



Corso di Laboratorio di Meccanica e Termodinamica

Modulo ROOT

Lezione Introduttiva

Silvia Arcelli

20 Marzo 2024

Laboratorio di Meccanica e Termodinamica- Modulo ROOT

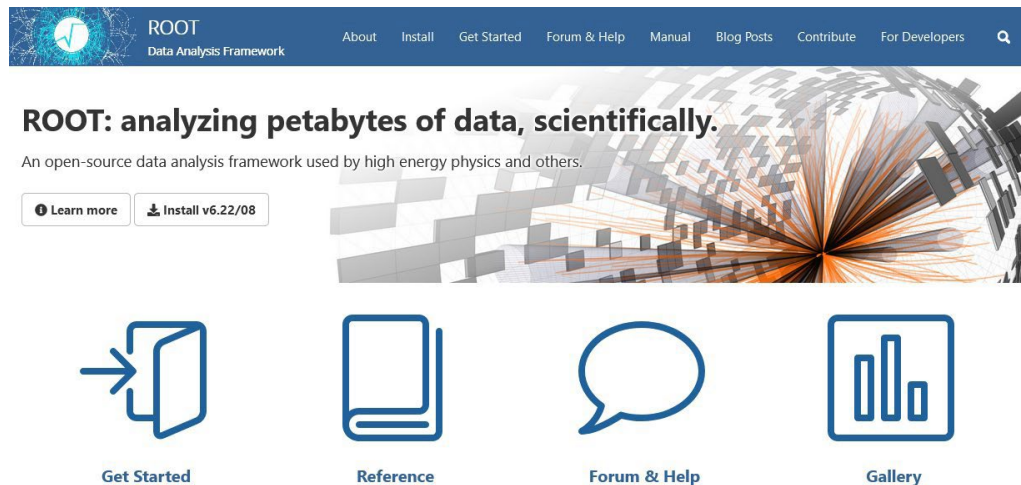
Il modulo prevede una serie di lezioni frontali, contenuti:

- Struttura Generale**: l'interfaccia utente (linea di comando, uso di macro, interfaccia grafica).
- Rappresentazione di dati sperimentali con ROOT**: classi per istogrammi, grafici e funzioni e loro utilizzo per gli esperimenti svolti in laboratorio, con esempi.
- Generazione Monte Carlo**: Nozioni di base (Generatori di numeri pseudocasuali, metodo del rigetto, metodo della trasformazione inversa). Funzionalità di ROOT per la generazione Monte Carlo.
- Esempi Monte Carlo** sulla distribuzione uniforme, gaussiana, esponenziale, binomiale. Verifica di risultati derivabili a priori (distribuzione Poissoniana e Gaussiana come limite della distribuzione Binomiale, etc.).

Per l'esame: un quesito allo scritto (elaborazione di un frammento di codice, 6 punti su 30) e **domande all'orale** (metodo Monte Carlo+ eventuale discussione scritto).

ROOT

ROOT è un pacchetto software di **calcolo scientifico e analisi dati** sviluppato al CERN (il Centro di Fisica Subnucleare di Ginevra), ed è attualmente utilizzato in molti laboratori di ricerca nazionali ed internazionali.



ROOT Home page:
<https://root.cern.ch>

ROOT Team:

- autori principali: Rene Brun & Fons Rademakers
- una decina di collaboratori “senior”
- >200 “contributors” (utenti)

✓-1

ROOT enables *statistically sound* scientific analyses and visualization of large amounts of data: today, more than 1 exabyte (1,000,000,000 gigabyte) are stored in ROOT files. The Higgs was found with ROOT!



As *high-performance* software, ROOT is written mainly in C++. You can use it on Linux, macOS, or Windows; it works out of the box. ROOT is open source: use it freely, modify it, contribute to it!

\$ _

ROOT comes with an incredible C++ interpreter, ideal for *fast prototyping*. Don't like C++? ROOT integrates super-smoothly with Python thanks to its unique dynamic and powerful Python \rightleftharpoons C++ binding. Or what about using ROOT in a Jupyter notebook?

ROOT

Framework di **Analisi Dati Object Oriented**, con moltissime funzionalità:

- **Scritto in C++**, come la maggior parte del software utilizzato nei grandi esperimenti di fisica dell'attuale generazione
- E' **Open Source**: sorgente ed eseguibile "free", scaricabili dall'utente
- **Molte piattaforme** supportate
- **Ottimale** - fra l'altro - per l'analisi e per la gestione **di grandi volumi di dati**: molti esperimenti di fisica delle particelle alle alte energie devono gestire **decine di PB/anno di dati ("BIG DATA")**

ROOT-Perché vi può servire

Una delle attività più frequenti in fisica è acquisire (ma anche simulare) dati, manipolarli, analizzarli in termini statistici e confrontarli con un modello eventualmente dipendente da un certo numero di parametri, estraendone una stima con relative incertezze e un grado di accordo in termini quantitativi.

ROOT (fra le molte cose che offre) consente di :

- Visualizzarli (grafici, istogrammi)
- Analizzarli (calcoli, fit a un modello)
- Simulare dati attraverso distribuzioni basate su numeri pseudo-random
- Salvare i risultati in diversi formati (persistenza)

ROOT si interfaccia naturalmente con programmi C++ (oltre che supportare altri linguaggi, quali Python, R). Un possibile modo per prendere ulteriormente confidenza e utilizzare il C++, che è fra uno dei linguaggi più potenti e diffusi, non solo in ambiente scientifico.

ROOT-Documentazione e Guide

Per iniziare, vi segnalo una guida molto «pragmatica» e direi efficace :

Il Root Primer:

<https://root.cern.ch/root/html/doc/guides/primer/ROOTPrimer.html>

Molti programmi di esempio:

ROOT Tutorials: https://root.cern/doc/master/group_Tutorials.html

Altra documentazione utile **di livello più dettagliato:**

- **Manuale Completo:** <https://root.cern/manual/>
- **Reference Guide:** <https://root.cern/doc/master/index.html>

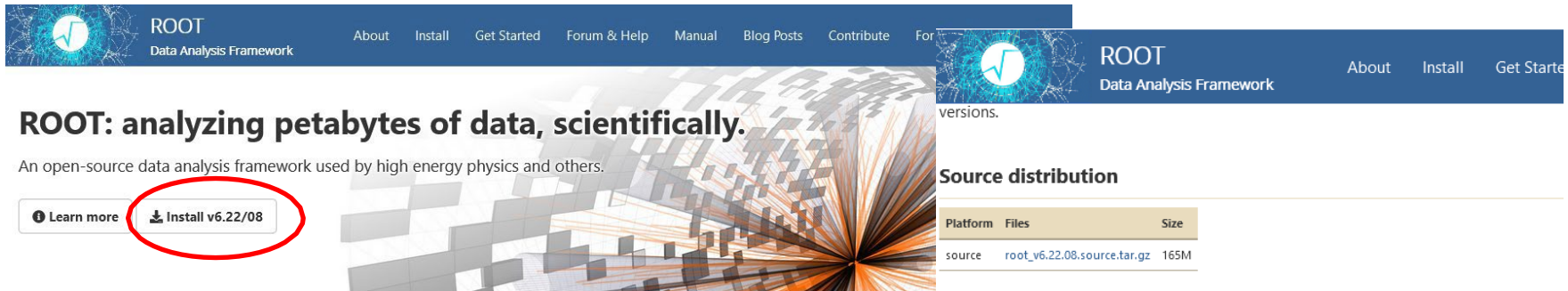
ROOT-Installazione

- ROOT funziona su **molte piattaforme** (Linux, Mac-OS, Windows), anche se sistemi di tipo Linux sono in effetti il riferimento (i più usati in ambiente scientifico).
- **Se avete Windows**, usate WSL con un X-server (per esempio MobaXterm), come da istruzioni del Prof. Giacomini
- Il sito di ROOT è piuttosto “user -friendly”. Per l’installazione, avete a disposizione due opzioni:
 - 1) Scaricarvi la **distribuzione binaria** (operazione veloce), se compatibile con la vostra piattaforma.
 - 2) Scaricarvi il **sorgente e compilarlo** (operazione un po’ più complessa)

Verificata la compatibilità della vostra piattaforma con il binario di ROOT, **consiglio vivamente la prima opzione.**

ROOT-Installazione

Per installare la versione più recente della serie 6, andate sul sito di ROOT:



ROOT Data Analysis Framework

About Install Get Started Forum & Help Manual Blog Posts Contribute For

ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

[Learn more](#) [Install v6.22/08](#)

Source distribution

Platform	Files	Size
source	root_v6.22.08.source.tar.gz	165M

Binary distributions

Platform	Files	Size
CentOS 7	root_v6.22.08.Linux-centos7-x86_64-gcc4.8.tar.gz	186M
Fedora 30	root_v6.22.08.Linux-fedora30-x86_64-gcc9.3.tar.gz	226M
Fedora 31	root_v6.22.08.Linux-fedora31-x86_64-gcc9.3.tar.gz	226M
Fedora 32	root_v6.22.08.Linux-fedora32-x86_64-gcc10.2.tar.gz	228M
Ubuntu 16	root_v6.22.08.Linux-ubuntu16-x86_64-gcc5.4.tar.gz	201M
Ubuntu 18	root_v6.22.08.Linux-ubuntu18-x86_64-gcc7.5.tar.gz	219M
Ubuntu 19	root_v6.22.08.Linux-ubuntu19-x86_64-gcc9.2.tar.gz	224M
Ubuntu 20	root_v6.22.08.Linux-ubuntu20-x86_64-gcc9.3.tar.gz	225M
macOS 10.14 x86_64 Xcode 10	root_v6.22.08.macos-10.14-x86_64-clang100.pkg	315M
macOS 10.14 x86_64 Xcode 10	root_v6.22.08.macos-10.14-x86_64-clang100.tar.gz	200M
macOS 10.15 x86_64 Xcode 12	root_v6.22.08.macos-10.15-x86_64-clang120.pkg	309M
macOS 10.15 x86_64 Xcode 12	root_v6.22.08.macos-10.15-x86_64-clang120.tar.gz	198M
macOS 11.2 x86_64 Xcode 12	root_v6.22.08.macos-11.2-x86_64-clang120.pkg	309M
macOS 11.2 x86_64 Xcode 12	root_v6.22.08.macos-11.2-x86_64-clang120.tar.gz	198M
preview Windows Visual Studio 2019 (debug)	root_v6.22.08.win32.vc16.debug.exe	154M
preview Windows Visual Studio 2019 (debug)	root_v6.22.08.win32.vc16.debug.zip	227M
preview Windows Visual Studio 2019	root_v6.22.08.win32.vc16.exe	84M
preview Windows Visual Studio 2019	root_v6.22.08.win32.vc16.zip	113M

- Troverete diversi **eseguibili** per varie piattaforme (**Binary Distribution**), o il sorgente (**Source Distribution**).
- A seconda di cosa avete deciso scaricate o il binario (se è compatibile con la vostra piattaforma) o il codice sorgente.

ROOT-Installazione

- Nel caso di installazione da sorgente, per compilare, seguite le istruzioni che trovate al link <https://root.cern.ch/building-root>.
- **controllate di avere le librerie necessarie** al link <https://root.cern/install/dependencies/>
- su **Virtuale** troverete una mini-guida per l'installazione di root su UBUNTU a partire dal binario o dal sorgente, e per Mac. Se avete difficoltà segnalatemelo (silvia.arcelli@unibo.it).
- UBUNTU è uno dei SO più diffusi. Vi suggerisco **vivamente** di installare **solo** versioni «**Long Term Support**» (LTS): ad es. **UBUNTU 20.04** .

ROOT-Nella lezione di oggi

- **Struttura Generale**
- **Primiissime considerazioni su Istogrammi e Grafici in ROOT**
- **L'interfaccia Utente di ROOT**
 - **Command Line Interpreter**
 - **Macros**
 - **Intefaccia Grafica (GUI)**

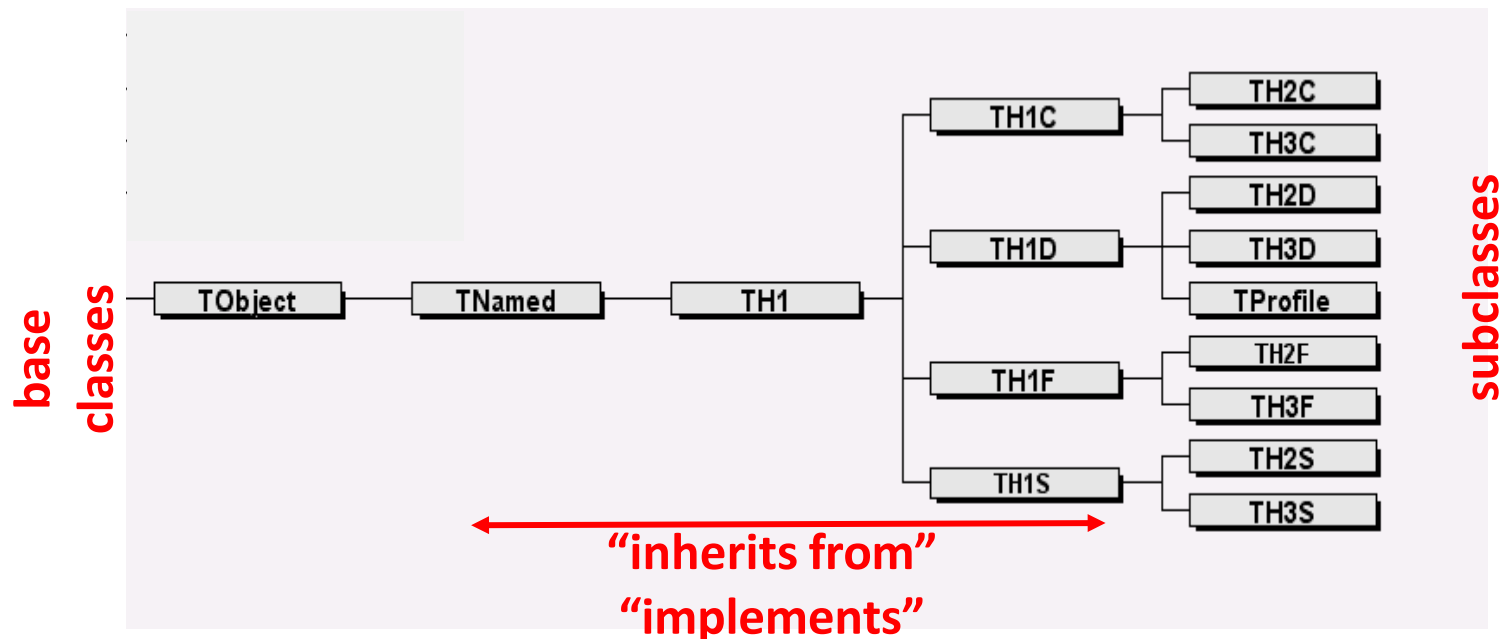
ROOT-Struttura Generale

Circa 650'000 linee di codice Object Oriented in C++, organizzato in una sessantina di librerie:

- Classi Base
- Classi per grafici e istogrammi (1,2,3 & n-D)
- Analisi avanzata di grafici e istogrammi (ad es. fit di dati sperimentali)
- Classi per la generazione di numeri random (metodo Monte Carlo)
- Classi per user interface (GUI, interattiva, macros)
- Strutture ottimizzate per la gestione e l'analisi di grandi volumi di dati dello stesso tipo, strutturalmente complessi : i Trees
- Classi di Grafica (2-D e 3-D)
- Classi per manipolazione di Matrici, Vettori , funzioni matematiche, Analisi statistica avanzata, Reti Neurali
- Classi Container (come la STL)
- Interfaccia con il sistema operativo
- Database SQL, Networking, Parallel Processing

ROOT-Struttura Generale

- Migliaia di classi (il cui nome inizia sempre per **T maiuscola**) con funzionalità molto diverse. Implementazioni che utilizzano in maniera massiccia **ereditarietà virtuale** e **polimorfismo**. Esempio per la **classe di Istogrammi TH1**:



N.B. L'ultima lettera dopo nel nome TH* indica il **tipo utilizzato** per rappresentare **gli ingressi** dell'istogramma (non il **tipo** della variabile istogrammata!)

ROOT-Struttura Generale

- oggetti della classe figlia sono anche del tipo della classe madre:
un istogramma TH1D „is a“ TObject (l'oggetto base di ROOT)
- Le classi figlie **ereditano** tutti i membri /metodi (pubblici e protected) delle classi madre:

```
TH1D* hist=new TH1D("hpx","example",100,0,10);  
cout <<"histogram name:" << hist->GetName()<<endl;
```

TH1D eredita la member function GetName() di TNamed
- Le classi figlie eventualmente **ridefiniscono** i metodi dichiarati virtuali (e con il polimorfismo al run-time si riconosce correttamente il metodo da invocare):

```
TObject* ob=new TH2I("map","example",50,0,1000, 50,0,1000);  
ob->Print(); //automatically calls TH2I::Print();  
TObject::Print() "sovrascritto" da TH1::Print()
```

ISTOGRAMMI in ROOT

- Un istogramma vi consente di visualizzare e di analizzare la distribuzione delle occorrenze di una o più variabili fisiche (per esempio gli esiti di una misura ripetuta più volte, o prevista in base a un calcolo, o da una generazione Monte Carlo). ROOT offre delle classi ad hoc per questo compito: **istogrammi di una variabile (1D), 2 variabili (2D) e tre variabili (3D)**.
- Ogni istogramma che create è un **OGGETTO** e come tale, va:
 - Costruito → **Costruttori**
 - Manipolato → **Metodi della Classe**
 - **Eventualmente distrutto** (ma ROOT lo farà per voi nella maggior parte dei casi, https://root.cern/manual/object_ownership/)
- Potete creare **istanze** o, nel caso di allocazione dinamica, **puntatori a istanze** (la seconda opzione è quella più utilizzata in ROOT, molti metodi utilizzano puntatori a oggetti e non pure istanze)

Istogrammi 1-D: TH1*

```
TH1F *name = new TH1F("name","Title", NxBins, xmin, xmax);
```

Esempio:

```
//dichiarazione istogramma (costruzione)
```

```
TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
```

```
for(Int_t i=0;i<1000;i++){
```

```
//variabile x da lettura dati da
```

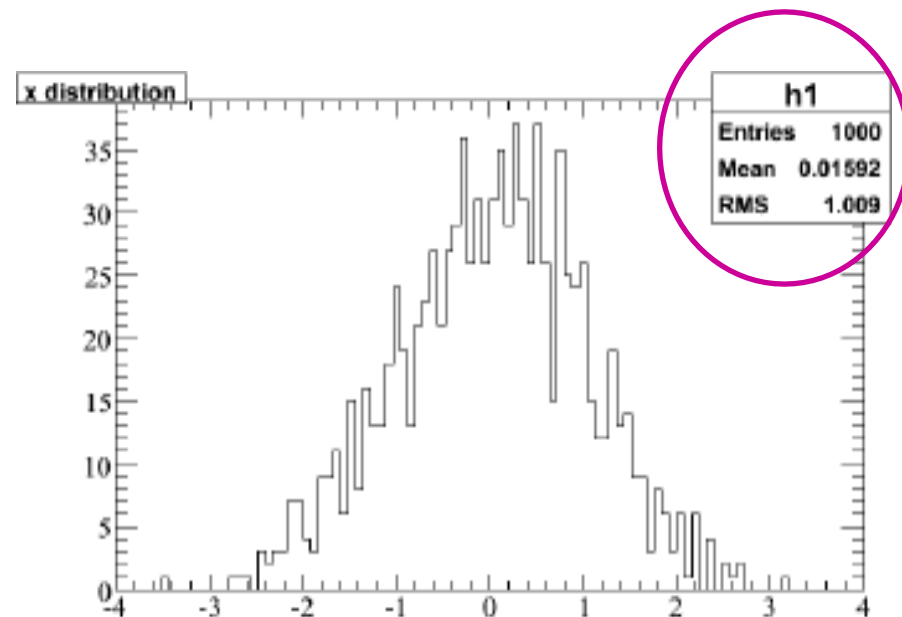
```
//file o generazione Monte Carlo
```

```
h1->Fill(x); //filling
```

```
}
```

```
h1->Draw(); //drawing
```

(Possibile anche impostare
una larghezza dei bin non
costante)



Istogrammi 2-D: TH2*

```
TH2F *name = new TH2F("name","Title", NxBins, xmin, xmax,  
    NyBins, ymin, ymax);
```

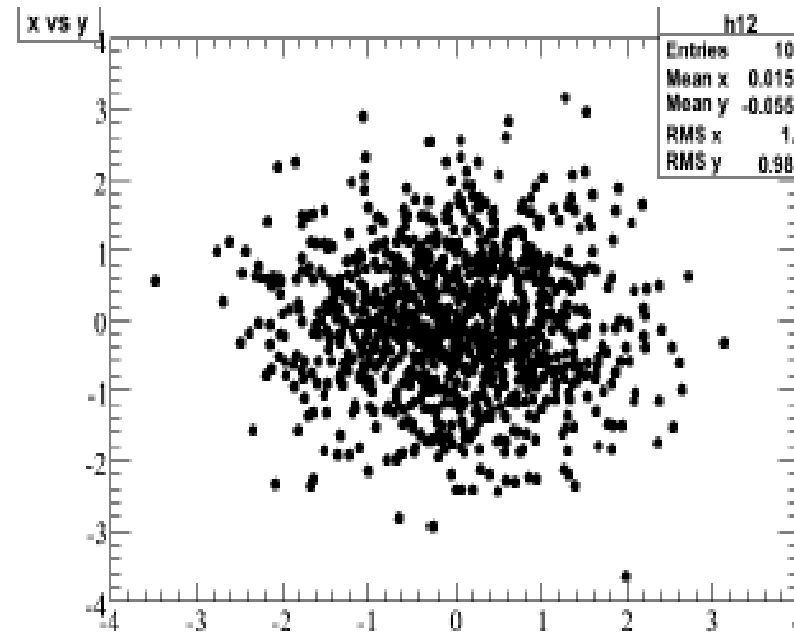
Esempio:

```
TH2F *h12 = new TH2F("h12","y vs x",100,-4,4,100,-4,4);  
for(Int_t i=0;i<1000;i++){  
    //...  
    h12->Fill(x,y); //filling, 2 variabili  
    //...  
}  
h12->Draw();
```

Possibilità di fare **Proiezioni** rispetto a una delle variabili (per istogrammi 2 e 3-D):

```
h->ProjectionX()
```

```
h->ProjectionY()
```

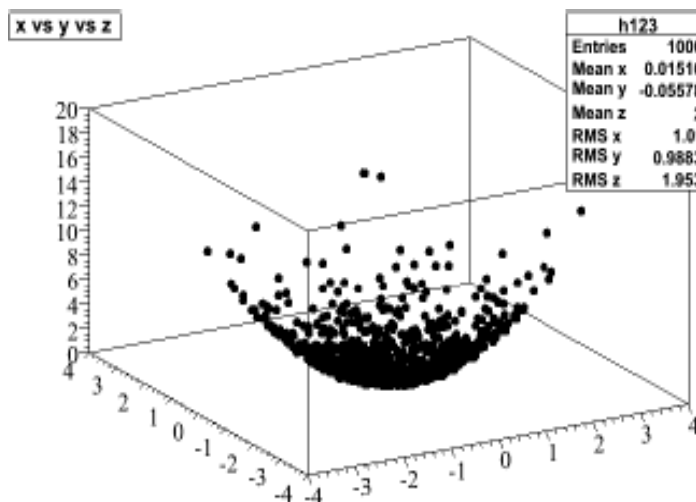


Istogrammi 3-D: TH3*

```
TH3F *h123 = new TH3F("name","Title", NxBins, xmin, xmax,  
    NyBins, ymin, ymax, NzBins, zmin, zmax);
```

Esempio:

```
TH3F *h123 = new TH3F("h123","x vs y vs z",100,-4,4,100, -4,  
    4,100,0,20);  
h123->Fill(x,y,z);  
h123->Draw();
```



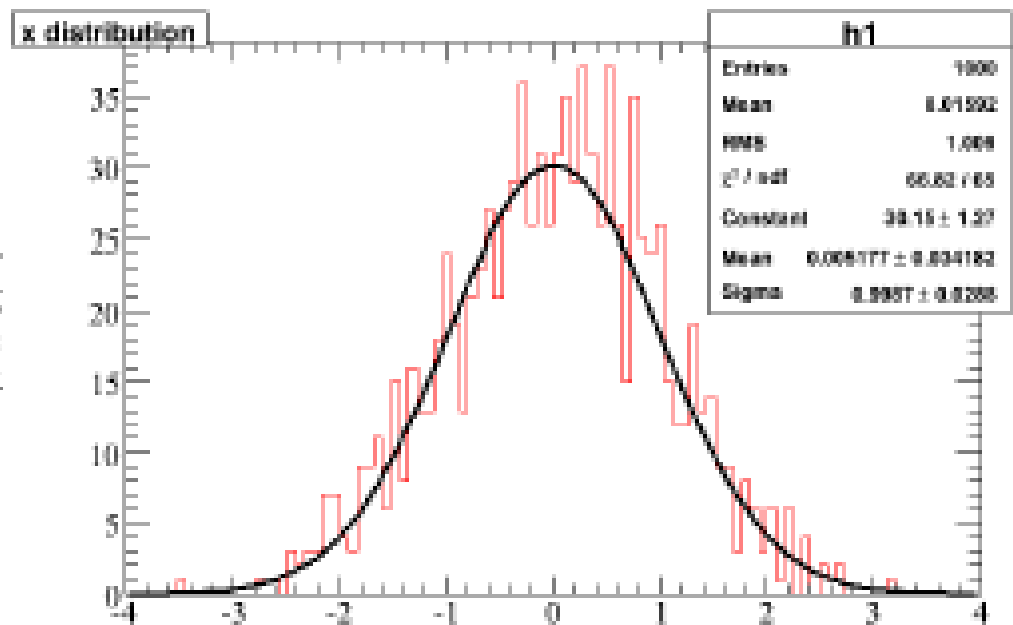
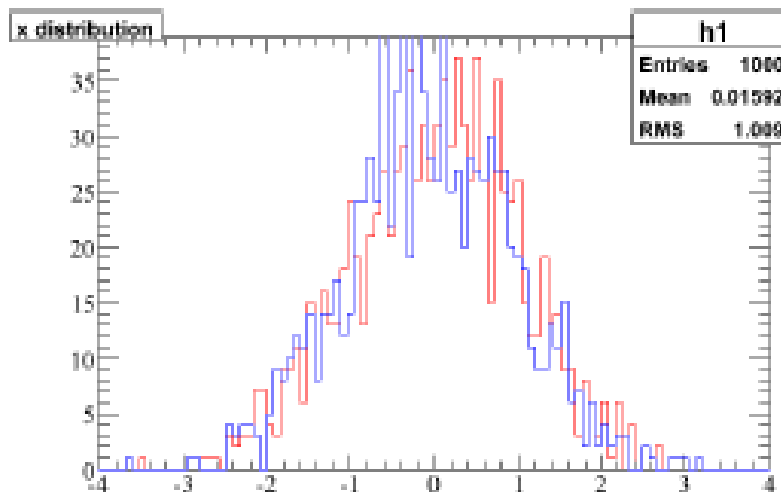
Per istogrammi N-dimensionali con $N > 3$, classe THNSparse

Istogrammi: Alcuni Metodi

Sovrapporre istogrammi per confrontarli:

```
h1->Draw() ;
```

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



Fare un Fit a una distribuzione:

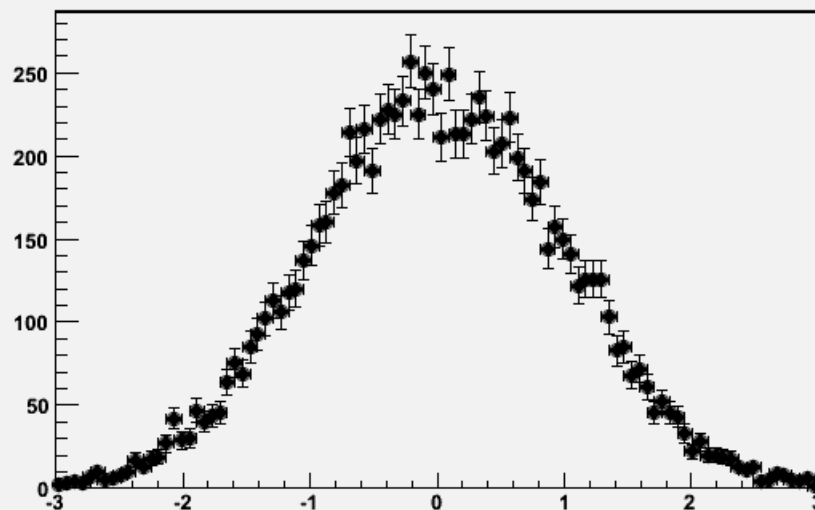
```
h1->Fit("gaus") ;
```

Istogrammi- Drawing Options

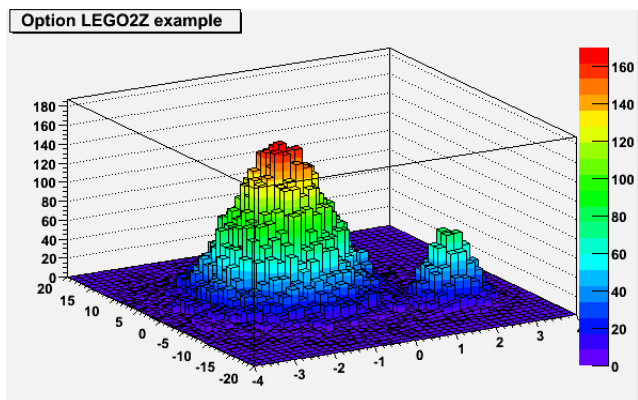
Alcune opzioni del metodo Draw():

- “E”: Mostra gli Errori
- “HIST”: disegna solo l’istogramma

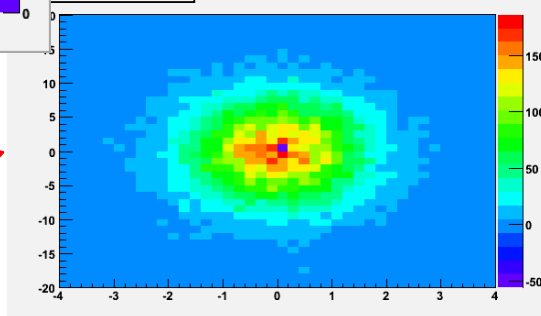
Distribution drawn with error bars (option E1)



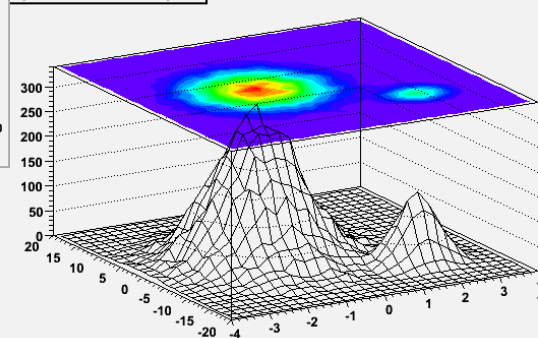
Option LEGO2Z example



on COLOr example



Option SURF3 example

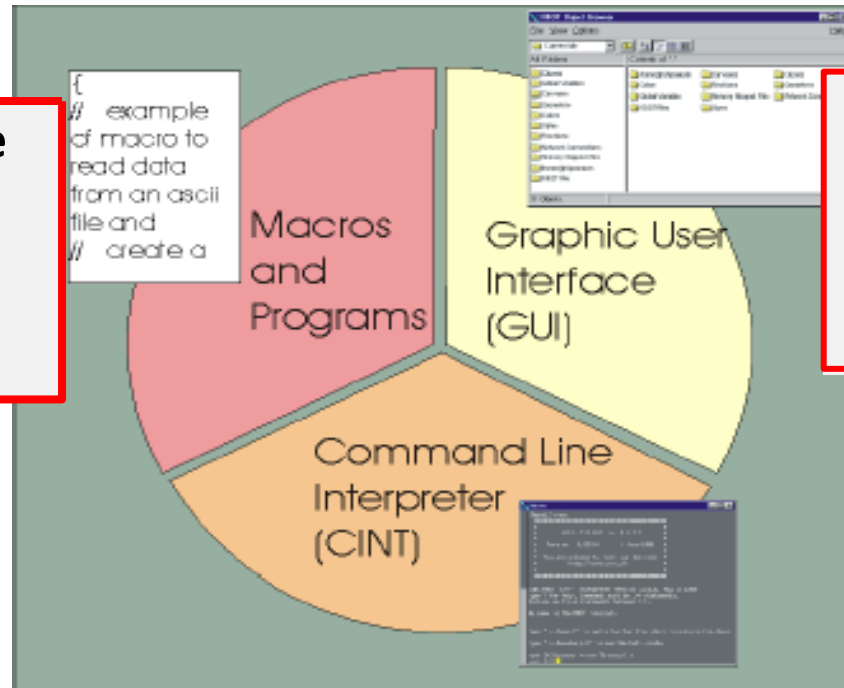


"LEGO"
"CONT"
"SURF"

ROOT User Interfaces

Le User Interfaces: come utilizzare ROOT

Uso di **macro**, codice interpretato o compilato (C++ compiler)



GUI: finestre grafiche, con diversi «menu». Più user-friendly, ideale per “cosmetica” interattiva

Command Line, con comandi diretti (C++ interpreter). Utile per operazioni semplici

ROOT - Command Line

La **command-line** User Interface:

```
Terminal
File Edit View Terminal Tabs Help
ROOT 5.25/04 (tags/v5-25-04@31410, Dec 01 2009, 12:38:41 on linux)

CINT/ROOT C/C++ Interpreter version 5.17.00, Dec 21, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] .q
bash-3.00$ root
*****
*                                     *
*      W E L C O M E  t o  R O O T      *
*                                     *
*   Version   5.25/04  23 November 2009   *
*                                     *
* You are welcome to visit our Web site *
*      http://root.cern.ch                *
*                                     *
*****

ROOT 5.25/04 (tags/v5-25-04@31410, Dec 01 2009, 12:38:41 on linux)

CINT/ROOT C/C++ Interpreter version 5.17.00, Dec 21, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] 
```

Da finestra di
shell inviate il
comando: **root**

“prompt” sessione interattiva di root

ROOT-Command Line

Due tipi di comandi:

- Comandi di base dell'interprete
- Comandi con sintassi C++

I comandi di base iniziano tutti con il punto, “.”

Esempio: `root[0] .q` vi fa uscire da root

Comandi di **SHELL** iniziano tutti con “.!”

Esempio:

`root[0] .! Ls` fa un listing della directory corrente

`root[0] .! pwd` stampa a schermo il path della directory corrente

ROOT-Command Line

Altri comandi di base dell'interprete:

- **.q** (.qqq, .qqqqqqq) per uscire da root
- **.files** fa vedere le librerie/sorgenti caricate
- **.L** carica in memoria i simboli definiti in una macro
- **.x** carica ed esegue una macro
- **.?** Lista/help sui comandi disponibili

ROOT-Command Line

- Comandi con sintassi di tipo C++

Esempio:

```
root[0] TH1F pippo=TH1F("pippo","prova",100,0.,1.)
```

crea un istogramma

root[1] int a=3; **definisce e assegna ad a il valore 3**

root[2] a **senza il ; stampa il tipo e il valore di a**

(int) 3 (il “;” in command line non è obbligatorio!)

ROOT-Command Line

Il prompt di ROOT è così anche una pratica calcolatrice tascabile:

root [0] 1+1

(int)2

root [1] 2*(4+2)/12.

(double) 1.000000e+00

root [2] sqrt(3.) (ROOT include anche la standard library del C++)

(double) 1.732051e+00

root [3] 1 > 2


(bool) false

root [4] TMath::Pi()

(double_t) 3.141593e+00

(TMath è una collezione di funzioni matematiche e costanti numeriche in ROOT)

ROOT-Command Line

- Per comandi C++ in linea di comando, sintassi standard con alcune “libertà” (che non si possono usare nelle macro...):
 - Il `;` può essere omissso su comando singolo
 - `TBrowser *b= new TBrowser() //` è ok!
 - Il tipo nelle dichiarazioni può essere omissso
 - `f= new TFile(“example.root”);`
 - Accesso agli oggetti attraverso il nome, non solo attraverso il loro puntatore:
 - `TH1F *histo=new TH1F(“histname”, “ Titolo”, 100, 0, 10)`
 - `histname->Draw()`  `equivalente a histo->Draw()`

ROOT-Command Line

- **Command-Line Help:**

Molto utile l'utilizzo del tasto <Tab> della tastiera:

Per esempio per creare un istogramma:

```
root[0] TH1F * h=new TH<Tab>
```

vi dà una prima lista di
match compatibili di classi
il cui nome inizia per TH

```
root[0] TH1F * h=new TH1<Tab>
```

raffinate la ricerca
(istogramma 1-D)

```
root[0] TH1F * h=new TH1F(<Tab>
```

vi dà una lista dei possibili
costruttori della classe TH1F

```
root[0] TH1F::<Tab>
```

vi dà una lista dei metodi
della classe TH1F

ROOT-Command Line

- **Per richiamare i comandi**, usare le frecce sulla tastiera, non è necessario riscriverli!
- La “storia” della sessione di ROOT è salvata nel file **.root_hist** nella vostra home Directory. Verificatelo con il comando **ls -a \$HOME/.root_hist**. Utile, perchè tiene traccia dei comandi che avete utilizzato e che possono essere eventualmente cut&pasted in una **macro**.
- Le **macro** sono la **seconda modalità** con cui l’utente può utilizzare ROOT, e conviene usare questo approccio e non la linea di comando non appena le operazioni che dobbiamo fare diventano un po’ più complesse.

ROOT Interfaces-MACRO

User Interface attraverso script (macro) :

- **Unnamed script**

- Inserire tutto il codice che si vuole eseguire tra { }
- Non possono essere dichiarate classi o funzioni
- Non si possono usare parametri

- **Named script**

- Come una qualunque funzione C++, stesse regole
- Si possono definire altre funzioni e classi, usare parametri
- La funzione **che ha lo stesso nome del file** viene eseguita con il comando `“ .x mymacro.C ”`. Per eseguirne una qualsiasi, prima “linkare” e poi chiamare il nome della funzione da riga di comando (`.L mymacro.C` e poi `funzione()`)

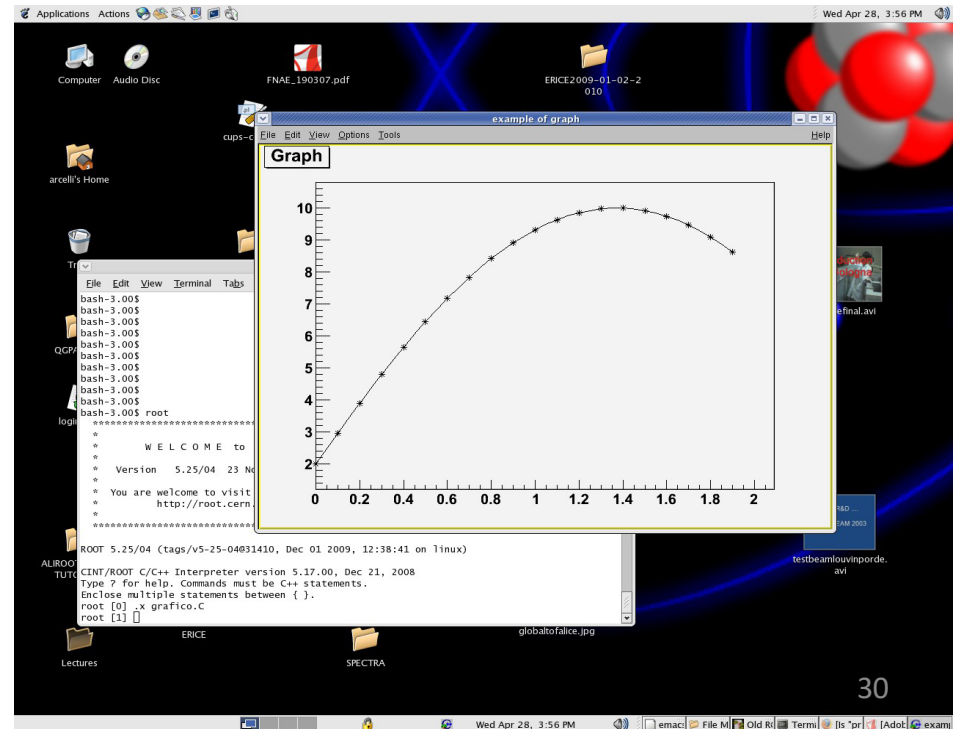
ROOT-MACRO

Esempio di **unnamed script** che fa un grafico:

Nel file **unnamed.C**:

```
{  
TCanvas *c1=new TCanvas("c1","Esempio di grafico",200,10,700,500);  
const Int_t n=20;  
Float_t x[n],y[n];  
for(Int_t i=0;i<n;i++){  
x[i]=i*0.1;  
y[i]=10*sin(x[i]+0.2);  
}  
TGraph *graph=new TGraph(n,x,y);  
graph->Draw("AC*");  
}
```

root[] **.x unnamed.C**

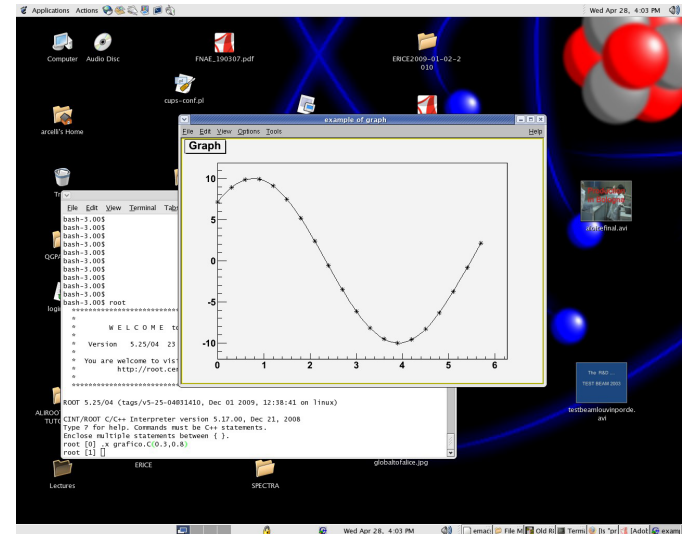


ROOT-MACRO

Esempio di **named script** che fa un grafico:

Nel file named.C:

```
void named(Float_t scale,Float_t offset){  
  TCanvas *c1=new TCanvas("c1","Esempio di grafico",200,10,700,500);  
  const Int_t n=20;  
  Float_t x[n],y[n];  
  for(Int_t i=0;i<n;i++){  
    x[i]=i*scale;  
    y[i]=10*sin(x[i]+offset);  
  }  
  TGraph*graph=new TGraph(n,x,y);  
  graph->Draw("AC*");  
}
```



root[] **.x named.C** ("linka" ed esegue **solo** la funzione
con lo stesso nome della macro)

oppure:

root[] **.L named.C** ("linka")

root[] **named(0.3,0.8)** (esegue)

ROOT Interfaces-GUI

Menus, toolbars

Contenuto **root** file:

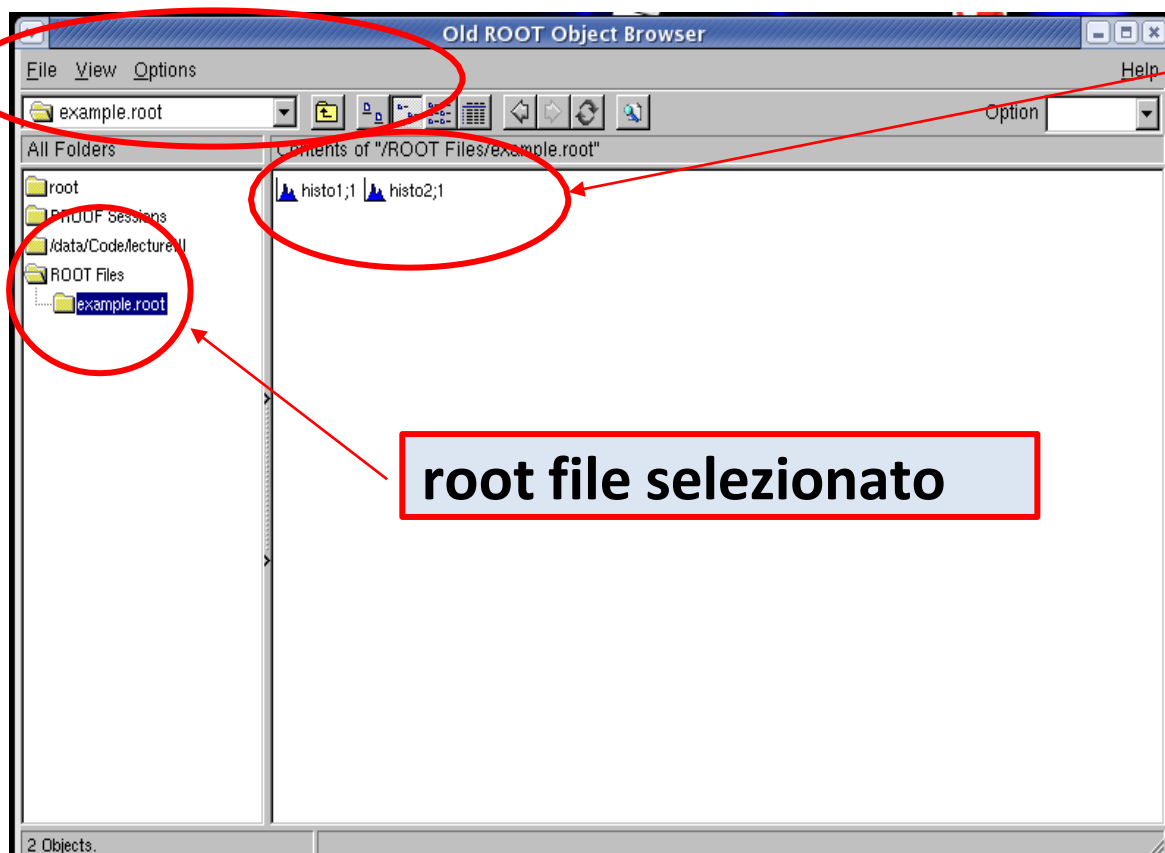
Oggetti root:
(istogrammi, grafici,...)

root file selezionato

TBrowser:

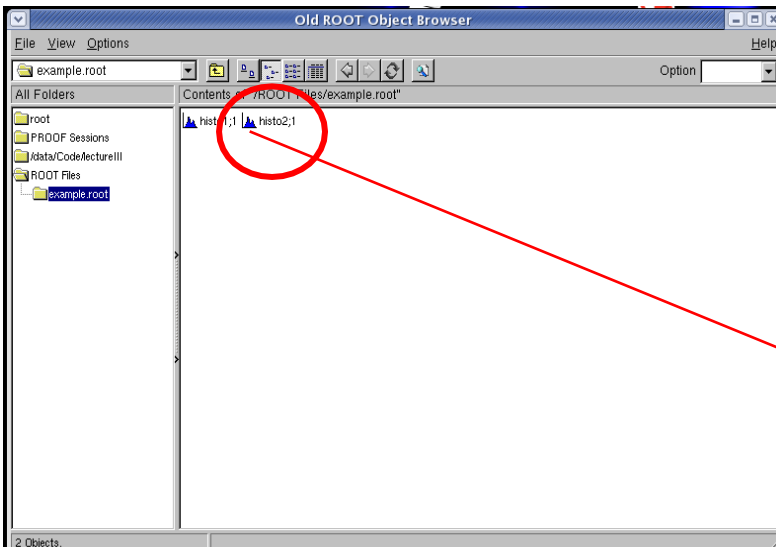
Finestra grafica per
accesso a **file di root**

da root[] dare il
comando **TBrowser b;**



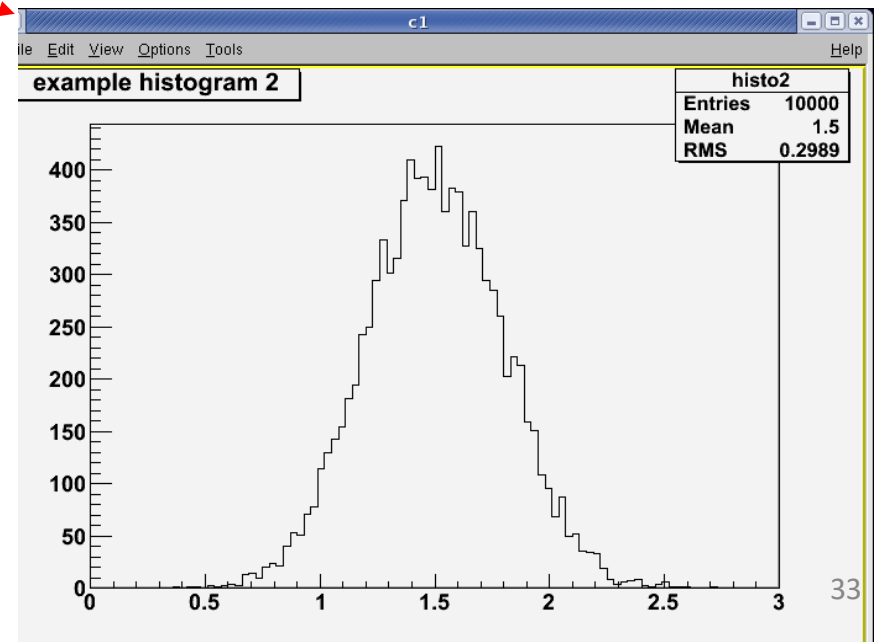
ROOT-GUI

TBrowser



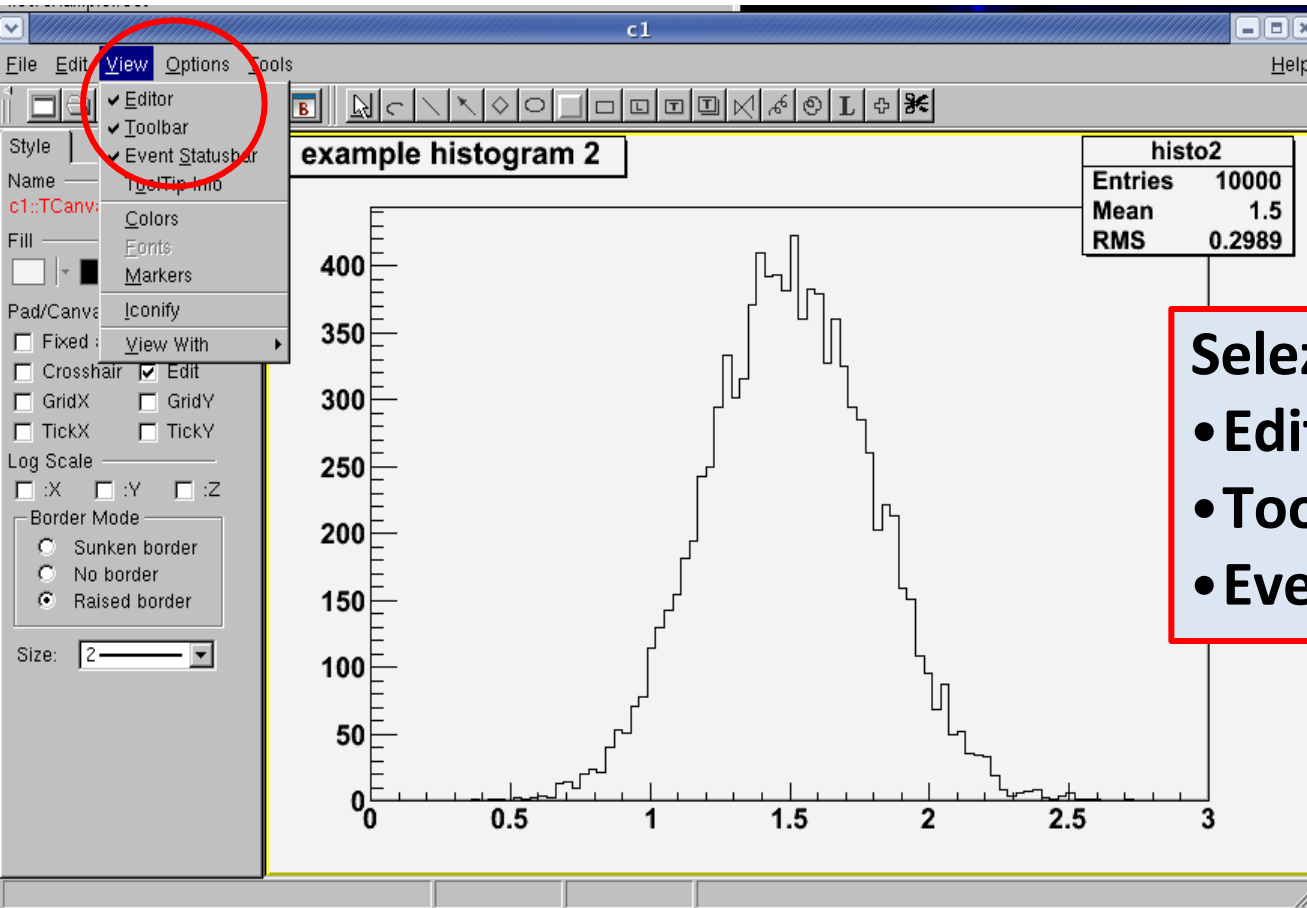
Clik 2 volte con il bottone sinistro del mouse su un oggetto nel file: si apre un **TCanvas**, la finestra di interfaccia per la grafica, che visualizza l'oggetto

TCanvas



ROOT-GUI

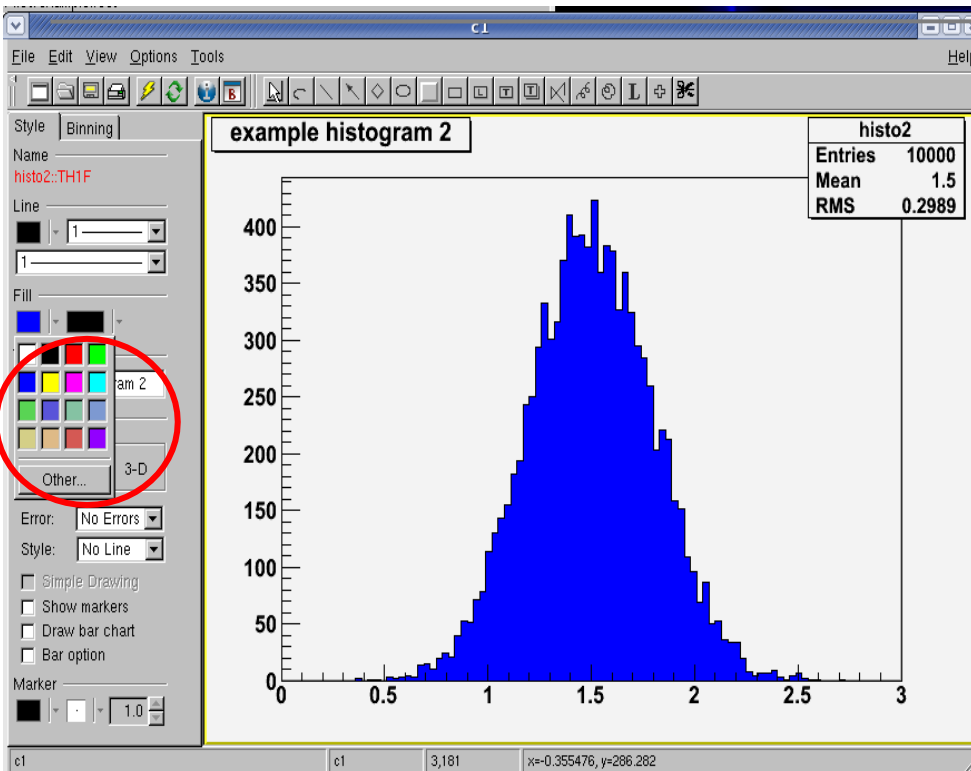
Come agire facilmente sulla canvas:



Selezionare in **View**:

- Editor
- ToolBar
- Event StatusBar

ROOT-GUI



Editor Menu

- Con il **botton** **sx** del mouse potete **selezionare** i vari oggetti nella canvas (1 click):
Pad, Frame, assi, istogramma, ..
- Andate sull'editor a sinistra per **cambiare** i parametri di visualizzazione dell'oggetto. L'editor cambia a seconda dell'oggetto selezionato
- Con il bottone sinistro potete anche postare/ridimensionare gli oggetti

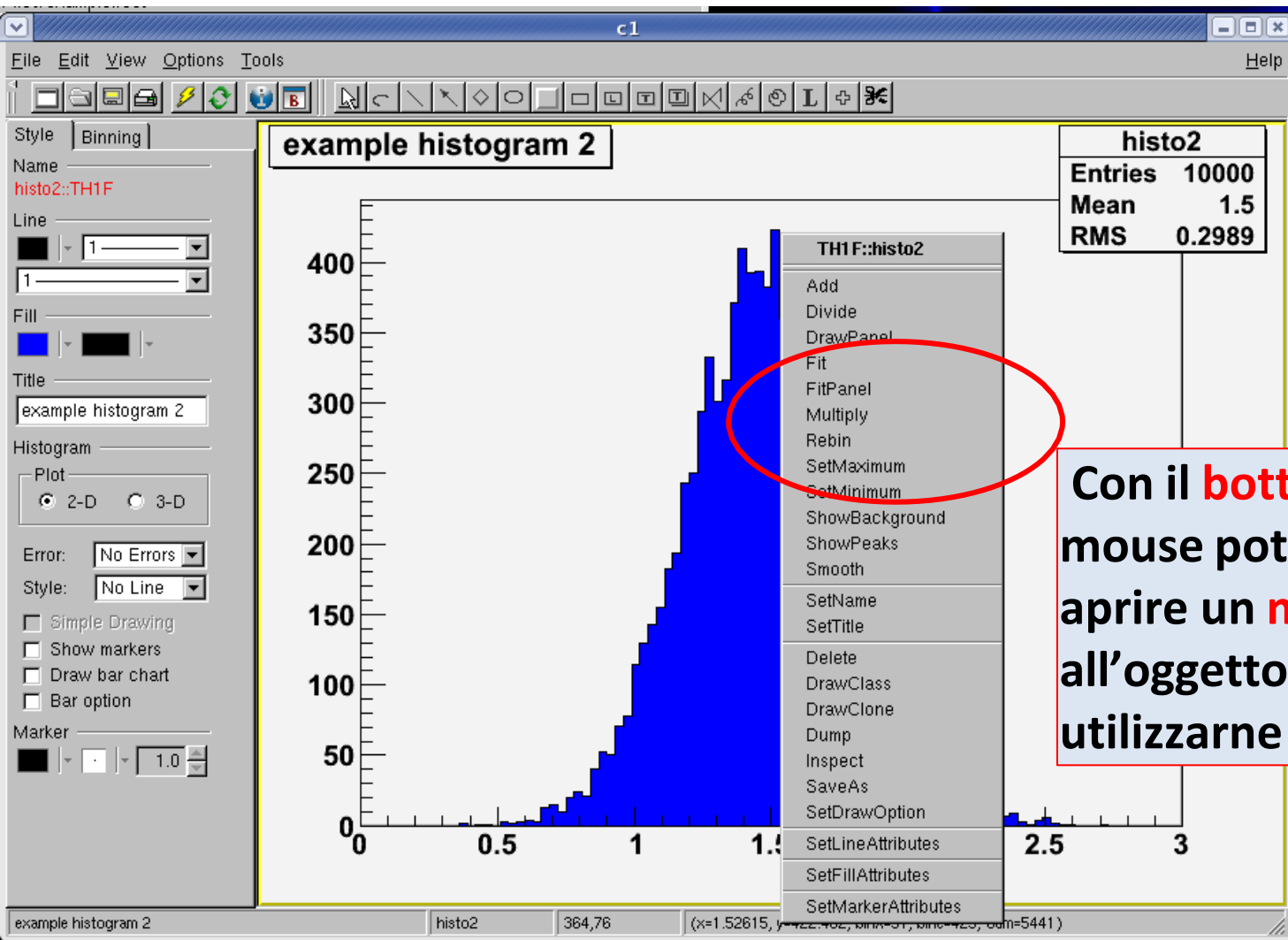
ROOT-GUI

Toolbar

- La ToolBar in alto vi consente di inserire testo, simboli, frecce, forme di base
- La Status Bar vi dice su quale oggetto è posizionato il mouse e la posizione del mouse sulla Canvas

Status Bar

ROOT-GUI

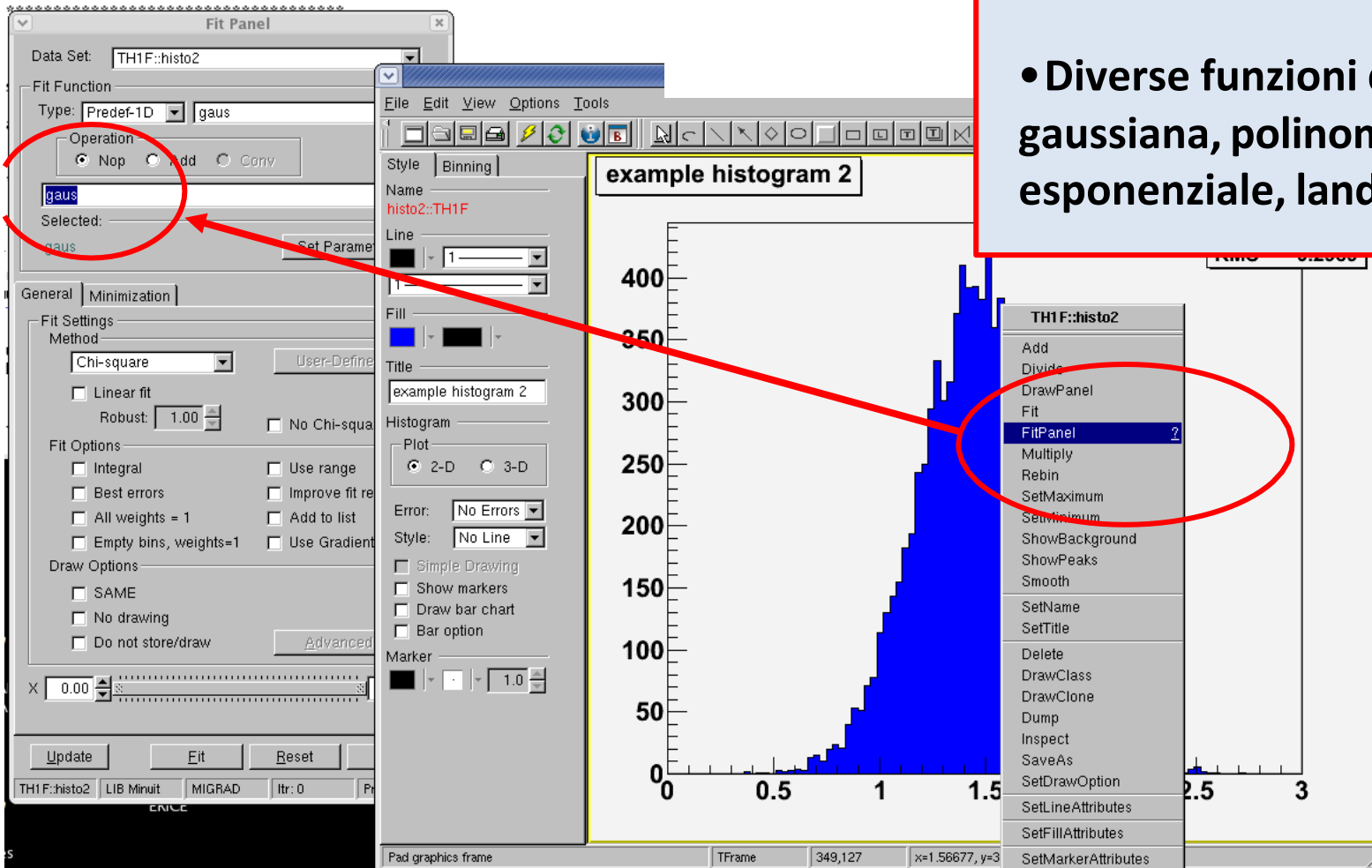


Con il **bottono dx** del mouse potete selezionare e aprire un **menu** contestuale all'oggetto selezionato, per utilizzarne alcuni metodi.

ROOT-GUI

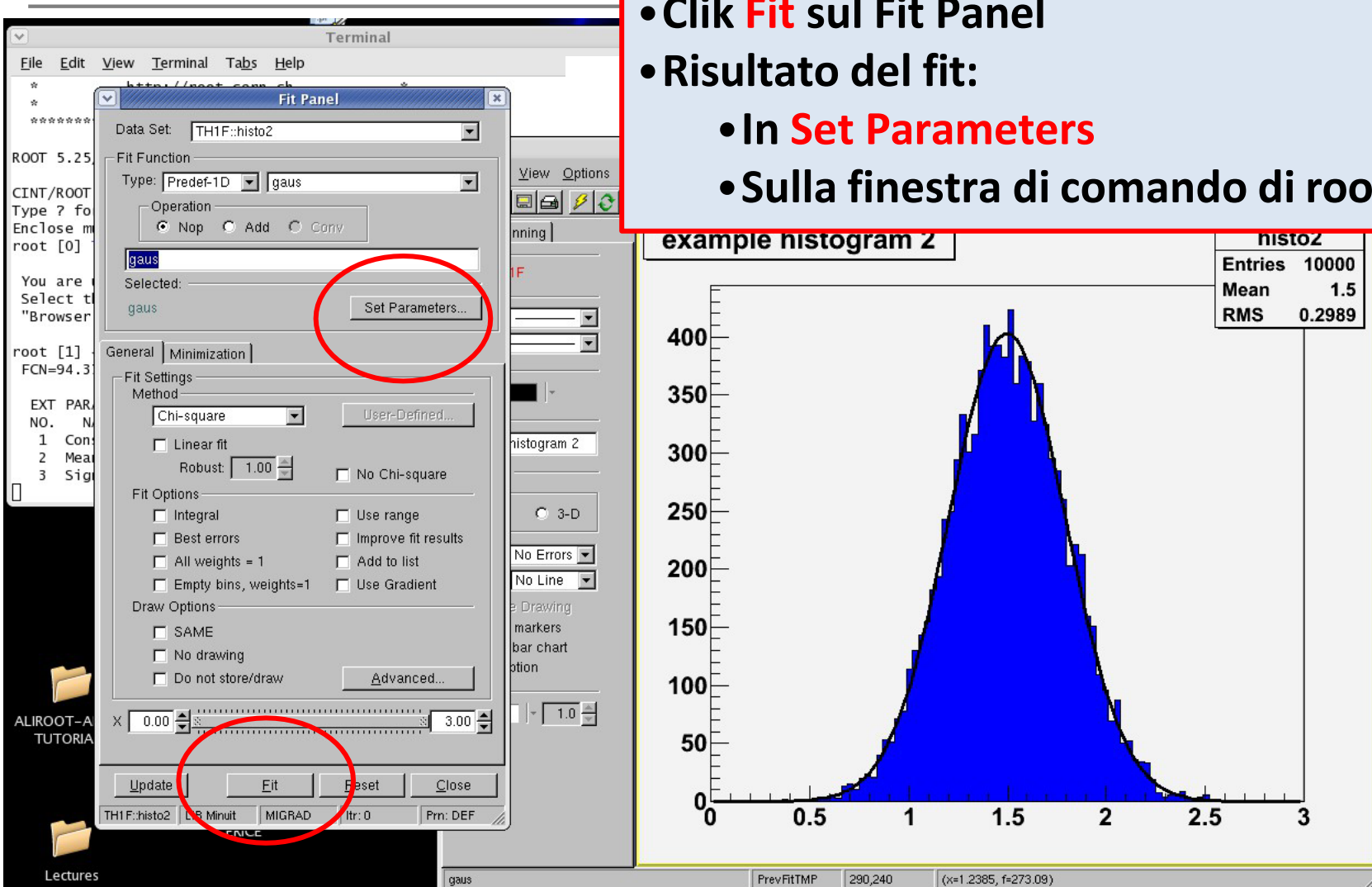
- Per esempio, se vogliamo fare un fit dell'istogramma, selezioniamo **"Fit"** o **"FitPanel"**:

- Diverse funzioni di default: gaussiana, polinomiale, esponenziale, landau



ROOT-GUI

- Klik **Fit** sul Fit Panel
- Risultato del fit:
 - In **Set Parameters**
 - Sulla finestra di comando di root



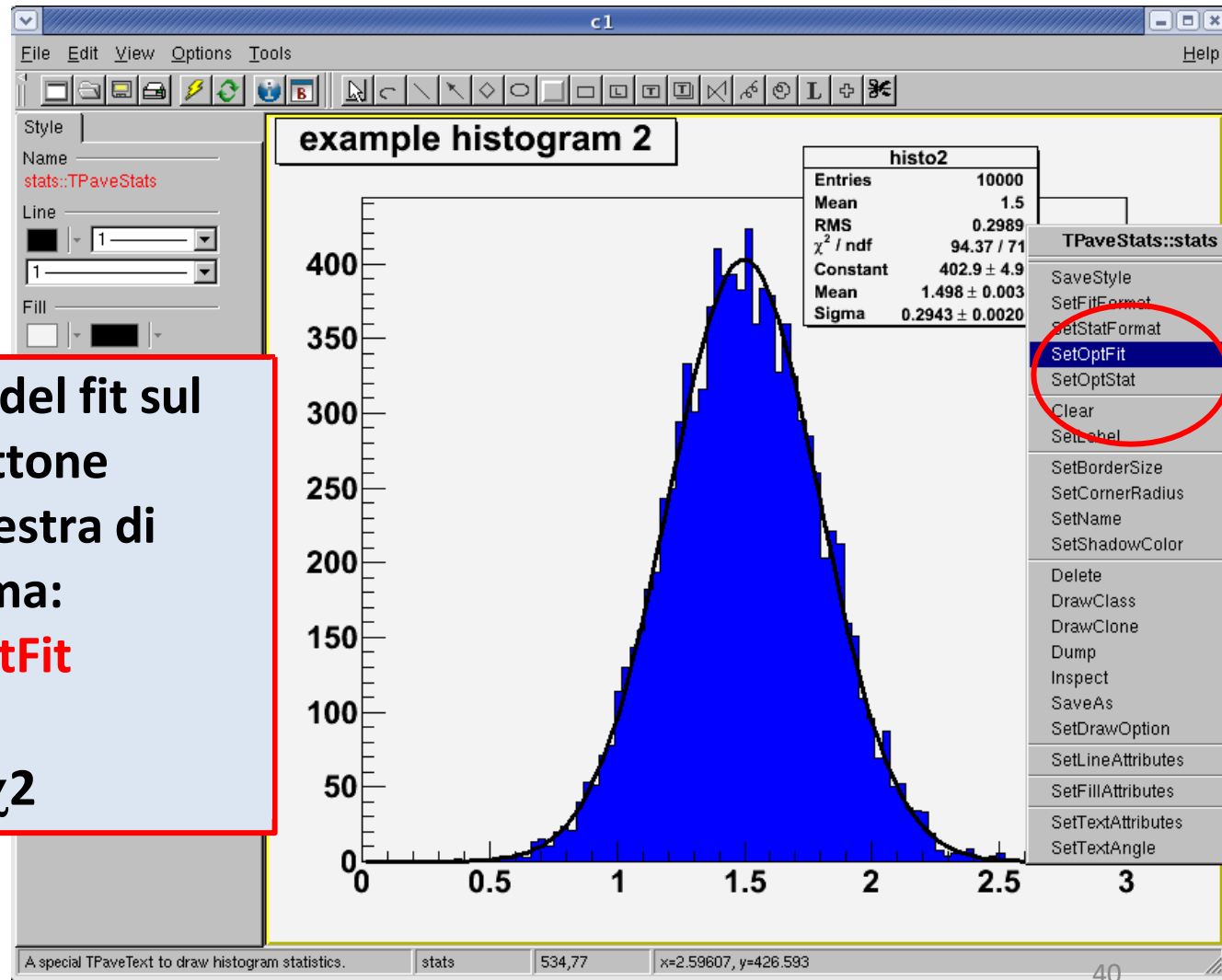
ROOT-GUI

Per visualizzare l'esito del fit sul grafico, aprire con il bottone destro il menu della finestra di statistica dell'istogramma:

- Selezionare **SetOptFit**

- Scrivere **111**

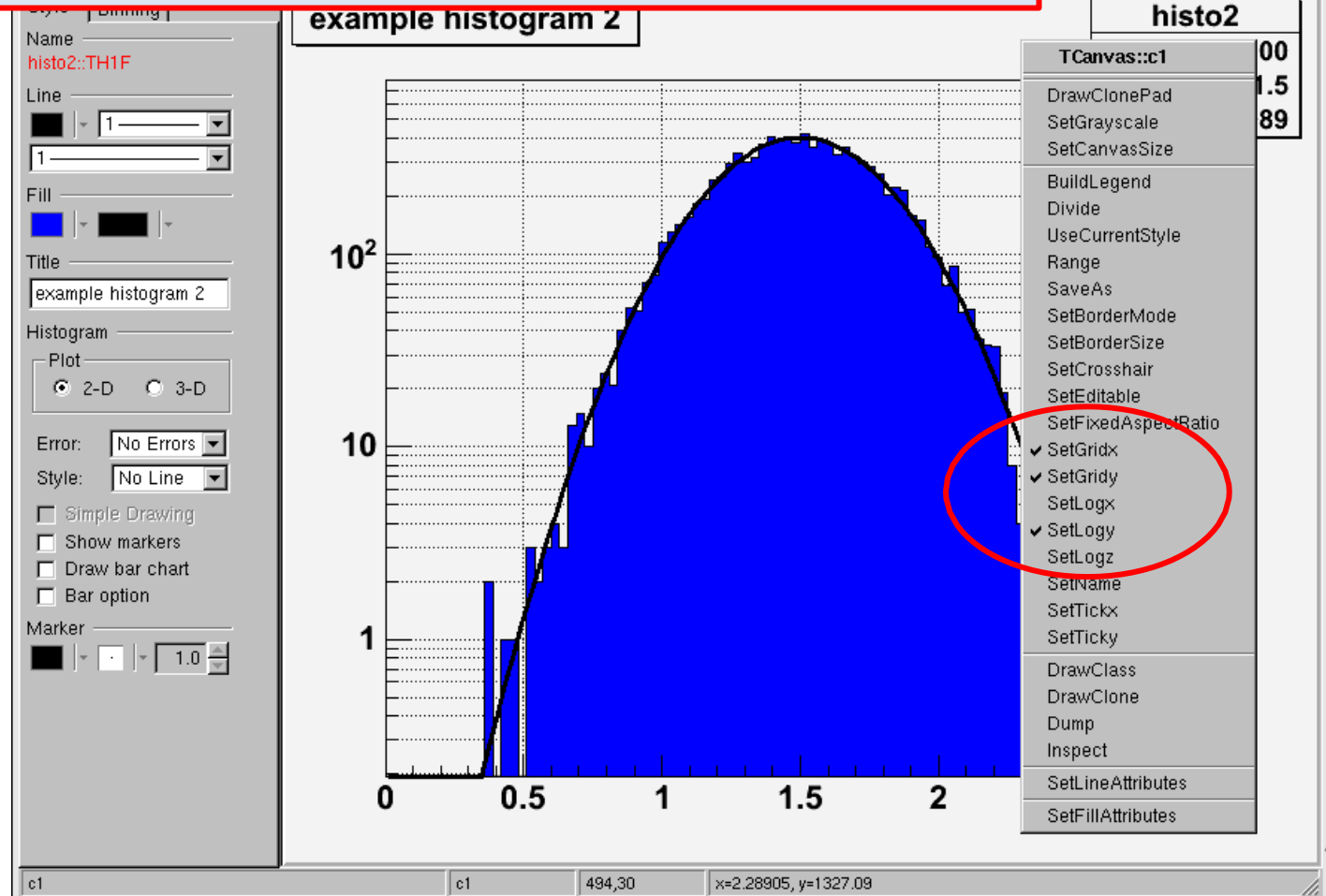
Parametri con errori e χ^2



ROOT-GUI

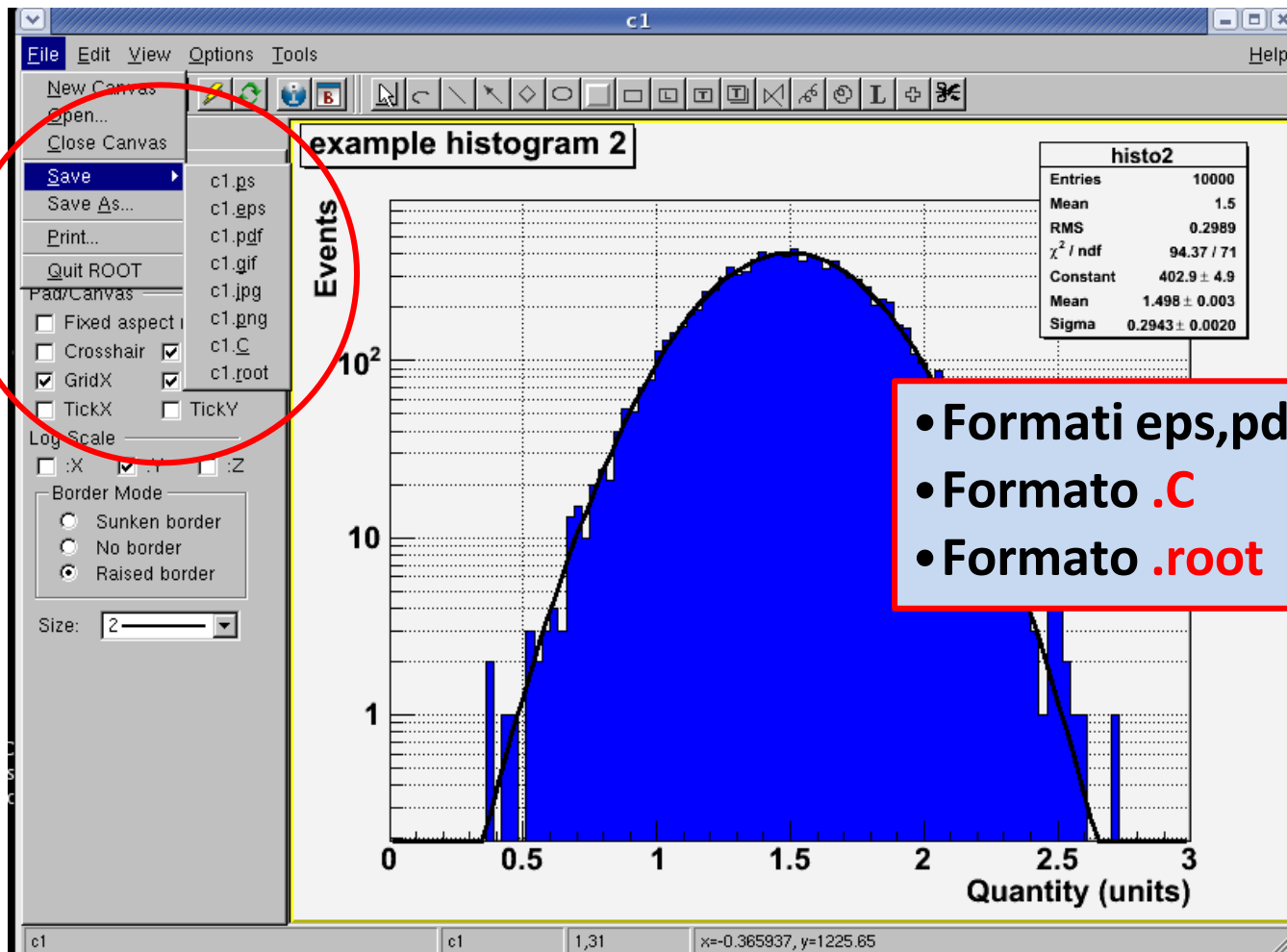
• Altro esempio: uso del bottone dx sulla canvas:

- Scala Logaritmica in y -> **SetLogy**
- Griglia x-y -> **SetGridx, SetGridy**



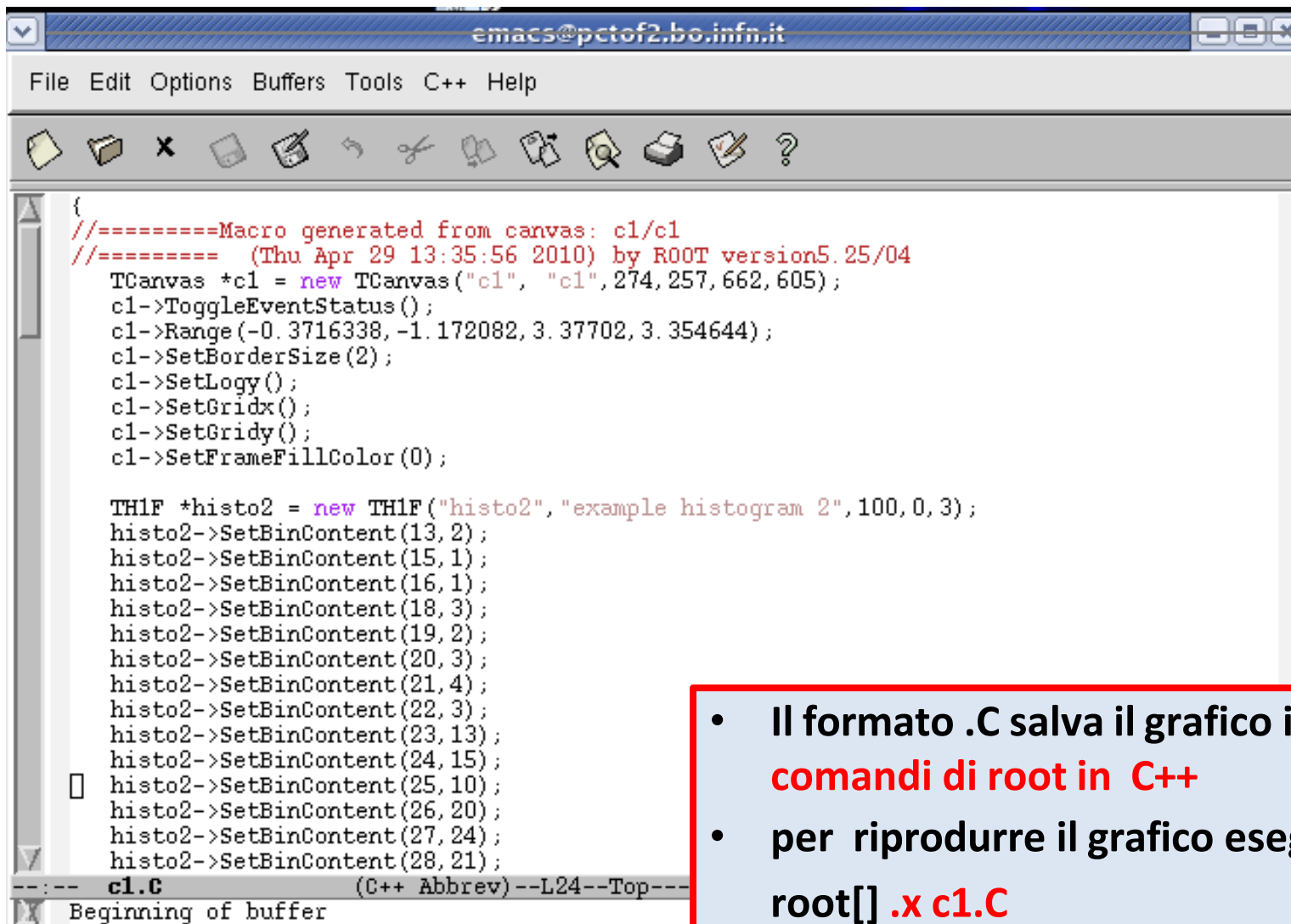
ROOT-GUI

Per salvare il vostro lavoro, diverse possibilità:
Nel menu **File** della canvas, selezionare **Save**



- Formati eps,pdf,png,jpg,gif
- Formato **.C**
- Formato **.root**

ROOT-GUI



The screenshot shows a window titled 'emacs@pctof2.bo.infn.it' with a menu bar (File, Edit, Options, Buffers, Tools, C++, Help) and a toolbar. The main text area contains a C++ macro. The macro starts with a comment indicating it was generated from a canvas 'c1/c1' on 'Thu Apr 29 13:35:56 2010' by 'ROOT version 5.25/04'. It defines a TCanvas object 'c1' with specific dimensions and settings, and then creates a TH1F histogram 'histo2' with 100 bins. The histogram's bin contents are set in a loop from bin 13 to 28. The status bar at the bottom shows 'c1.C (C++ Abbrev) --L24--Top--' and 'Beginning of buffer'.

```
{
//=====Macro generated from canvas: c1/c1
//===== (Thu Apr 29 13:35:56 2010) by ROOT version 5.25/04
TCanvas *c1 = new TCanvas("c1", "c1", 274, 257, 662, 605);
c1->ToggleEventStatus();
c1->Range(-0.3716338, -1.172082, 3.37702, 3.354644);
c1->SetBorderSize(2);
c1->SetLogy();
c1->SetGridx();
c1->SetGridy();
c1->SetFrameFillColor(0);

TH1F *histo2 = new TH1F("histo2", "example histogram 2", 100, 0, 3);
histo2->SetBinContent(13, 2);
histo2->SetBinContent(15, 1);
histo2->SetBinContent(16, 1);
histo2->SetBinContent(18, 3);
histo2->SetBinContent(19, 2);
histo2->SetBinContent(20, 3);
histo2->SetBinContent(21, 4);
histo2->SetBinContent(22, 3);
histo2->SetBinContent(23, 13);
histo2->SetBinContent(24, 15);
histo2->SetBinContent(25, 10);
histo2->SetBinContent(26, 20);
histo2->SetBinContent(27, 24);
histo2->SetBinContent(28, 21);
}
```

--- c1.C (C++ Abbrev) ---L24---Top---

Beginning of buffer

- Il formato .C salva il grafico in forma di **comandi di root in C++**
- per riprodurre il grafico eseguite la macro:
root[] .x c1.C

ROOT-GUI

The screenshot displays the ROOT GUI interface. On the left, the 'Old ROOT Object Browser' window shows a tree structure with 'c1.root' selected. A red circle highlights the toolbar of the Object Browser. In the center, a histogram plot titled 'example' is shown, with a blue-filled area under a black curve. The y-axis is labeled 'Events' on a logarithmic scale from 1 to 100. The x-axis is labeled 'Quantity (units)' from 0 to 3. A statistics box in the top right corner of the plot shows: Mean 1.498 ± 0.003 and Sigma 0.2943 ± 0.0020 . Below the plot, a status bar shows 'Pad graphics frame', 'TFrame', and coordinates 'x=0.918251, y=16.2106'. At the bottom left, a desktop-like area shows folders for '872_08 MODUL compiti did.-scie', '08_08 doc', 'laura', 'ERICE', and 'Lectures'.

Il formato .root salva la canvas e tutti gli oggetti di root in essa contenuti

- per ricreare il grafico aprite il root file c1.root e fate doppio click sulla canvas
- Tutti gli oggetti sono nuovamente disponibili per esser ulteriormente manipolati