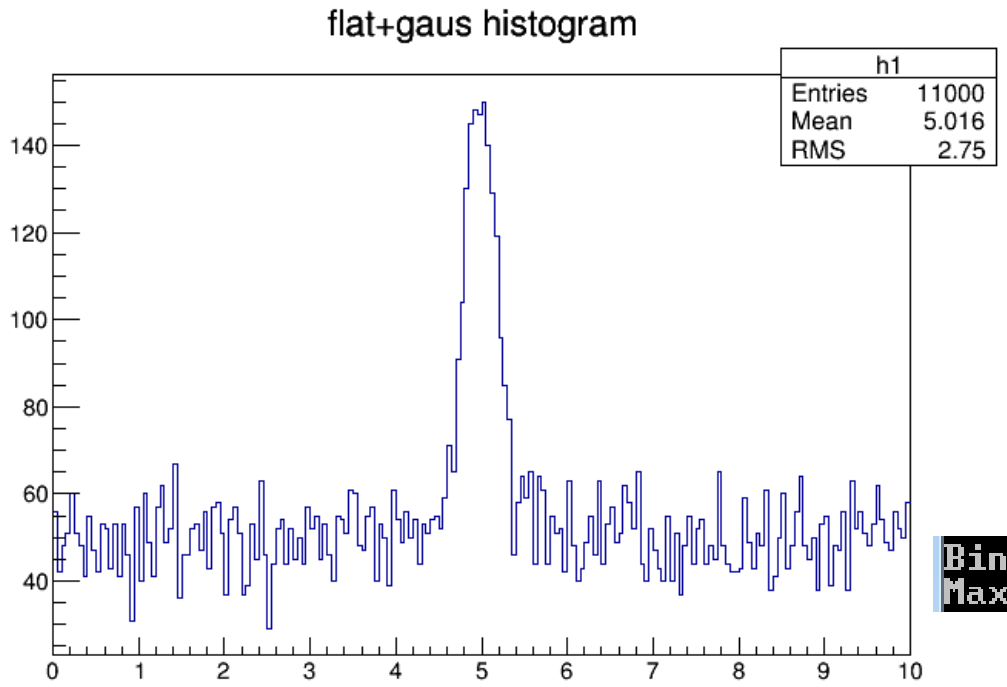


Resultados del ajuste:

```
root [52]  FCN=30.1188 FROM MIGRAD    STATUS=CONVERGED      52 CALLS          53
 TOTAL
                     EDM=8.6323e-010    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant   7.94729e+002   9.78210e+000   2.16808e-002  -4.54331e-006
   2  Mean       6.48153e-003   1.00535e-002   2.73823e-005   3.36610e-004
   3  Sigma      1.00132e+000   7.19927e-003   5.29839e-006  -1.92294e-002
```

Histogramas: Ejercicio 1

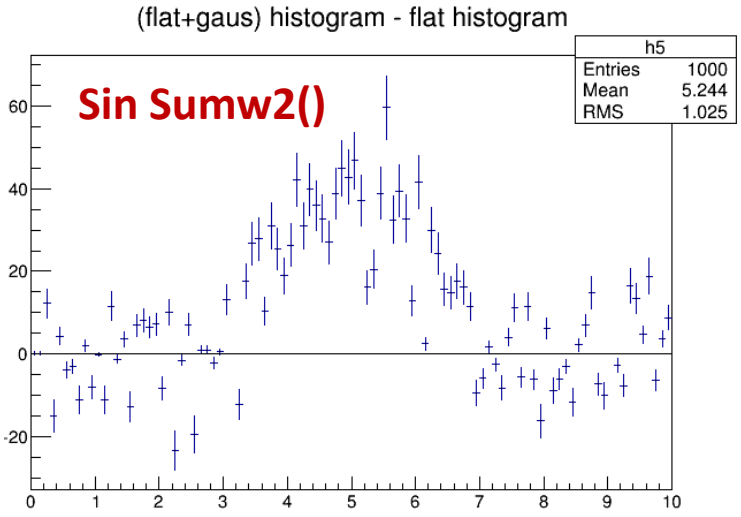
- Crear un histograma con 200 bins entre 0 y 10.
- Rellenar el histograma con 1000 número aleatorios de una Gaussiana con media 5 y sigma 0.3 y 10000 número aleatorios de una distribución uniforme entre 0 y 10.
- Graficar el histograma.
- Encontrar el número del bin del histograma con mayor altura y este valor.
- ¿Cuál es el centro del bin y su error?

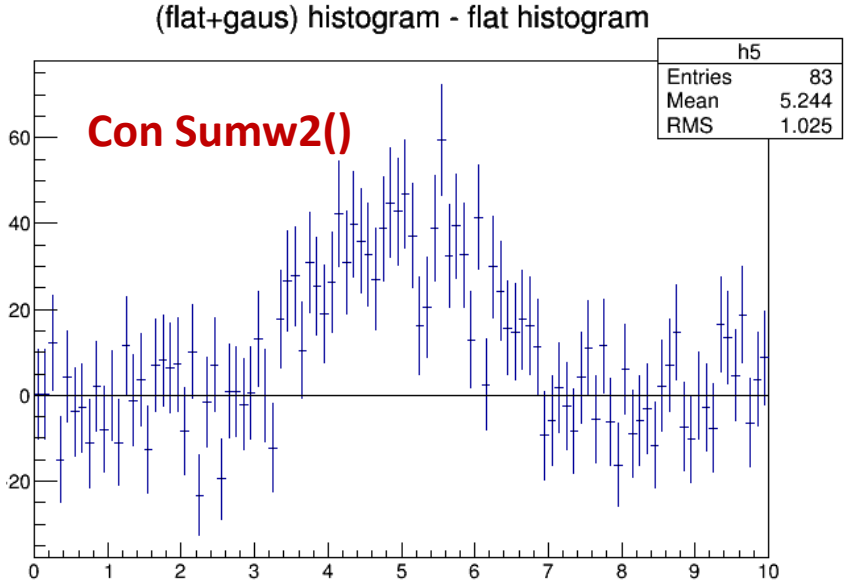



```
Bin with maximum is 101 at x = 5.025
Maximum Content   is 150 +/- 12.2474
```

Histogramas: Ejercicio 2

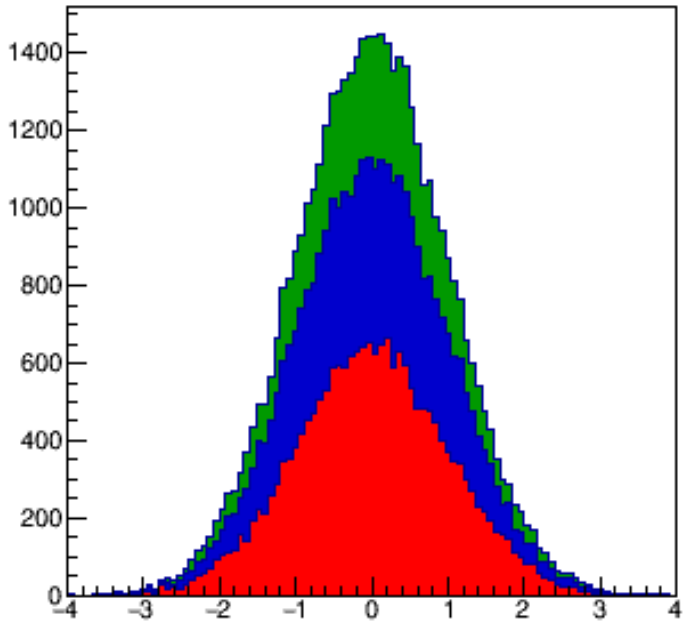
- Crear un histograma de 100 bins entre 0 y 10 rellenado con 1000 valores tomados de una gaussiana con media 5 y sigma 1.
- Crear otro histograma similar pero de una distribución uniforme y con 10000 entradas.
- Sumar los dos histogramas en uno nuevo.
- Preparar otro histograma pero con 100000 valores uniformes. Normalizar este histograma para que su integral de 10000.
- Restar al histograma de la suma el histograma normalizado.
- Graficar el resultado con los errores.





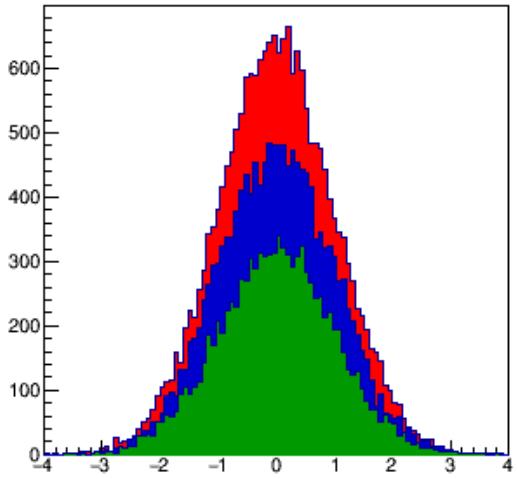
Stacked histos

THStack *hs = new **THStack**("hs","");




```
TH1F *h1 = new TH1F("h1","",100,-4,4);
  h1->FillRandom("gaus",20000);
  h1->SetFillColor(kRed);
  hs->Add(h1);
  TH1F *h2 = new TH1F("h2","",100,-4,4);
  h2->FillRandom("gaus",15000);
  h2->SetFillColor(kBlue+1);
  hs->Add(h2);
  TH1F *h3 = new TH1F("h3","",100,-4,4);
  h3->FillRandom("gaus",10000);
  h3->SetFillColor(kGreen+2);
  hs->Add(h3);
  hs->Draw();
```

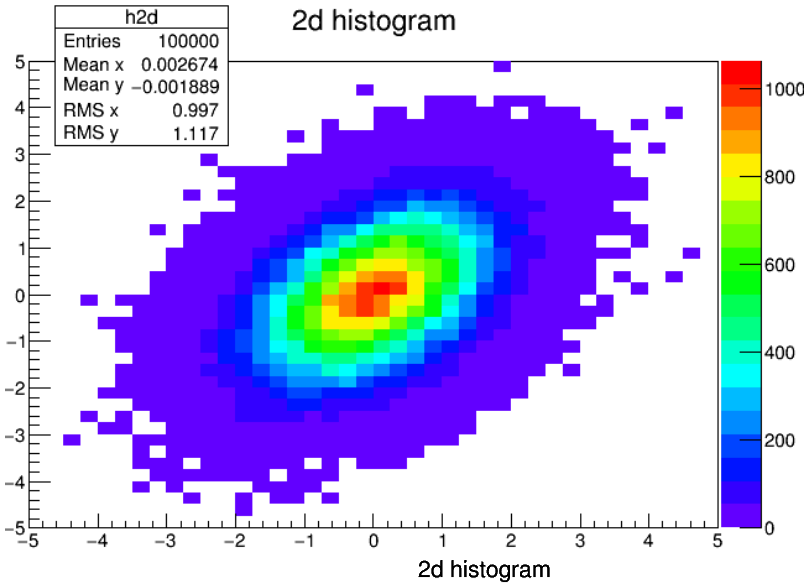
```
h1->Draw();
h2->Draw("same");
h3->Draw("same");
```



TH2

TH2D * h2d = new TH2D("h2d","2d histogram",40,-5,5, 40, -5, 5);

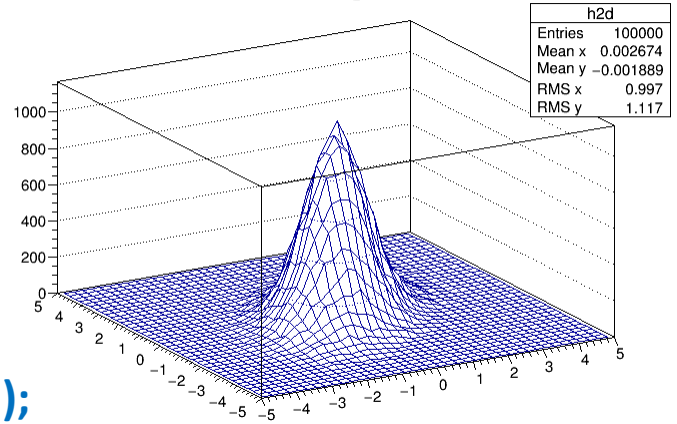
```
for (int i = 0; i < 100000; ++i) {
   double u =  gRandom->Gaus(0,1);
   double w =  gRandom->Gaus(0,1);
   double x = u;
   double y = w + 0.5 * u;
   h2d->Fill( x,y );
}
```



h2d->**Draw("COLZ");**

h2d->**GetCorrelationFactor**()

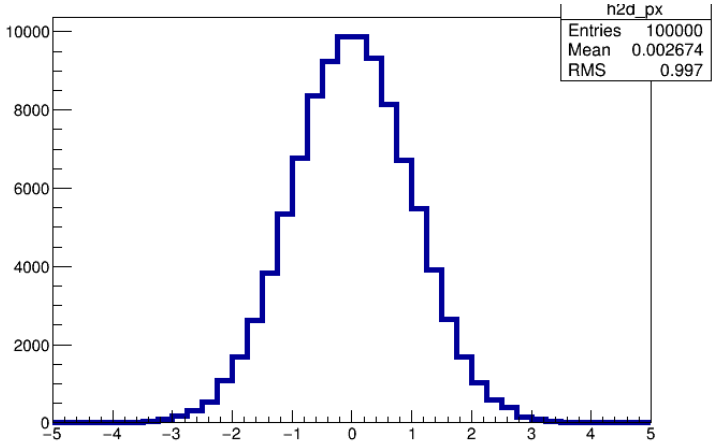
```
correlation factor 0.448124
```



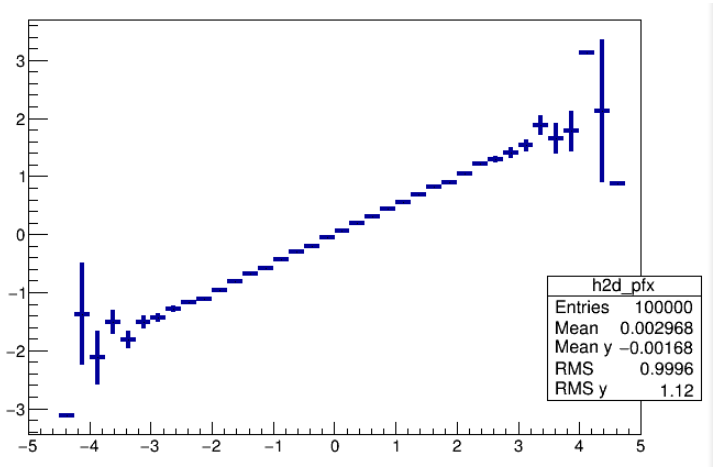
h2d->**Draw("SURF");**

Perfiles y proyecciones

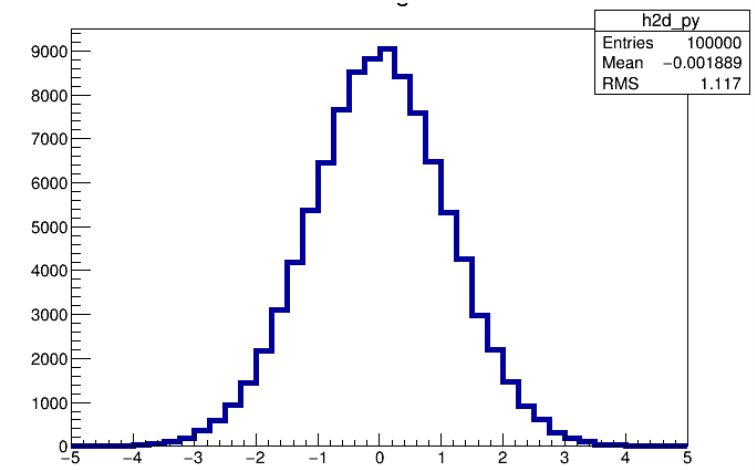
TH1D *hx = h2d->**ProjectionX**();



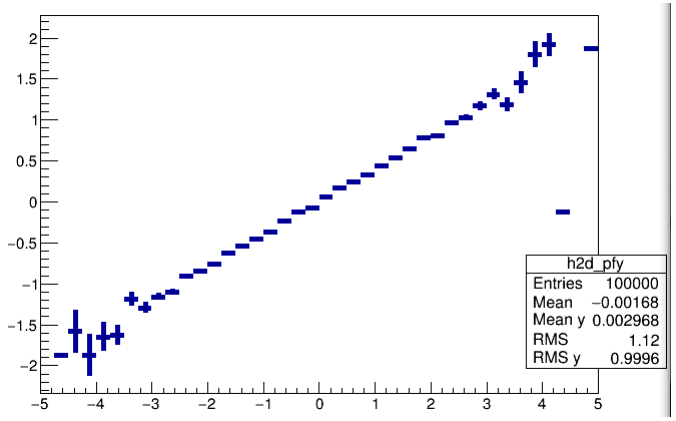
TH1D *px = h2d->**ProfileX**();



TH1D *hy = h2d->**ProjectionY**();



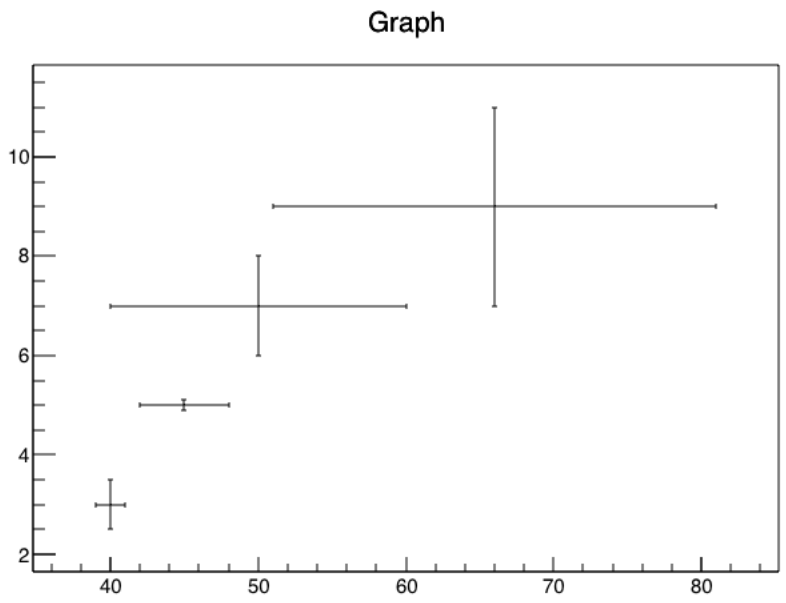
TH1D *py = h2d->**ProfileY**();



TGraphErrors

Para graficar un conjunto de puntos con errores usar clase [TGraphErrors]():

```
root [0] TGraphErrors gr("ExampleData.txt");
root [1] gr.Draw("AP");
```



"A"	Axis are drawn around the graph
"L"	A simple polyline is drawn
"F"	A fill area is drawn ('CF' draw a smoothed fill area)
"C"	A smooth Curve is drawn
"*"	A Star is plotted at each point
"P"	The current marker is plotted at each point
"B"	A Bar chart is drawn

Archivo con puntos

```
40 3 1  0.5
45 5 3  0.1
50 7 10  1
66 9 15  2
```

Opciones de error

Option	Description
"Z"	Do not draw small horizontal and vertical lines the end of the error bars. Without "Z", the default is to draw these.
">"	An arrow is drawn at the end of the error bars. The size of the arrow is set to 2/3 of the marker size.
"I>"	A filled arrow is drawn at the end of the error bars. The size of the arrow is set to 2/3 of the marker size.
"X"	Do not draw error bars. By default, graph classes that have errors are drawn with the errors (TGraph itself has no errors, and so this option has no effect.)
"II"	Draw only the small vertical/horizontal lines at the ends of the error bars, without drawing the bars themselves. This option is interesting to superimpose statistical-only errors on top of a graph with statistical+systematic errors.
"[]"	Does the same as option "II" except that it draws additional marks at the ends of the small vertical/horizontal lines. It makes plots less ambiguous in case several graphs are drawn on the same picture.
"0"	By default, when a data point is outside the visible range along the Y axis, the error bars are not drawn. This option forces error bars' drawing for the data points outside the visible range along the Y axis (see example below).
"2"	Error rectangles are drawn.
"3"	A filled area is drawn through the end points of the vertical error bars.
"4"	A smoothed filled area is drawn through the end points of the vertical error bars.
"5"	Error rectangles are drawn like option "2". In addition the contour line around the boxes is drawn. This can be useful when boxes' fill colors are very light or in gray scale mode.

gStyle->SetErrorX(dx) controla el tamaño del error en el eje x.
Con dx = 0 se remueve el error en x.

TGraphErrors

Ejemplo con opciones para título, nombre de ejes, atributos estéticos:

```
root [0] TGraph g
root [1] g.SetTitle("My graph;myX;myY")
root [2] g.SetPoint(0,1,0)
root [3] g.SetPoint(1,2,3)
root [4] g.SetPoint(2,3,4)
root [5] g.SetMarkerStyle(kFullSquare)
root [6] g.SetMarkerColor(kRed)
root [7] g.SetLineColor(kOrange)
root [8] g.Draw("APL")
```

```
g.GetXaxis()->SetRangeUser(0,4);
g.GetYaxis()->SetRangeUser(0,4);
```

```
TLegend *leg = new TLegend(0.1,0.7,0.48,0.9);
leg->SetHeader("The Legend Title");
leg->AddEntry(h1,"Histogram filled with random numbers","f");
leg->AddEntry("f1","Function abs(#frac{sin(x)}{x})","l");
leg->AddEntry("gr","Graph with error bars","lep");
leg->Draw();
```

Punteros Globales

gROOT: punto de entrada al sistema ROOT, instancia de la clase TROOT. Se puede acceder a todos los objetos creados en un programa de ROOT

gStyle: acceder al estilo predeterminado de ROOT. Se pueden determinar los siguientes atributos de objetos:
– Canvas, Pad, ejes de histogramas, líneas, relleno de áreas, texto, markers, funciones, estadísticas de histogramas, títulos, etc.

gSystem: interface al sistema operativo subyacente.

gInterpreter: TCling.

gStyle

Frame

```
gStyle->SetFrameBorderMode(kFALSE);
```

Canvas

```
gStyle->SetCanvasBorderMode(kFALSE);
gStyle->SetCanvasColor(kWhite);
gStyle->SetCanvasDefX(500);
gStyle->SetCanvasDefY(500);
gStyle->SetCanvasDefH(500);
gStyle->SetCanvasDefW(500);
```

Pad

```
gStyle->SetPadBorderMode(kFALSE);
gStyle->SetPadColor(kWhite);
gStyle->SetPadTickX(kTRUE);
gStyle->SetPadTickY(kTRUE);
gStyle->SetPadTopMargin(0.055);
gStyle->SetPadRightMargin(0.15);
gStyle->SetPadBottomMargin(0.15);
gStyle->SetPadLeftMargin(0.17);
```

Title

```
gStyle->SetOptTitle(kFALSE);
gStyle->SetTitleColor(kBlack);
gStyle->SetTitleBorderSize(0);
gStyle->SetTitleX(0.25);
gStyle->SetTitleY(0.98);
gStyle->SetTitleOffset(0.9,"x");
gStyle->SetTitleOffset(1.2,"y");
gStyle->SetTitleSize(0.07,"xyz");
gStyle->SetTitleFont(42,"xyz");
```

Stat

```
gStyle->SetOptStat(kFALSE);
gStyle->SetStatColor(kWhite);
gStyle->SetStatFont(42);
```

gStyle

Legend

```
gStyle->SetLegendBorderSize(0);
```

Label

```
gStyle->SetLabelFont(font_type,"xyz");
gStyle->SetLabelSize(0.05,"xyz");
gStyle->SetLabelColor(kBlack,"xyz");
```

Text

```
gStyle->SetTextFont(font_type);
gStyle->SetTextSize(1.1);
```

Fit

```
gStyle->SetOptFit(kFALSE);
```

Paper

```
gStyle->SetPaperSize(20,26);
```

Histos

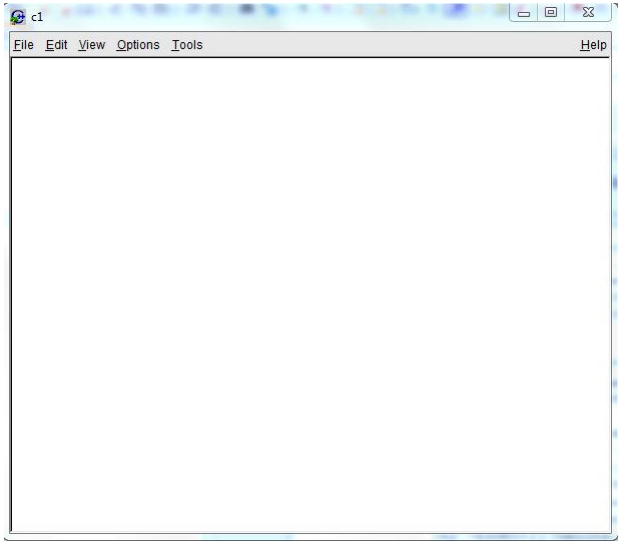
```
gStyle->SetMarkerStyle(20);
gStyle->SetHistLineWidth(2);
gStyle->SetHistFillColor(kWhite);
gStyle->SetLineStyleString(2,"[12 12]");
gStyle->SetLineWidth(3);
gStyle->SetErrorX(0.001);
gStyle->SetPalette(1);
gStyle->SetPaintTextFormat("5.2f");
```

TCanvas

TCanvas Class Reference

Canvas área mapeada en una ventana que puede estar subdividida en pads.

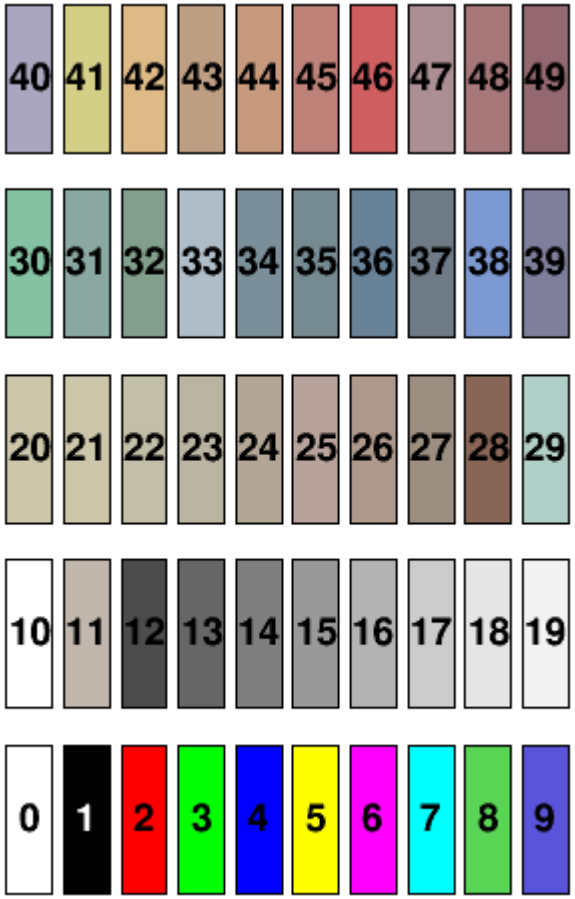
```
TCanvas *c1 = new TCanvas("c1","",600,500);

  c1->Divide (2, 1);  crea TPad::c1_1 y c2_2
  c1->cd(1);
  c1_1->SetGrid(1,1);
  c1_1->SetLogx();
  c1_1->SetLogy();
  f1->Draw();
  c1->cd(2);
  c1->SaveAs("plot.png");
  c1->Close();
```



TColor

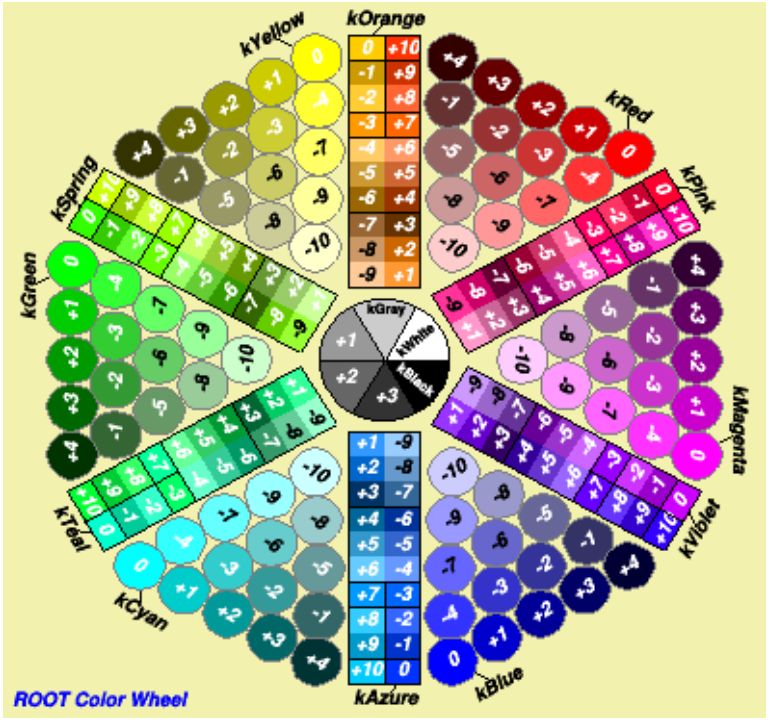
TColor Class Reference

canvas->DrawColorTable()



TColorWheel

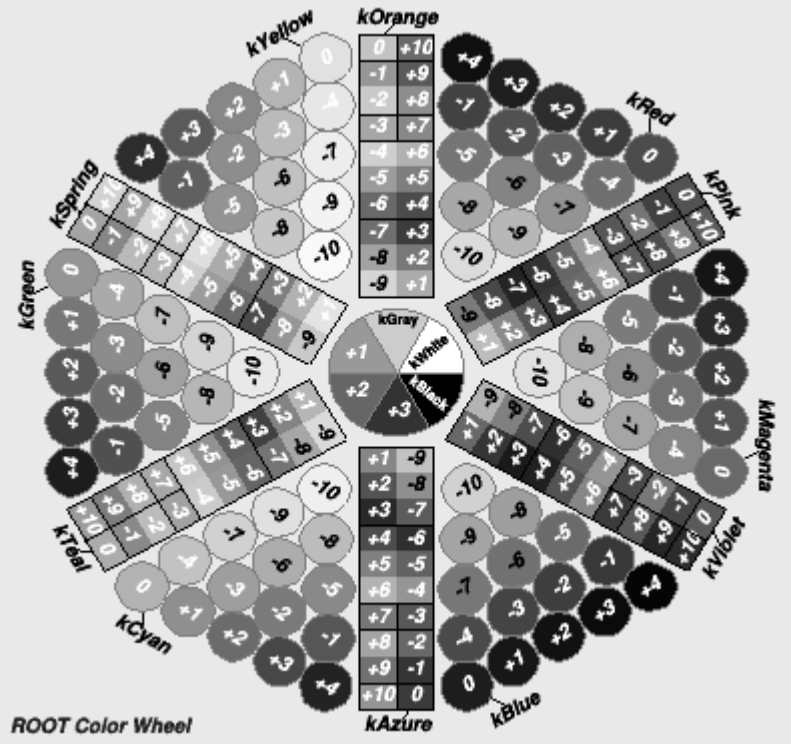
TColorWheel *w = new TColorWheel(); w->Draw();



myObject.SetFillColor(kRed);
myObject.SetFillColor(kGreen+2);

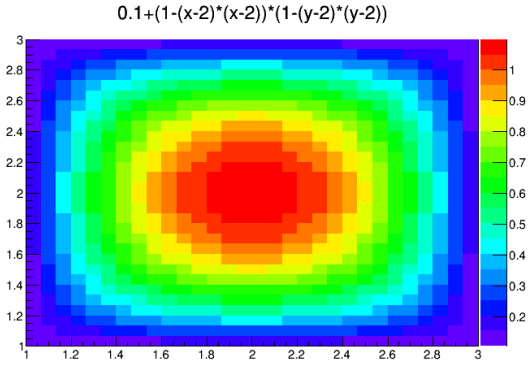
TColor

```
w->GetCanvas()->SetGrayscale();
w->GetCanvas()->Modified();
w->GetCanvas()->Update();
```

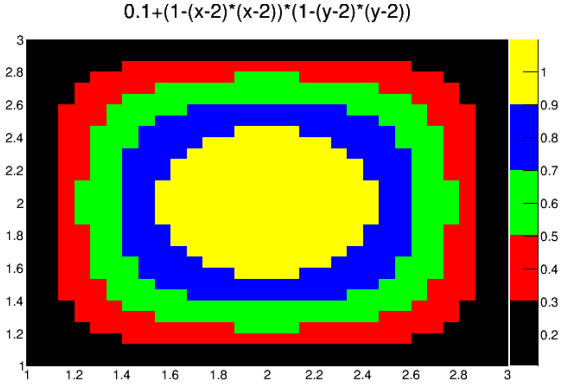


Color palettes

gStyle->SetPalette(1);

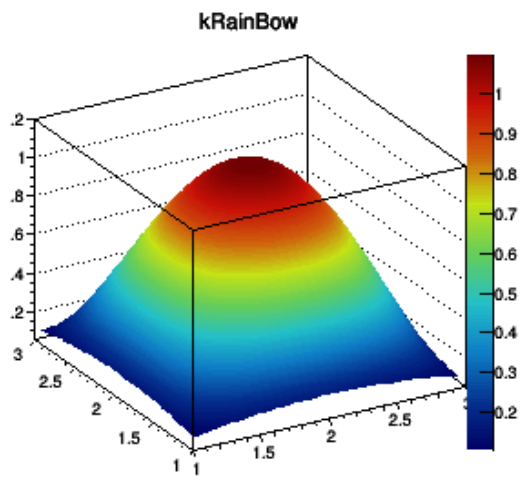


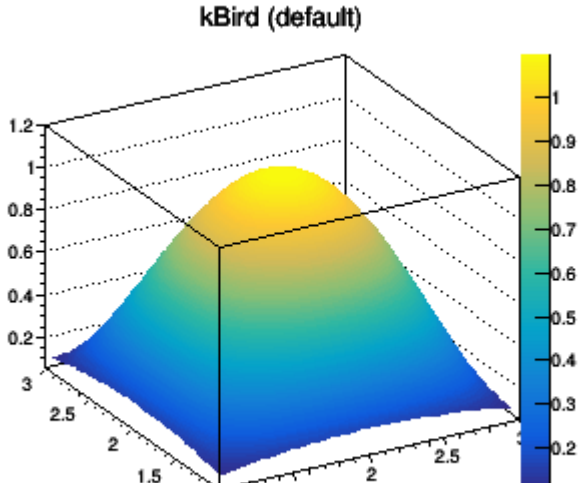

```
Int_t palette[5]={1,2,3,4,5};
gStyle->SetPalette(5,palette);
f1->Draw("colz");
```

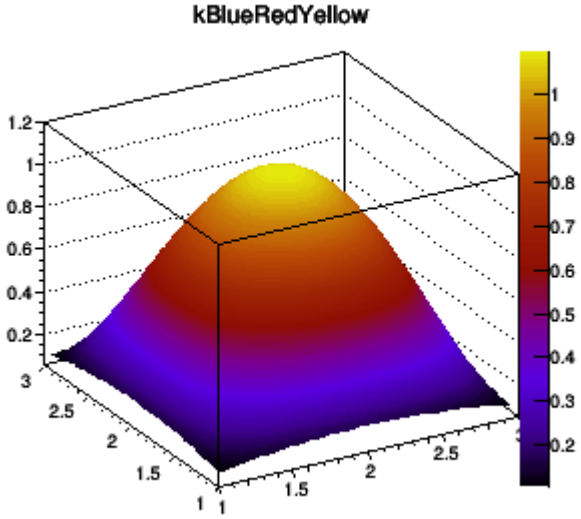


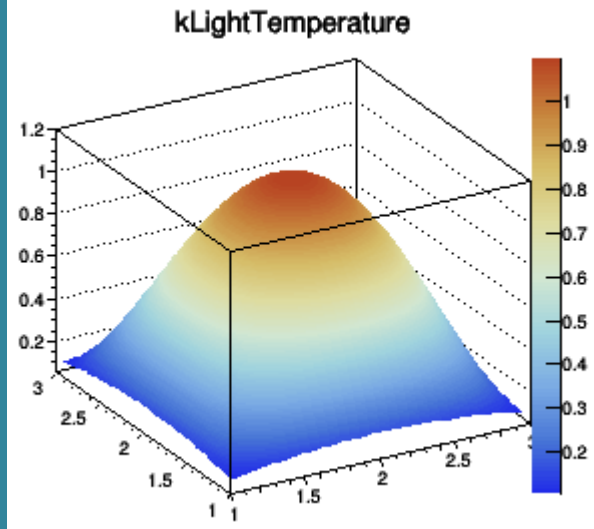
Palettes predefinidas

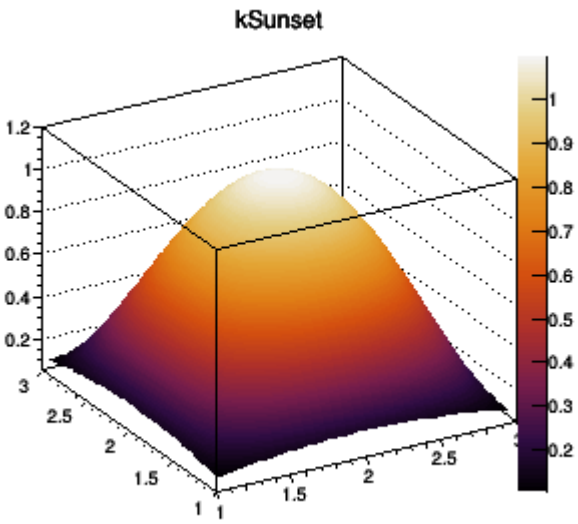
Desde versión 6.04 gStyle->SetPalette(num)











kRainBow=55
kBird=57
kLightTemperature=87
kSolar=100
kSunset=103

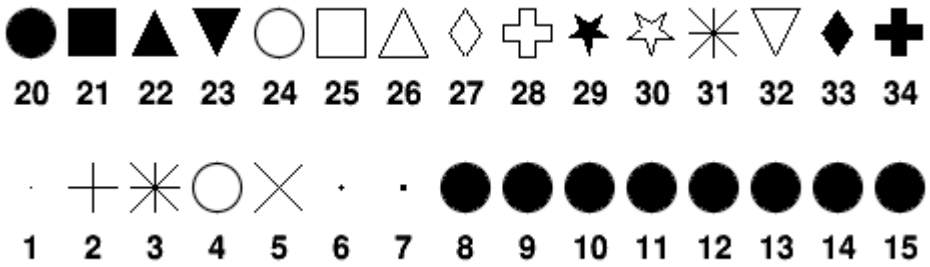
y muchas más …

Markers

TAttMarker

Transparencia y color: hist->SetMarkerColorAlpha(kBlue,0.35);

SetMarkerStyle(num)



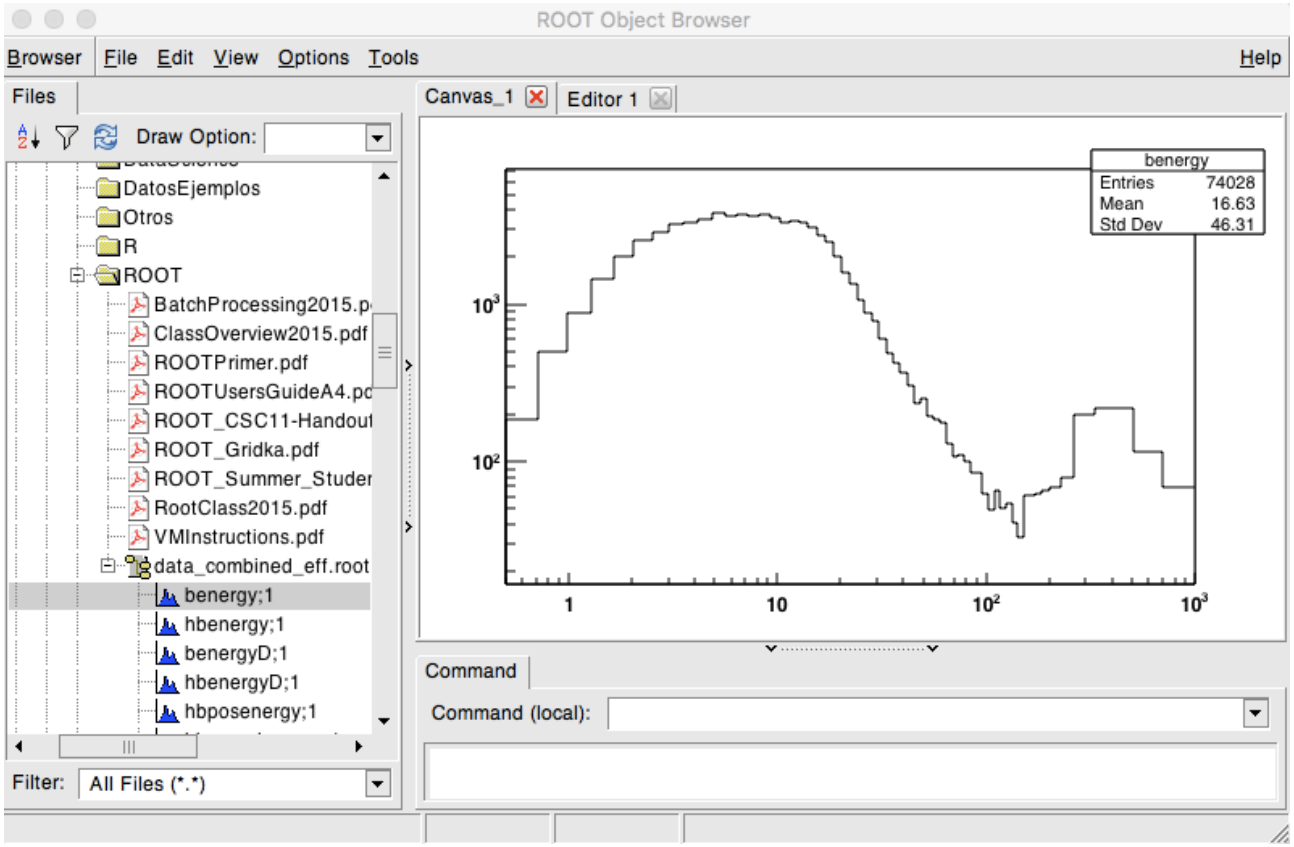
SetMarkerSize()

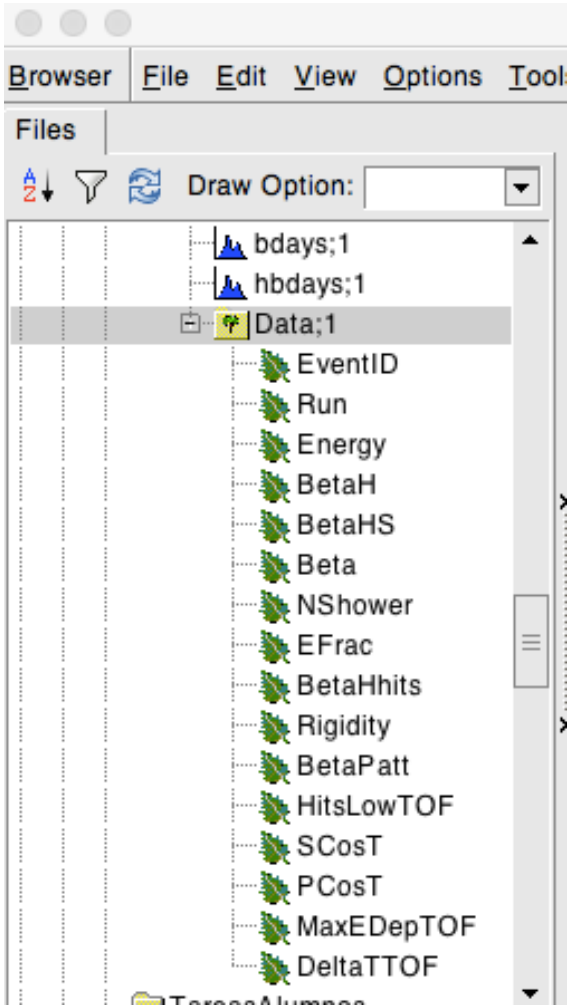
Marker size equal to 1 correspond to 8 pixels

kPlus=2
kCircle=4
kFullCircle=20
kFullSquare=21
kFullTriangleUp=22
kFullTriangleDown=23
kOpenCircle=24
kOpenSquare=25
kOpenTriangleUp=26
kOpenTriangleDown=32

TBrowser

new TBrowser ← Explorar interactivamente las ntuplas





Archivos

Grabar un objeto en un nuevo archivo

```
TFile* f = TFile::Open("myfile.root",”NEW”);
TH1D* h1 = new TH1D(“h1”,”h1”,100,-5.,5.);
h1->FillRandom(“gaus”);
h1->Write();
delete f;
```

Ver contenidos del archivo:

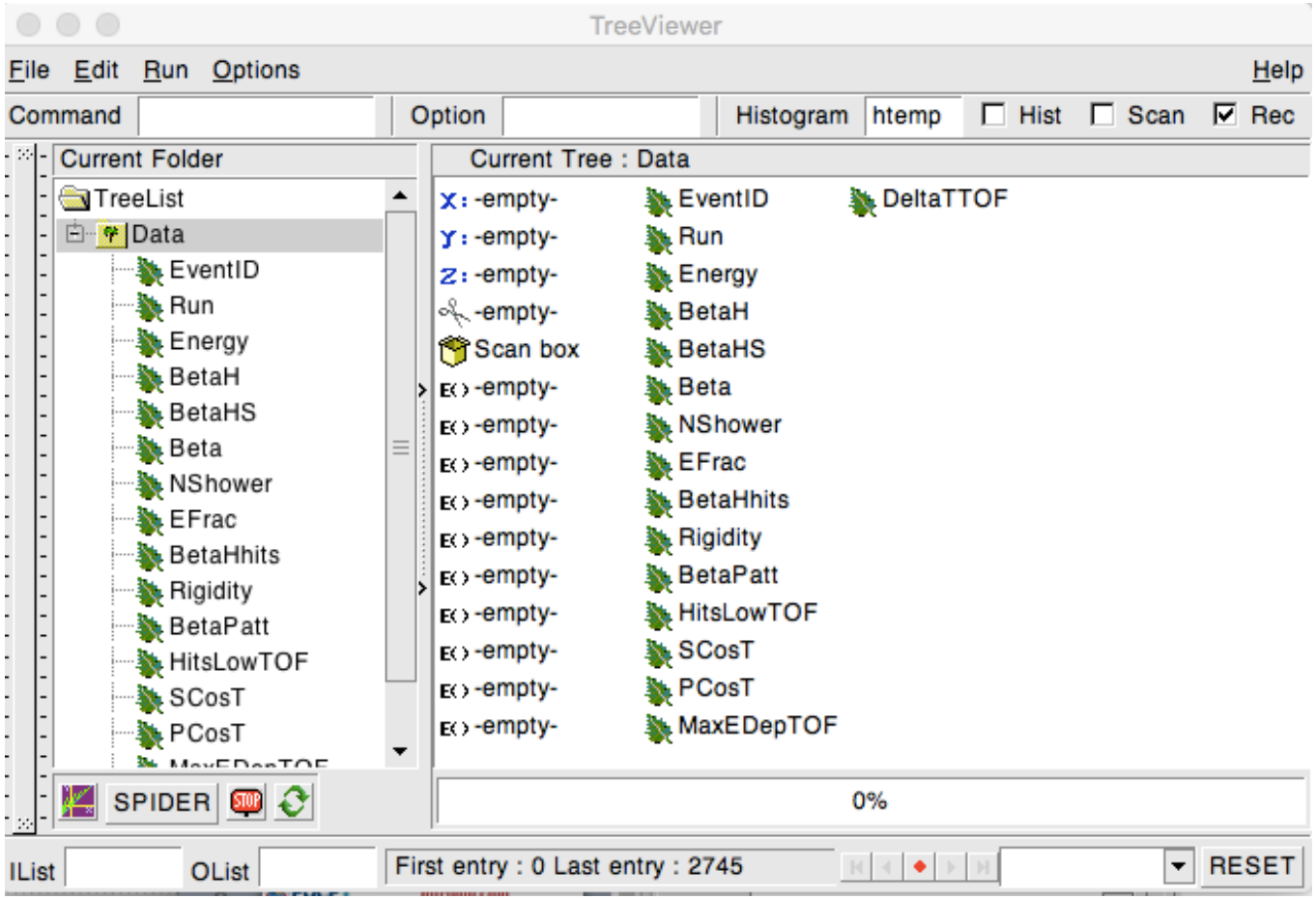
```
f->ls();
```

Abrir un archivo existente y leer un objeto

```
TFile* f = TFile::Open("myfile.root",”RECREATE”);
TH1* h = 0; f->GetObject("h1", h);                    //opción 1
TH1 *h = (TH1*) f->Get(“h1”);                         //opción 2
TH1 *h = (TH1*) f->GetObjectChecked(“h1”,”TH1”);      //opción 3
h->Draw();
delete f;
```

TreeViewer

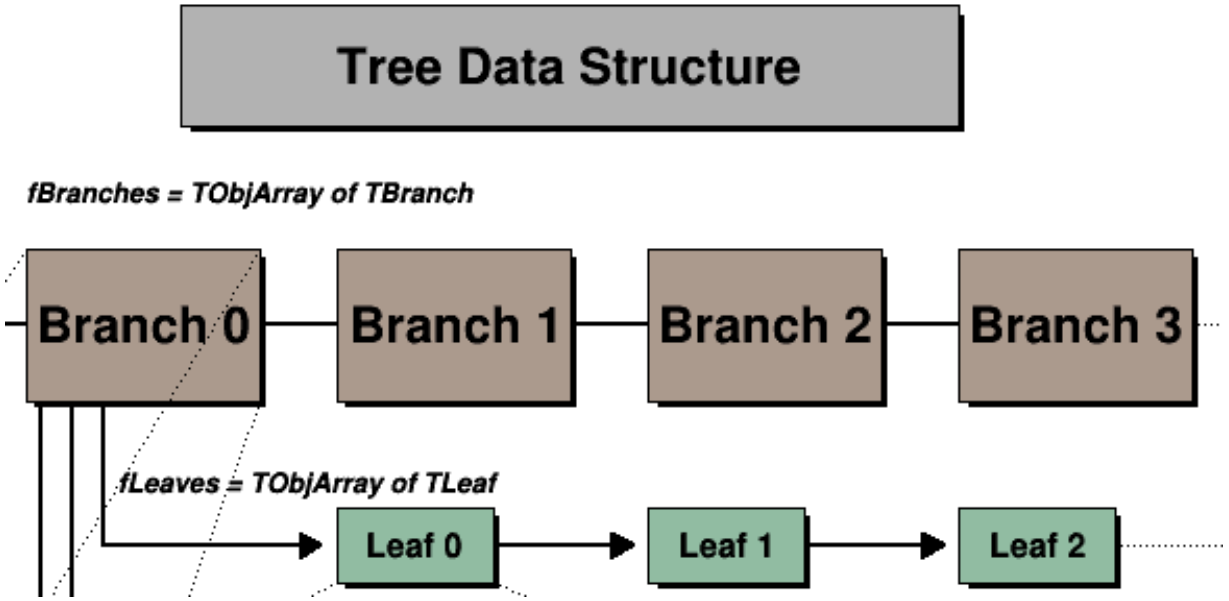
StartViewer



TTree

N-tupla: lista ordenada de números

TTree: lista ordenada de objetos de C++



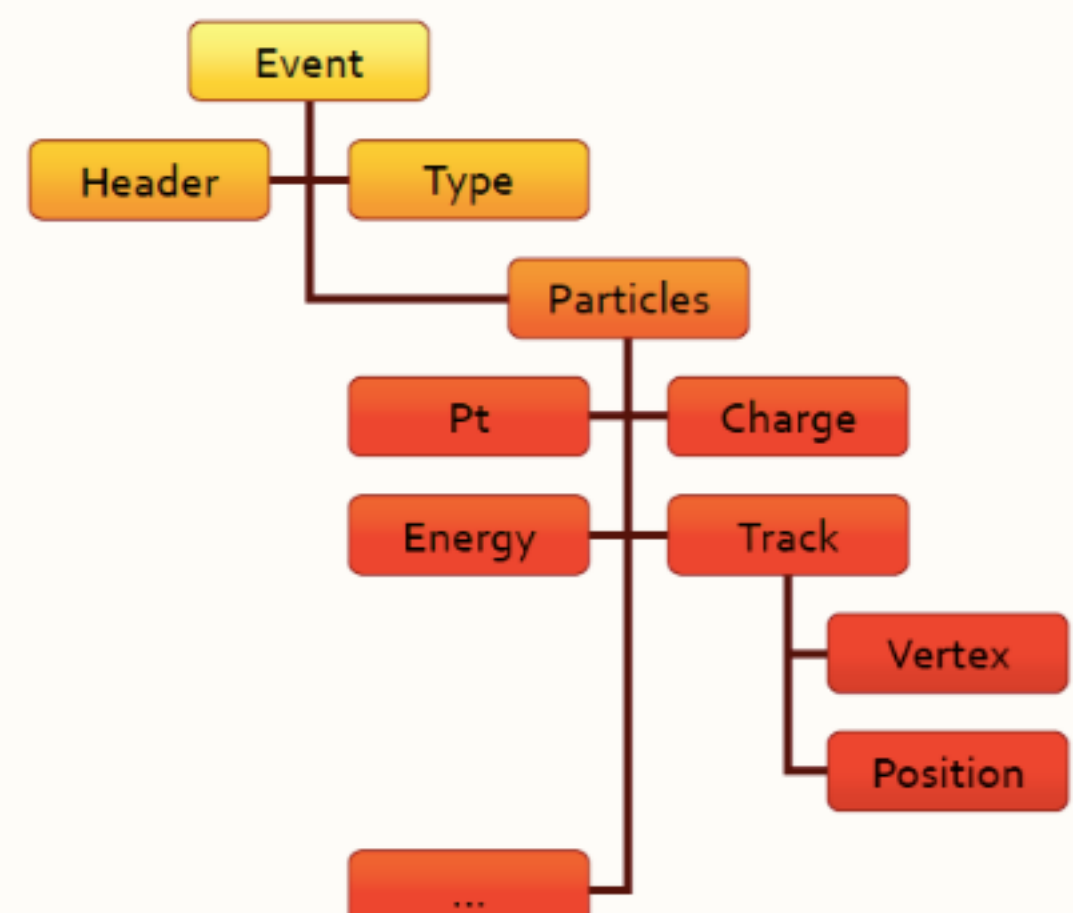
fType codes

- C : a character string
- B : an 8 bit signed integer
- b : an 8 bit unsigned integer
- S : a 16 bit signed short integer
- s : a 16 bit unsigned short integer
- I : a 32 bit signed integer
- i : a 32 bit unsigned integer
- F : a 32 bit floating point
- D : a 64 bit floating point
- TXXXX : a class name TXXXX

Branch: puede almacenar una simple variable, una lista de variables o una colección de objetos

Leaf: contenedores de datos del branch

TTree


Event
Header
Type
Particles
Pt
Charge
Energy
Track
Vertex
Position
...

TTree

Crear un Tree

```
TTree *tree = new TTree(name, title)

TBranch *branch = tree->Branch(branchname, address, leaflist, bufsize)
```

```
Float_t var;
TBranch *branch = tree->Branch("var", &var, "var/F");
```

Tipos de variables:

B : an 8 bit signed integer (Char_t)
b : an 8 bit unsigned integer (UChar_t)
S : a 16 bit signed integer (Short_t)
s : a 16 bit unsigned integer (UShort_t)
I : a 32 bit signed integer (Int_t)
i : a 32 bit unsigned integer (UInt_t)
F : a 32 bit floating point (Float_t)
D : a 64 bit floating point (Double_t)
L : a 64 bit signed integer (Long64_t)
l : a 64 bit unsigned integer (ULong64_t)
O : a boolean (Bool_t)

TTree

Añadir un Branch a un Tree existente:

```
TFile f("tree3.root", "update");

Float_t new_v;
TTree *t3 = (TTree*)f->Get("t3");
TBranch *newBranch = t3->Branch("new_v", &new_v, "new_v/F");

Long64_t nentries = t3->GetEntries(); // read the number of entries in the t3

for (Long64_t i = 0; i < nentries; i++) {
    new_v= gRandom->Gaus(0, 1);
    newBranch->Fill();
}

t3->Write("", TObject::kOverwrite); // save only the new version of the tree
```

TTree

```
// Define some simple structures
typedef struct {Float_t x,y,z;} POINT;
typedef struct {
   Int_t ntrack,nseg,nvertex;
   UInt_t flag;
   Float_t temperature;
} EVENTN;
static POINT point;
static EVENTN eventn;

// Create a ROOT Tree
TTree *tree = new TTree("T","An example of ROOT tree with a few branches");
tree->Branch("point",&point,"x:y:z");
tree->Branch("eventn",&eventn,"ntrack/I:nseg:nvertex:flag/i:temperature/F");
```

```
for ( Int_t i=0; i<1000; i++) {
Float_t random = gRandom->::Rndm(1);
point.y = 5*random;
point.z = 20*random;
eventn.ntrack  = Int_t(100*random);
eventn.nseg    = Int_t(2*eventn.ntrack);
eventn.nvertex = 1;
eventn.flag    = Int_t(random+0.5);
eventn.temperature = 20+random;
tree->Fill();}
```

Ttree: ejemplo

Inspeccionar un TTree:

TFile f(”data_combined_eff.root");
f.ls();

```
KEY: TH1D     benergy;1
KEY: TH1D     hbenergy;1
KEY: TH1D     benergyD;1
KEY: TH1D     hbenergyD;1
KEY: TH1D     hbposenergy;1
KEY: TH1D     hbposrelenergy;1
KEY: TH1D     cbenergy;1
KEY: TH1D     chbenergy;1
KEY: TH1D     chbposenergy;1
KEY: TH1D     chbposrelenergy;1
KEY: TH1D     c2benergy;1
KEY: TH1D     c2hbenergy;1
KEY: TH1D     c2hbposenergy;1
KEY: TH1D     c2hbposrelenergy;1
KEY: TH1D     trenergy;1
KEY: TH1D     htrenergy;1
KEY: TH2D     hpbetaEoR;1
KEY: TH2D     hbetahE;1
KEY: TH1D     hbetapatt;1
KEY: TH1D     hhbetapatt;1
KEY: TH1D     hhposbetapatt;1
KEY: TH1D     trdays;1
KEY: TH1D     htrdays;1
KEY: TH1D     bdays;1
KEY: TH1D     hbdays;1
KEY: TTree    Data;1
```

Data->Print();

```
root [3] Data->Print()
******************************************************************************
*Tree    :Data      :                                                        *
*Entries :     2746 : Total =         1299692 bytes  File  Size =    1103585 *
*        :          : Tree compression factor =   1.04                       *
******************************************************************************
*Br    0 :EventID   : Eventid/D                                              *
*Entries :     2746 : Total  Size=      81404 bytes  File Size  =      67554 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.01     *
*.............................................................................*
*Br    1 :Run       : Run/D                                                  *
*Entries :     2746 : Total  Size=      78872 bytes  File Size  =      58562 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.13     *
*.............................................................................*
*Br    2 :Energy    : Energy/D                                               *
*Entries :     2746 : Total  Size=      80771 bytes  File Size  =      67885 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    3 :BetaH     : BetaH/D                                                *
*Entries :     2746 : Total  Size=      80138 bytes  File Size  =      67181 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    4 :BetaHS    : BetaHS/D                                               *
*Entries :     2746 : Total  Size=      80771 bytes  File Size  =      67804 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    5 :Beta      : Beta/D                                                 *
*Entries :     2746 : Total  Size=      79505 bytes  File Size  =      66541 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    6 :NShower   : NShower/D                                              *
*Entries :     2746 : Total  Size=      81404 bytes  File Size  =      61275 *
*Baskets :      629 : Basket Size=      32000 bytes  Compression=   1.12     *
*.............................................................................*
```

Ttree: ejemplo

Inspeccionar un TTree:

Data->Show(0);

```
[root [12] Data->Show(0)
======> EVENT:0
 Eventid          = 995131
 Run              = 1.36742e+09
 Energy           = 54.3314
 BetaH            = 1.05725
 BetaHS           = 1.05725
 Beta             = 1.06938
 NShower          = 1
 EFrac            = 1
 BetaHhits        = 4
 Rigidity         = 146.939
 BetaPatt         = 0
 HitsLowTOF       = 3
 SCosT            = -0.99089
 PCosT            = -0.990634
 MaxEDepTOF       = 0.490645
 DeltaTTOF        = 8.44711
```

Data->Scan("Energy:Beta:NShower");

```
*************************************************
*    Row   *    Energy *      Beta *   NShower *
*************************************************
*        0 * 54.331386 * 1.0693808 *         1 *
*        1 * 55.488609 * 1.0101925 *         1 *
*        2 * 265.00006 * 0.9789484 *         1 *
*        3 * 500.95790 * 1.0512663 *         1 *
*        4 * 54.238796 * -1.256966 *         3 *
```

Data->Scan() //8 primeros

Data->Scan("*") //todos

Ttree: ejemplo

Dibujar directamente desde el TTree:

TFile f(”data_combined_eff.root");

Histograma:

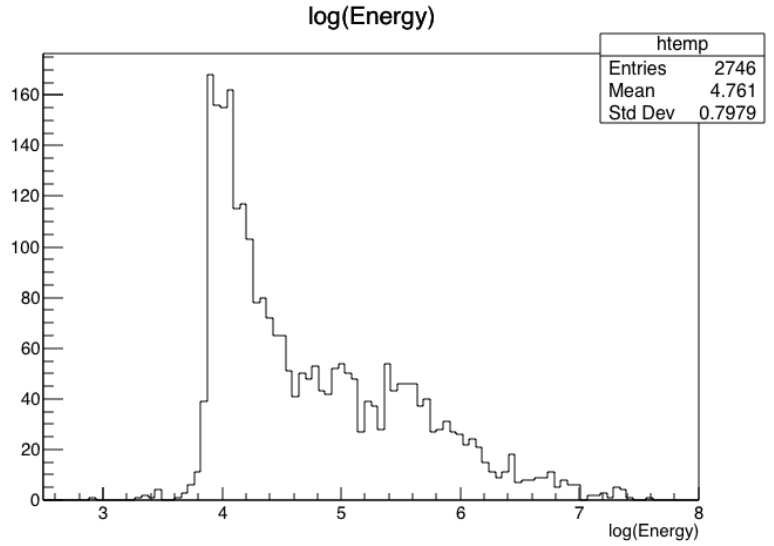
Data->Draw(“log(Energy)”);

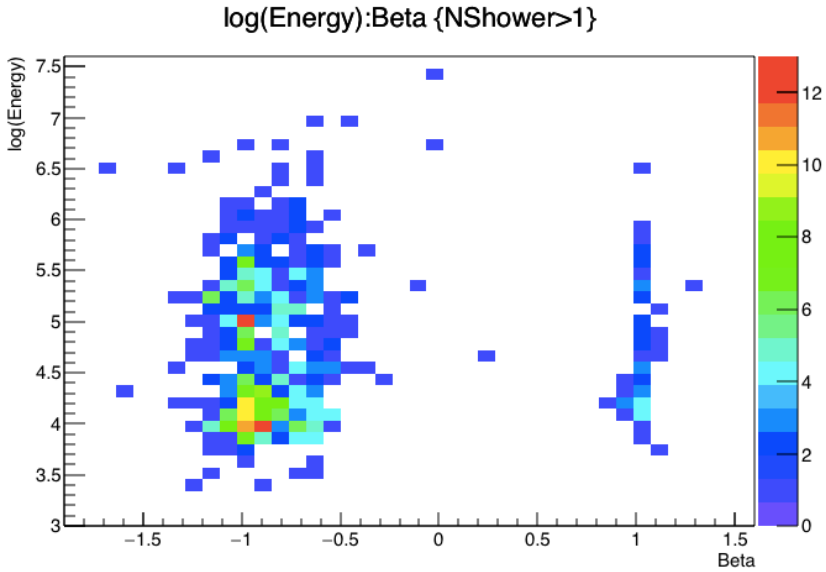
Scattered plot

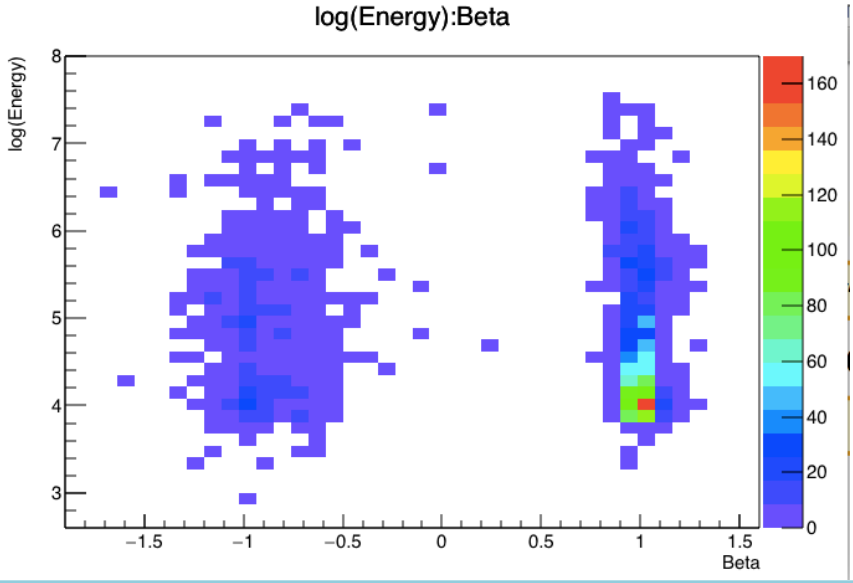
Data->Draw(“log(Energy):Beta”,””,”colz”);

Agregar condición

Data->Draw(“log(Energy):Beta”,”Beta>0”,”colz”);







Ttree: ejemplo

Loop de datos TTree:

```
TFile f(”data_combined_eff.root");

  Ttree *tree=0;
  f.GetObject(”Data”,tree);

  float energy=0;
  tree->SetBranchAddress(“Energy”,&energy);

  for (int i = 0; i < tree->GetEntries(); i++) {
  tree->GetEntry(i);
  cout<<“energy ”<<energy<<endl;
  }
```

Si hubiera un objeto habría que crearlo antes de establecer la dirección del branch, e.g.:
Event *event = new Event();

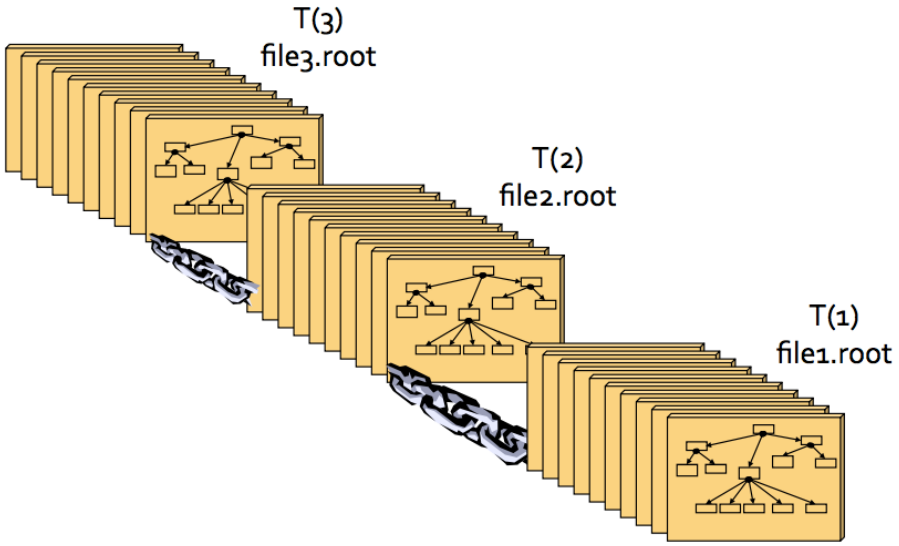
Ttree: ejemplo

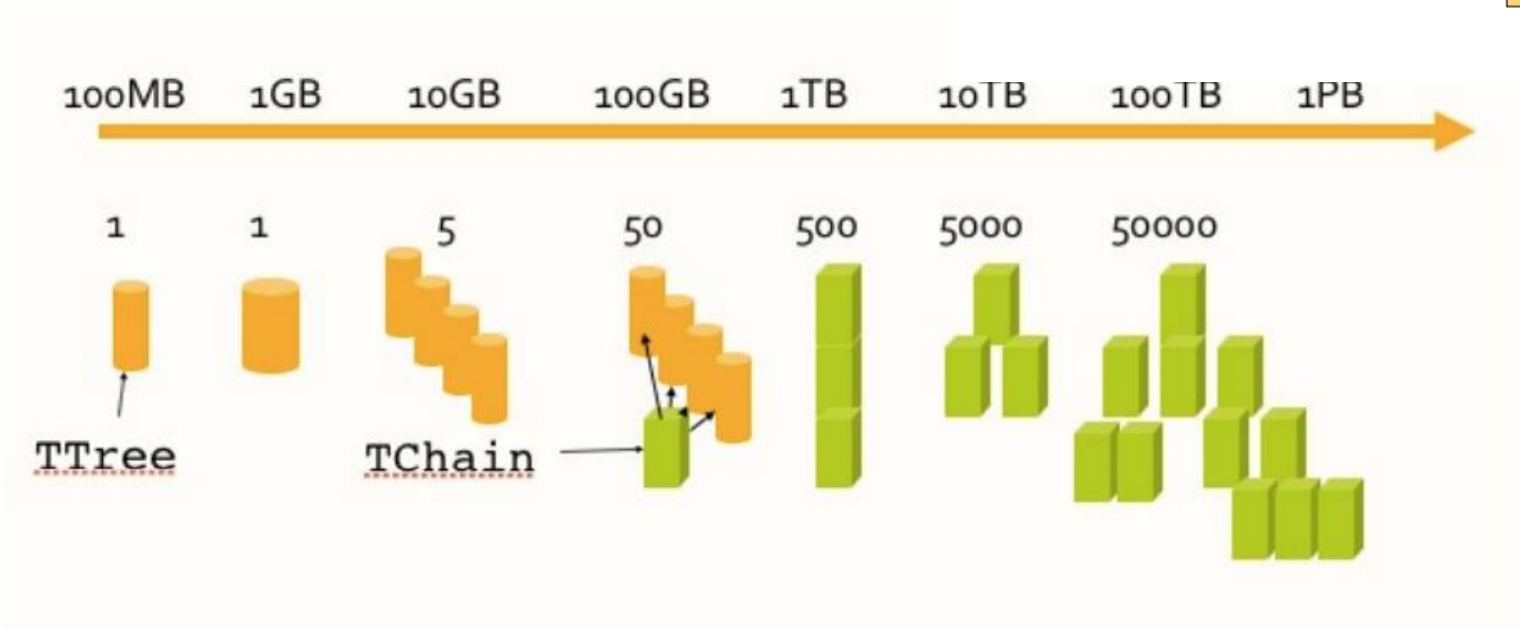
En general se con GetEntry() se leen todos los branches, para agilizar se pueden leer solo los deseados:

```
TClonesArray* myMuons = 0;
// disable all branches
myTree->SetBranchStatus("*", 0);
// re-enable the "muon" branches
myTree->SetBranchStatus("muon*", 1);
myTree->SetBranchAddress("muon", &myMuons);
// now read (access) only the "muon" branches
for (Long64_t i = 0; i < myTree->GetEntries(); ++i) {
     myTree->GetEntry(i);
```

TChain

Un **TFile** contiene normalmente un Ttree
Un **TChain** es una colección de TTrees o TChains





TChain

TChain: colección de TTrees iguales (mismo nombre) contenidos en archivos diferentes:

```
TChain chain("T"); // argument: tree name
chain.Add("file1.root");
chain.Add("file2.root");
chain.Add("file3.root");
```

Luego chain se puede utilizar como un TTree.

pyroot

```
import ROOT

# Open the file.
myfile = ROOT.TFile( 'experiment.root' )

# Retrieve the n-tuple of interest.
mychain = ROOT.gDirectory.Get( 'tree1' )
entries = mychain.GetEntriesFast()

# Create a 2D histogram
myHist = ROOT.TH2D("hist2D","chi2 vs ebeam",100,0,20,100,149,151)
myHist.GetXaxis().SetTitle("chi2")
myHist.GetYaxis().SetTitle("ebeam [GeV]")

for jentry in xrange( entries ):

   # Copy next entry into memory and verify.
   nb = mychain.GetEntry( jentry )
   if nb <= 0:
      continue

   # Fetch the variables from the entry and fill the histogram.
   chi2 = mychain.chi2
   ebeam = mychain.ebeam
   myHist.Fill(chi2,ebeam)

# Display the scatterplot.
myHist.Draw()
```