METODI STATISTICI E COMPUTAZIONALI

Stefania Spagnolo

Dipartimento di Matematica e Fisica, Univ. del Salento

INTRODUZIONE

PRESENTAZIONE

STEFANIA ANTONIA SPAGNOLO

Professore II Fascia (Associato)

Settore Scientifico Disciplinare FIS/01: FISICA SPERIMENTALE.

stefania.spagnolo@unisalento.it
https://www.unisalento.it/people/stefania.spagnolo
Vai alla mappa

Dipartimento di Matematica e Fisica "Ennio De Giorgi"
Ex Collegio Fiorini - Via per Arnesano - LECCE (LE)
Ufficio, Piano terra
Telefono **+39 0832 29 7439**

Fisica Sperimentale delle Interazioni Fondamentali (02/A1) Settore Scientifico Disciplinare FIS04 (Fisica Nucleare e Sub.Nucleare)

Area di competenza:
Esperimenti: **ATLAS** a LHC e OPAL a LEP (CERN), PADME a BTF e KLOE a DAPHNE (LNF).
Ricerca di nuova fisica esotica in pp a 13-14 TeV, produzione associata di b-jets e W/Z in pp a 7 TeV; Produzione di coppie di fermioni e accoppiamenti di gauge anomali in e+e- a 200 GeV; sezione d'urto adronica in e+e- a E<1GeV per contributo adronico a g-2 del muone.
Rivelatori per la fisica delle alte energie: Pixel (ATLAS Upgrade per HL-LHC) RPC e Spettrometro a Muoni (ATLAS)
Rivelatori a diamante (PADME e **DIAPIX** [R&D]), rivelatori di tracciamento a drift in miscele leggere (**KLOE**)

Orario di ricevimento
Giovedi 15-17; Lunedì 11-13

Recapiti aggiuntivi
Ufficio 225, primo piano tel +39 0832 297439

Visualizza QR Code
Scarica la Visit Card

Orari indicativi, venite o fissate appuntamento

Oggetto: Corso MSC
per favore

INFORMAZIONI

Didattica

Torna all'elenco

Scheda compilata correttamente **METODI STATISTICI E COMPUTAZIONALI** Modifica

Scarica scheda insegnamento

Corso di laurea	**FISICA**
Settore Scientifico Disciplinare	**FIS/01**
Tipo corso di studio	**Laurea**
Crediti	**6.0**
Ripartizione oraria	**Ore totali di attività frontale: 52.0**
Per immatricolati nel	**2022/2023**
Anno accademico di erogazione	**2023/2024**
Anno di corso	**2**
Semestre	**Primo Semestre (dal 18/09/2023 al 15/12/2023)**
Lingua	**ITALIANO**
Percorso	**PERCORSO COMUNE (999)**
Sede	**Lecce**

Prerequisiti

Breve descrizione del corso

Obiettivi formativi

Metodi didattici

Modalità d'esame

Altre informazioni utili

Link a diario delle lezioni con raccolta di slide ed esercizi

Programma esteso

Testi di riferimento

Pagina del corso sul sito di Unisalento:

- Scheda del corso

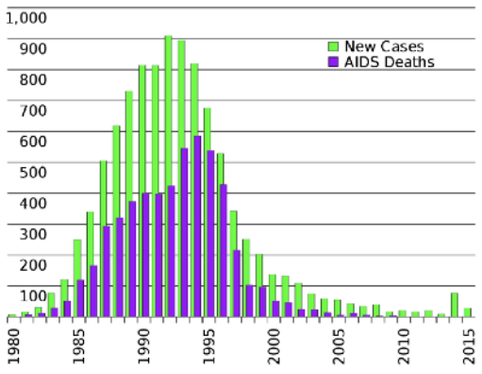
- **Programma delle lezioni**
  - Pagina di lavoro, costruita durante il corso
  - Più utile

Lezioni

aa2023-2024

Metodi statistici e computazionali

CdL Fisica



Link e informazioni generali

Scheda insegnamento; Comunicazioni

Orario sul sito di Unisalento (tutte le lezioni del secondo anno CdL Fisica)

In generale, aula Anni (F8) (Ecotekne, Fiorini):

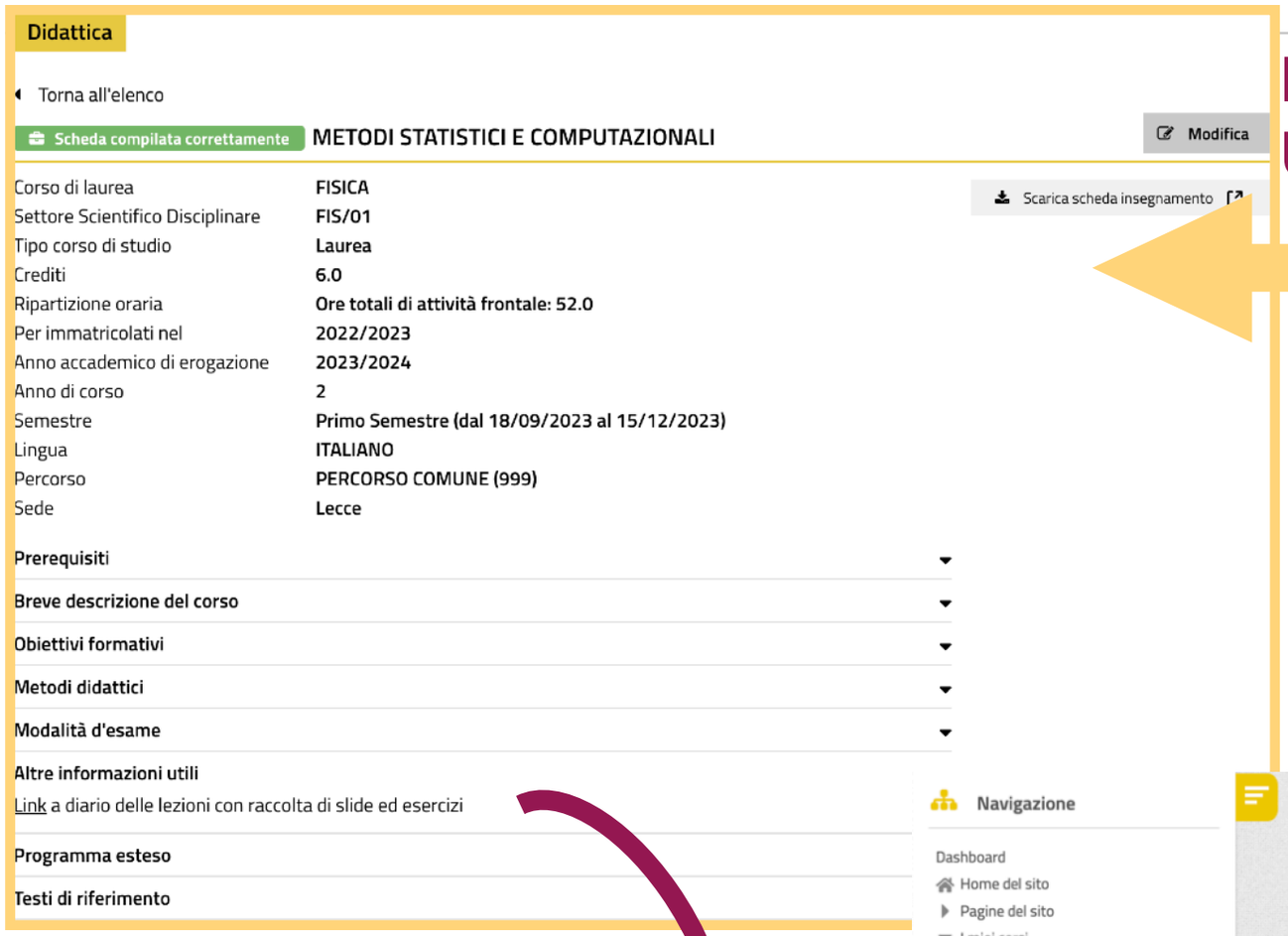
1. Martedì 9:00 - 11:00
2. Mercoledì 11:00 - 13:00
3. Venerdì 11:00 - 13:00 (non tutte le settimane)

Inizio lezioni **19 Settembre**

Root @ CERN

1. Tutorials
2. Manuale
3. Slides di introduzione a ROOT: ufficiali (indirizzate a utenti CERN), ma molte altre risorse sono reperibili con una ricerca su google

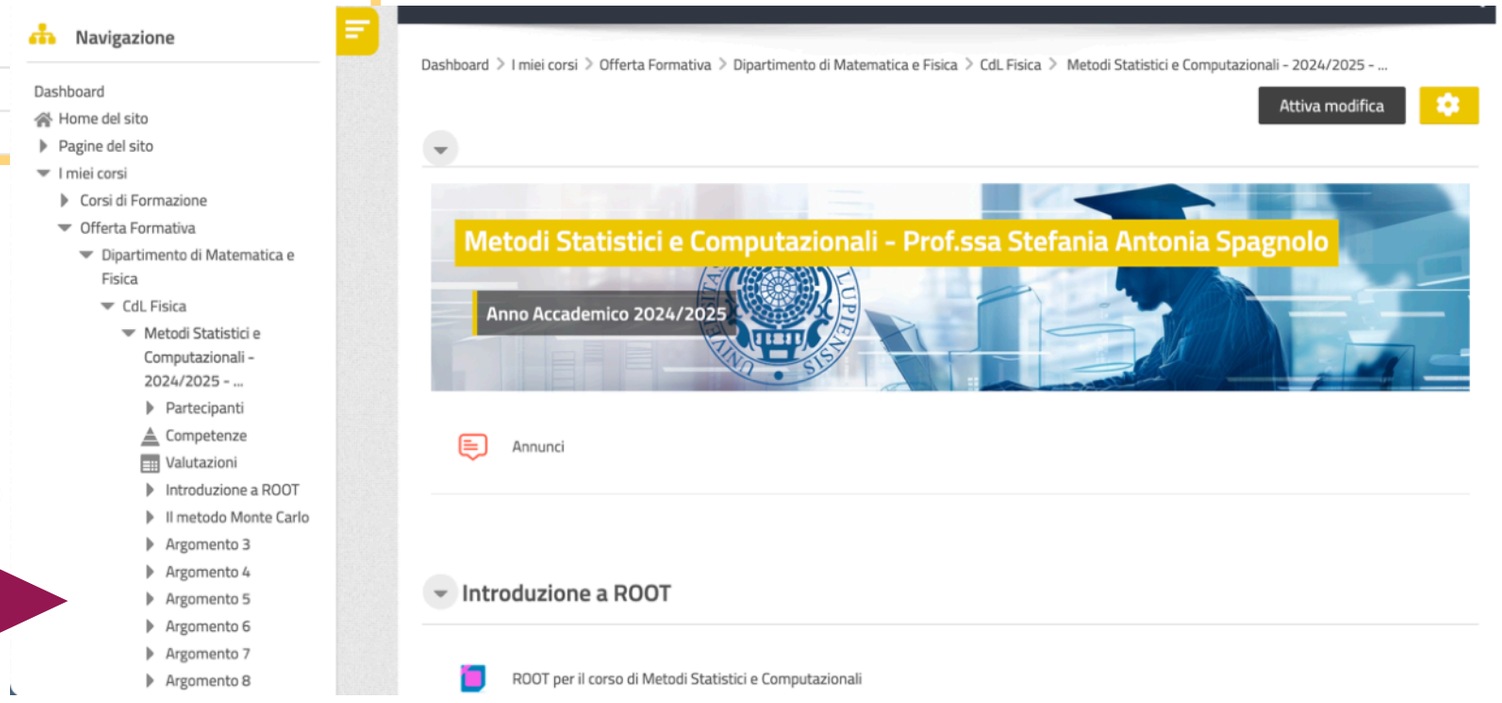
INFORMAZIONI



Pagina del corso sul sito di Unisalento:

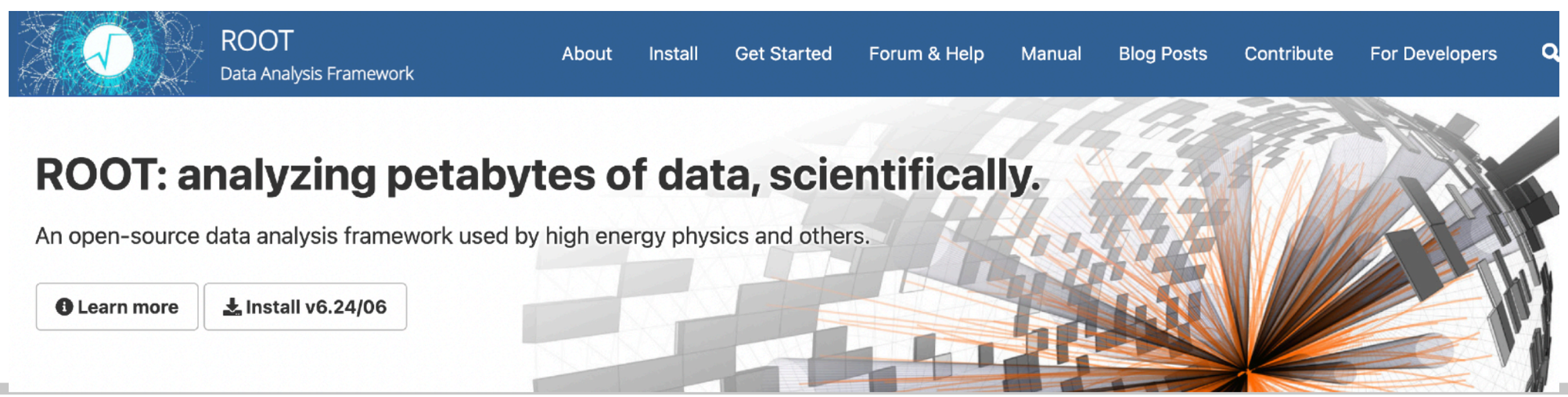
- Scheda del corso

- **Materiale didattico**
  - Piattaforma e-learning Unisalento



Organizzazione del corso

- Oltre alle **lezioni frontali** il corso prevede **esercitazioni organizzate usando i propri laptop o PC**.
  - Si consiglia di frequentare le lezioni muniti di laptop per affrontare insieme le applicazioni pratiche
- Non viene proposto un testo di riferimento ma gli argomenti trattati sono via via estratti da più testi e articoli.
  - Il materiale di ogni lezione sarà raccolto sulle pagine web del corso.
- Sarà utilizzato come informatico di supporto il software **ROOT:** https://root.cern
- ROOT è un insieme di librerie/ambiente di lavoro sviluppato dal CERN per analisi dati che comprende strumenti grafici e sofisticati strumenti statistici e numerici.
- ROOT è sviluppato per Linux, ma esistono versioni per sistemi operativi Windows e Mac
- *L'esame finale consisterà in un seminario di 15 minuti su un argomento e un esercizio assegnati con quindici giorni di anticipo per coloro che hanno partecipato alle esercitazioni. In una prova orale (+ un esercizio) per coloro che non hanno potuto partecipare.*



https://root.cern/

ROOT

Nel corso

- produrremo e analizzeremo insiemi di dati rappresentati da distribuzioni statistiche (==> istogrammi) o grafici a punti (==> grafici) [metodi statistici]

- applicheremo tecniche numeriche per trovare soluzioni di problemi matematici (eq. differenziali, per esempio) [metodi computazionali; rappresentiamo la soluzione come un grafico
 (==> grafici di funzione)

https://root.cern/

ROOT

ROOT Basic Course

MINICORSO SU ROOT

ROOT in a Nutshell

- ROOT is a software framework with building blocks for:
  - Data processing
  - Data analysis
  - Data visualisation
  - Data storage

An Open Source Project

We are on github

github.com/root-project

All contributions are warmly welcome!

- ROOT is written mainly in C++ (C++11/17 standard)
  - Bindings for Python available as well

- Adopted in High Energy Physics and other sciences (but also industry)
  - 1 EB of data in ROOT format
  - Fits and parameters' estimations for discoveries (e.g. the Higgs)
  - Thousands of ROOT plots in scientific publications

10

(cc) BY

Dipartimento di Matematica e Fisica
"Ennio De Giorgi"
INFN

ROOT in a Nutshell

- ROOT can be seen as a collection of building blocks for various activities, like:
  - **Data analysis: histograms, graphs, functions**
  - **I/O: row-wise, column-wise** storage of any C++ object
  - **Statistical tools** (RooFit/RooStats): rich modeling and statistical inference
  - Math: **non trivial functions** (e.g. Erf, Bessel), optimised math functions
  - **C++ interpretation**: full language compliance
  - **Multivariate Analysis** (TMVA): e.g. Boosted decision trees, NN
  - **Advanced graphics** (2D, 3D, event display)
  - **Declarative Analysis**: RDataFrame
  - And more: HTTP servering, JavaScript visualisation

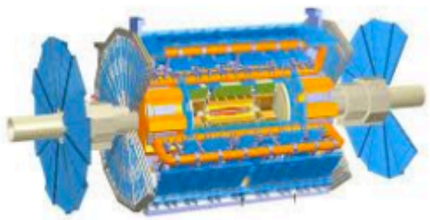
11

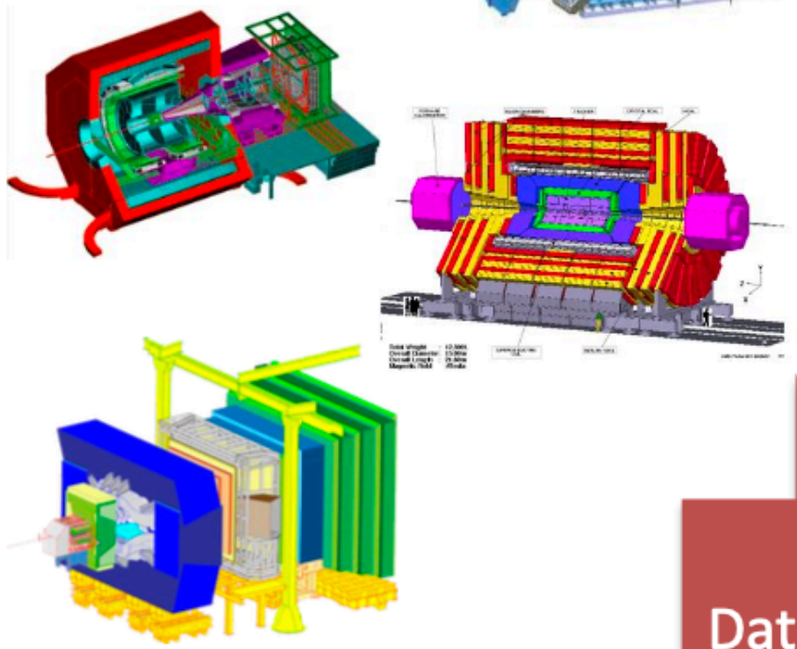
(cc) BY

MINICORSO SU ROOT

ROOT Application Domains

A selection of the experiments adopting ROOT





Event Filtering

Data

Offline Processing

Analysis

Reconstruction

Further processing, skimming

Event Selection, statistical treatment …

Raw

Reco

…

Analysis Formats

Images

Data Storage: Local, Network

12

(cc) BY

Interpreter

- ROOT has a built-in interpreter : CLING
  - C++ interpretation: highly non trivial and not foreseen by the language!
  - One of its kind: Just In Time (JIT) compilation
  - A C++ interactive shell

```
$ root
root[0] 3 * 3
(const int) 9
```

- Can interpret "macros" (non compiled programs)
  - Rapid prototyping possible
- ROOT provides also Python bindings
  - Can use Python interpreter directly after a simple *import ROOT*
  - Possible to "mix" the two languages (see more later)

Dipartimento di Matematica e Fisica
"Ennio De Giorgi"
INFN

Persistency or Input/Output (I/O)

- ROOT offers the possibility to write C++ objects into files
  - This is impossible with C++ alone
  - Used the LHC detectors to write several petabytes per year
- Achieved with serialization of the objects using the reflection capabilities, ultimately provided by the interpreter
  - Raw and column-wise streaming
- As simple as this for ROOT objects: one method - *TObject::Write*

Cornerstone for storage of experimental data

MINICORSO SU ROOT

- ROOT Website: https://root.cern
- Material online: https://github.com/root-project/training
- More material: https://root.cern/getting-started
  - Includes a booklet for beginners: the “ROOT Primer”
- Reference Guide: https://root.cern/doc/master/index.html
- Forum: https://root-forum.cern.ch

ROOT INSTALLAZIONE

- https://root.cern/install/

- Se dovete installarlo da scratch:
  - The latest stable ROOT release is 6.28/06 (a.a. 23-24) -> 6.32/04 (oggi)

- Download a pre-compiled binary distribution
  - pre-compiled ROOT for several major Linux distributions as well as MacOS and (as a beta) Windows. The steps to install a pre-compiled binary are simple:
    - 1) Install all required dependencies with the system package manager
    - 2) Download the release for the desired platform and ROOT version
    - 3) Unpack the archive
    - 4) Add the ROOT libraries and executables to your environment by sourcing the appropriate thisroot.* script. These setup scripts can be found in the ROOT binary release, in the bin directory

Attenzione all'ordine delle operazioni

ROOT INSTALLAZIONE

And on Windows, for example, after the installation, open a `x86 Native Tools Command Prompt for VS 2019`, cd to your home directory (`cd %USERPROFILE%`) and call `thisroot.bat` (let's assume you installed ROOT in `C:\root`). Then you can start ROOT:

```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.11.3
** Copyright (c) 2021 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86'

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>cd %USERPROFILE%

C:\Users\username>c:\root\bin\thisroot.bat

C:\Users\username>root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/06                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for win32 on Aug 28 2023, 11:34:39                         |
  | From tags/6-28-06@6-28-06                                        |
  | With MSVC 19.29.30133.0                                          |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0]
```

Una volta installato root è necessario definire delle variabili di ambiente per poter lanciare root da una directory qualunque del vostro PC

ROOT DA PROMPT DI COMANDI SU WINDOWS

- Qualcosa a cui non siete abituati: usare root da terminale (prompt dei comandi)
  - Scrivi/copia la tua macro nella directory (nome e path arbistrario: C:\Users\username\Documenti\MetodiSC)
  - Lanciare la finestra del prompt di comandi
  - Spostarsi nella directory in cui ci sono le macro che volete eseguire:
    - ***cd Documenti\MetodiSC***
      - NOTA: all'avvio del terminale / finestra dei prompt di comandi ci si trova nella propria home directory, ossia nella cartella chiamata C:\Users\username
      - RICORDA:
        - per spostarsi in una sotto-directory (di quella attuale) chiamata ***Univ***, bisogna eseguire ***cd Univ***
        - ***.. => e' il nome della directory superiore***
        - ***. => e' il nome della directory attuale***
        - Per sapere in che directory ci troviamo, basta seguire ***dir*** che elencherà tutti i file e sotto-directory della directory attuale il cui nome sara' scritto all'inizio
  - Eseguire lo script che definisce le variabili di ambiente:
    - ***C:\directory_installazione_root\bin\thisroot.bat***
      - NOTA: ***directory_installazione_root*** e' la directory in cui root e' stato istallato; normalmente e' \root\ ma potrebbe avere un atro nome; se non la conosci, osserva le proprietà del link logico a root sul tuo desktop: da li capirai qual e' la directory in cui si trova l'eseguibile di root e quindi la directory di installazione.
  - Lancia root dal terminale con il semplice comando:
    - ***root***
  - Esegui la tua macro (chiamata miaMacro.C) con
    - root[0] .x miaMacro.C
  - Oppure carica in memoria le funzioni definite nella tua macro (miaCollezioneDiMacro.C) e poi lancia una di esse (macro1):
    - root[0] .L miaCollezioneDiMacro.C
    - root[1] macro1()

RICHIAMI DI C++

A Simple Spec

C++ consists of

- expressions: have a value or result

```
cos(0.1)
```

- declarations: introduce a name

```
int i;
```

31

Declarations

Declarations introduce

- variables

```
int i; TMyClass obj;
```

- functions

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

- types

```
class TMyClass { int fMember; };
```

32

RICHIAMI DI C++

A typical class

```
class TMyClass {
  int fNum;
public:
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```


A typical class (2)

Looking at the parts:

```
class TMyClass {
  int fNum;
public:
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

Everything inside {} is part of the class. It's all declarations.

A typical class (4)

Looking at the parts:

```
class TMyClass {
  int fNum;
public:
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

The data of the class. Any value of this class type will just be "an int", internally - its value will be a number.

A typical class (5)

Looking at the parts:

```
class TMyClass {
  int fNum;
public:
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

fNum is the result of the function call.

RICHIAMI DI C++

A typical class (6)

Looking at the parts:

```
class TMyClass {
  int fNum;
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

The function does not change the MyClass object - it's `const`

40

A typical class (7)

Looking at the parts:

```
class TMyClass {
  int fNum;
public:
  void Set(int n) { fNum = n; }
};
```

This function takes a parameter - and sets the data member to the new value `n`.

41

A typical class (8)

Looking at the parts:

```
class TMyClass {
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

A "constructor" is called by the compiler when an object is created:

```
MyClass obj;
```

42

A typical class (9)

Looking at the parts:

```
class TMyClass {
  TMyClass(): fNum(42) {}
  int Get() const { return fNum; }
  void Set(int n) { fNum = n; }
};
```

Constructors can have member initializers and a function body `{}`

43

RICHIAMI DI C++

Using a class

Put MyClass to action. What does this print?

```
void func() {
  MyClass obj;
  // printf() on screen
  //   "%d" means "an int"
  //   "\n" means "new line"
  printf("%d\n", obj.Get());
}
```

```
class MyClass {
  int fNum;
public:
  MyClass(): fNum(42) {}
  int Get() { return fNum; }
//...
};
```

44

- Who is calling func(). ??? It can be
  - another function, that in turns os called by another function,
  - a method of another class …

The outermost function

C++ has main()

```
int main(int, char**) {
  func();
  return 0;
}
```

47

The outermost function (2)

ROOT uses file name:

calls `MyCode()` for code in MyCode.C

root -l MyCode.C

or

root [0] .x MyCode.C

```
// MyCode.C:
int MyCode() {
  func();
  return 0;
}
```

48

POINTERS

Pointers

Values are in memory, at a location (address)

```
int value = 17; // the value
int* addr = &value;
```

`&` takes the address of a value

`addr` now contains the memory address of `value`

51

Pointers (2)

Can access values through pointers:

```
int value = 17; // the value
int* addr = &value;
*addr = 42;
```

`*addr` goes through the reference:
assigns 42 to value!

52

Pointers (2): AAA

Can access values through pointers:

```
auto value = 17; // the value
auto addr = &value;
*addr = 42;
```

`*addr` goes through the reference:
assigns 42 to value!

53

Pointers (3)

Can access members through pointers:

```
// call as print(&obj)
void print(TMyClass* ptr) {
  printf("value is %d\n", ptr->Get());
}
```

`->` accesses member of object pointed to

54

LIFETIME OF VARIABLES

Scope

Variables have a lifetime

```
void f() {
  TNamed n("a", "b");
} // destroys n
```

55

Scope (2)

It's the `{}` that defines lifetime:

```
void f(bool flag) {
  if (flag) {
    TNamed n("a", "b");
  } // destroys n
  // ERROR: n is not known here!
  printf("%s\n", n.GetName());
}
```

56

Heap

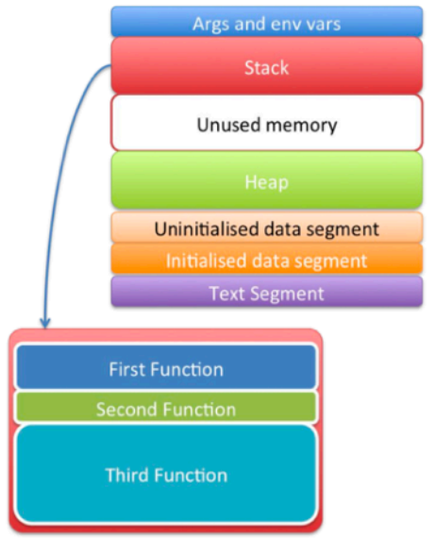
`new` creates, `delete` destroys

```
auto ptr = new int(17);
delete ptr;
```

57

A Program in Memory

- **Text Segment**: code to be executed.
- **Initialized Data Segment**: global variables initialized by the programmer.
- **Uninitialized Data Segment**: This segment contains uninitialized global variables.
- **The stack**: The stack is a collection of stack frames. It grows whenever a new function is called. "Thread private".
- **The heap**: Dynamic memory (e.g. requested with "new").



58

ROOT & MACROS

ROOT As a Calculator

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots = \sum_{n=0}^{\infty} x^n$$

Here we make a step forward.
We declare **variables** and use a ***for*** control structure.

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

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

64

ROOT Macros

- We have seen how to interactively type lines at the prompt
- The next step is to write "ROOT Macros" – lightweight programs
- The general structure for a macro stored in file *MacroName.C* is:

Function, no main, same name as the file

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

70

Controlling ROOT

- Special commands which are not C++ can be typed at the prompt, they start with a "."

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

- For example:
  - To quit root use **.q**
  - To issue a shell command use **.! <OS_command>**
  - To load a macro use **.L <file_name>** (see following slides about macros)
  - **.help** or **.?** gives the full list

65

Running a Macro

- A macro is executed at the system prompt by typing:

```
> root MacroName.C
```

- or executed at the ROOT prompt using .x:

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

- or it can be loaded into a ROOT session and then be run by typing:

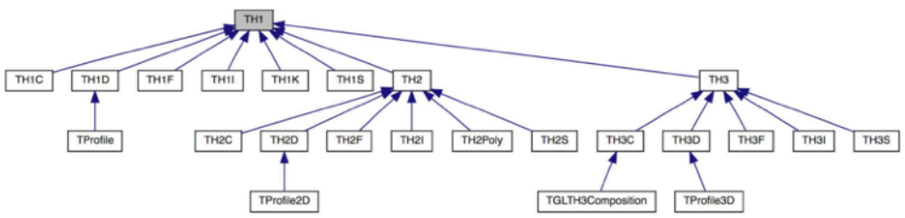
```
root [0] .L MacroName.C
root [1] MacroName();
```

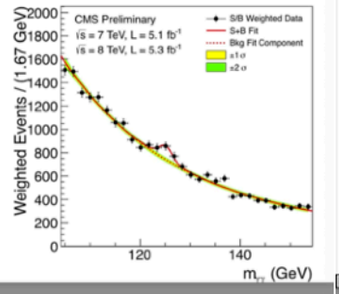
72

HISTOGRAMS, GRAPHS AND FUNCTIONS

Histograms

- Simplest form of data reduction
  - Can have billions of collisions, the Physics displayed in a few histograms
  - Possible to calculate momenta: mean, rms, skewness, kurtosis ...
- Collect quantities in discrete categories, the bins
- ROOT Provides a rich set of histogram types
  - We'll focus on histogram holding a *float* per bin

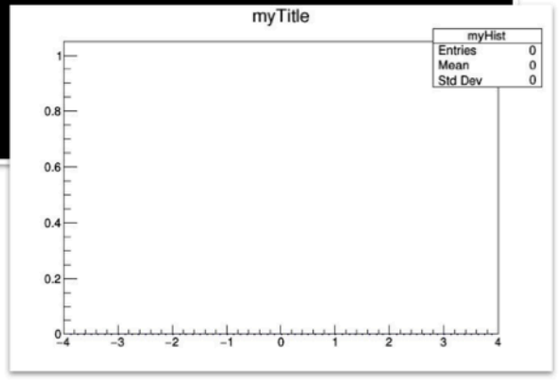




78

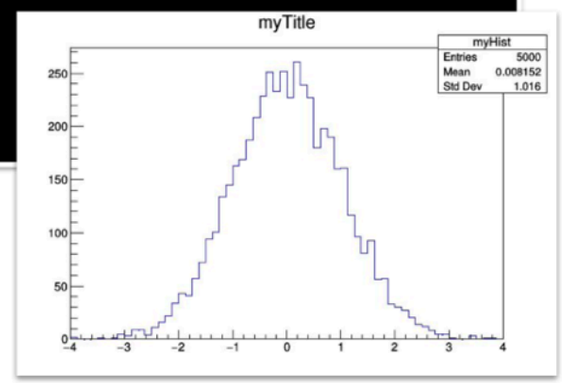
My First Histogram

```
root [0] TH1F h("myHist", "myTitle", 64, -4, 4)
root [1] h.Draw()
```



My First Histogram

```
root [0] TH1F h("myHist", "myTitle", 64, -4, 4)
root [1] h.Draw()
root [2] h.Draw()
```



79

Interlude: Scope

Bad for graphics:

```
// makeHist.C:
void makeHist() {
  TH1F hist("hist", "My Histogram");
  hist.Draw(); // shows histogram
} // destroys hist
```

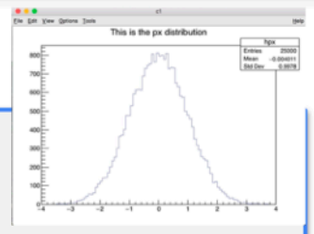
ROOT doesn't show my histogram!

81

HISTOGRAMS, GRAPHS AND FUNCTIONS

Interlude: Heap

Need a way to control lifetime




```
// makeHist.C:
void makeHist() {
  TH1F *hist = new TH1F("hist","My Histogram");
  hist->Draw(); // shows histogram
} // does not destroy hist!
```

`new` puts object on “heap”, escapes scope

82

Statistics and Fit parameters

- ROOT histograms have additional information called "statistics"
- ROOT adds them automatically to the plot when a histogram is drawn
- They can be turned on or off with the histogram method `SetStats()`
- `gStyle->SetOptStat()` defines which statistics parameters must be shown.

Also, after a fit, the fit result can be drawn on the plot and customized with `gStyle->SetOptFit()`

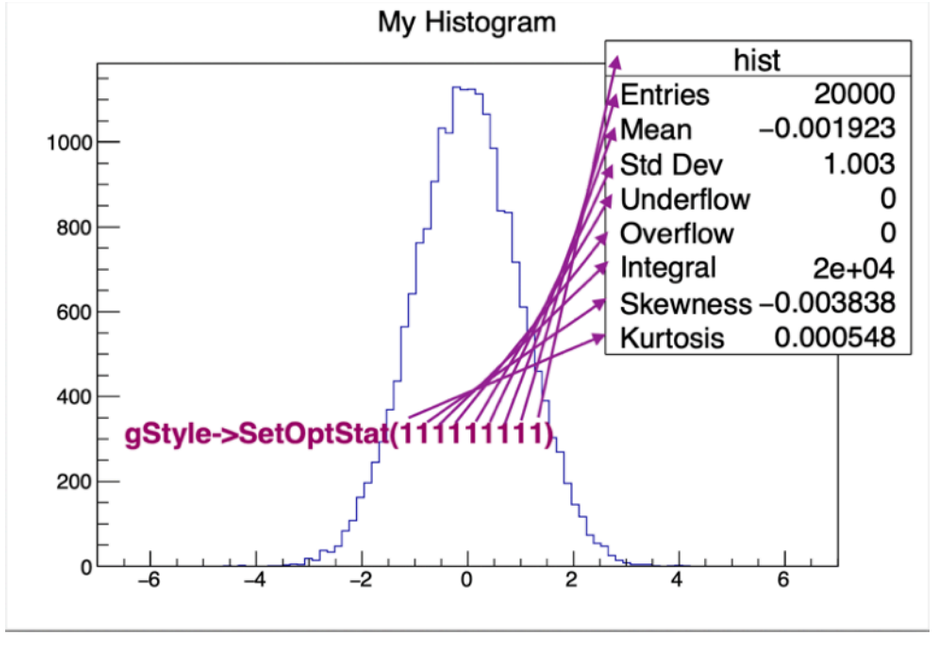
hpx	
Entries	25000
Mean	−0.004011
Std Dev	0.9978

- gStyle->SetOptStat(0) ==>> elimina il box della statistica
- gStyle->SetOptStat(1111) ===>> mostra il box con titolo, entries, media e dev standard

83

```
mb-spagnolo:11:54 [ ~/Documents ] root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.24/04                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Aug 25 2021, 12:59:28                   |
  | From tags/v6-24-04@v6-24-04                                      |
  | With Apple clang version 12.0.5 (clang-1205.0.22.9)              |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] void MakeHist(){
root (cont'ed, cancel with .@) [1]TH1F * hist = new TH1F("h1","myHistogram",100,0.,100);
root (cont'ed, cancel with .@) [2]hist->Draw();
root (cont'ed, cancel with .@) [3]}
root [4] MakeHist()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [5] .ls
 OBJ: TH1F      h1      myHistogram : 0 at: 0x11f476b50
root [6]
```

Elenca gli oggetti in memoria



HISTOGRAMS, GRAPHS AND FUNCTIONS

editor di testo:
notepad, wordpad (Windows)
emacs su sistemi Linux
(compreso Mac)

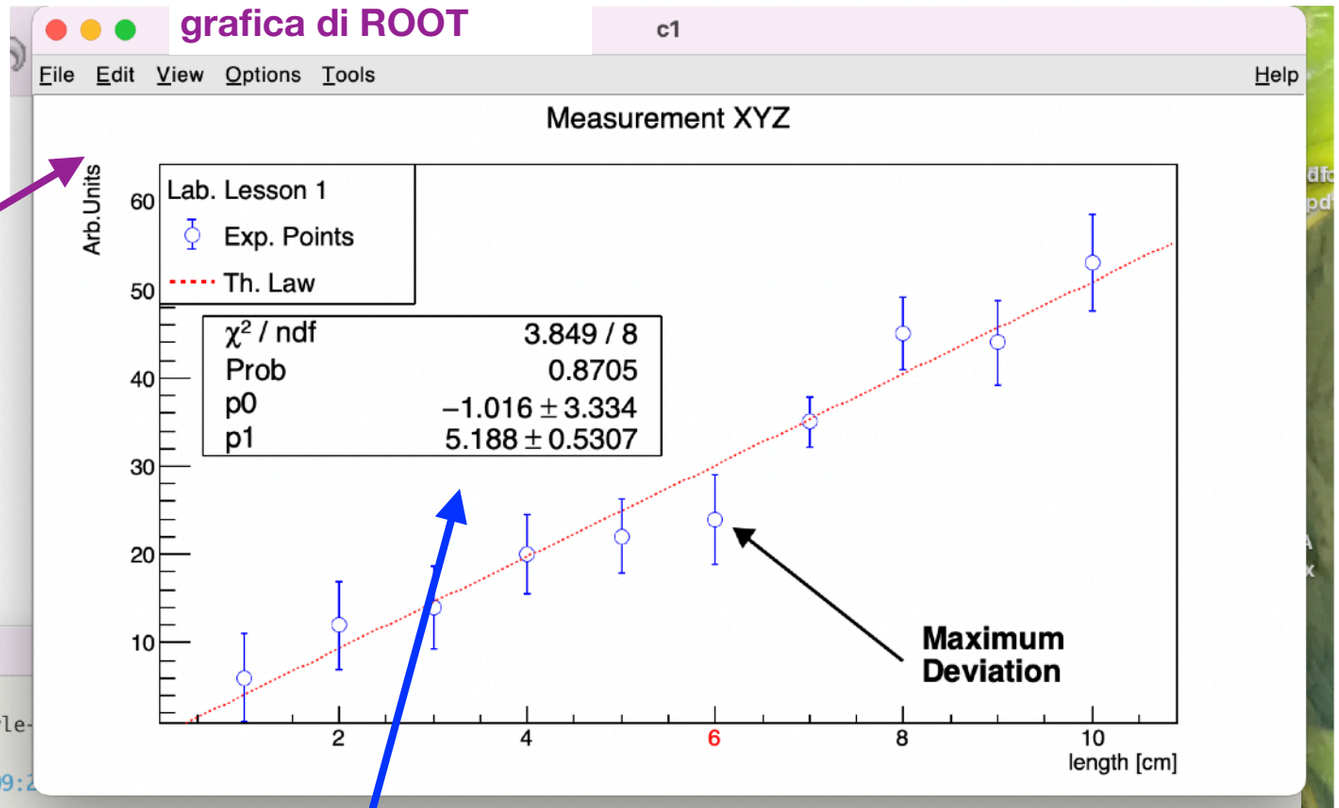
macroCollection.C

```
void macro1() {

  // The values and the errors on the Y axis
  const int n_points=10;
  double x_vals[n_points] = {1,2,3,4,5,6,7,8,9,10};
  double y_vals[n_points] = {6,12,14,20,22,24,35,45,44,53};
  double y_errs[n_points] = {5,5,4.7,4.5,4.2,5.1,2.9,4.1,4.8,5.43};

  // Instance of the graph
  auto graph = new TGraphErrors(n_points,x_vals,y_vals,nullptr,y_errs);
  graph->SetTitle("Measurement XYZ;length [cm];Arb.Units");

  // Make the plot esthetically better
  graph->SetMarkerStyle(kOpenCircle);
  graph->SetMarkerColor(kBlue);
  graph->SetLineColor(kBlue);

  // The canvas on which we'll draw the graph
  auto mycanvas = new TCanvas();
  // Draw the graph !
  graph->Draw("APE");

  // Define a linear function
  auto f = new TF1("Linear law","[0]+x*[1]",.5,10.5);
  // Let's make the function line nicer
  f->SetLineColor(kRed);
  f->SetLineStyle(2);

  // Fit it to the graph and draw it
  graph->Fit(f);
  // Build and Draw a legend
  auto legend = new TLegend(.1,.7,.3,.9,"Lab. Lesson 1");
  legend->AddEntry(graph,"Exp. Points","PE");
  legend->AddEntry(f,"Th. Law", "L");
  legend->Draw();

  // Draw an arrow on the canvas
  auto arrow = new TArrow(8,8,6.2,23,0.02,"|>");
  arrow->SetLineWidth(2);
  arrow->Draw();

  // Add some text to the plot and highlight the 3rd label
  auto text = new TLatex(8.2,7.5,"#splitline{Maximum}{Deviation}");
  text->Draw();
  graph->GetXaxis()->ChangeLabel(3,-1,-1,-1,kRed);
}
```

-:--- **macroCollection.C** 19% L56 (C/*l Abbrev)

Fit del grafico con una funzione

Canvas, finestra grafica di ROOT




```
   2  p1
root [2] gStyle-
root [3] .q
mb-spagnolo:09:2
[1] 21231
mb-spagnolo:09:28 [ ~ ] root
   ------------------------------------------------
  | Welcome to ROOT 6.24/04                    https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Aug 25 2021, 12:59:28                |
  | From tags/v6-24-04@v6-24-04                                   |
  | With Apple clang version 12.0.5 (clang-1205.0.22.9)           |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'    |
   ------------------------------------------------

root [0] .L macroCollection.C
root [1] macro1()
 FCN=3.84883 FROM MIGRAD    STATUS=CONVERGED      31
                    EDM=5.96982e-12    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0          -1.01604e+00   3.33409e+00   1.48321e-03  -8.98235e-12
   2  p1           5.18756e+00   5.30717e-01   2.36095e-04   9.40487e-12
root [2] gStyle->SetOptFit(1111)
root [3]
```

Avvio root da **terminale (prompt di comandi)**

Carico in memoria le funzioni, classi, ecc nel file

Eseguo la funzione macro1()

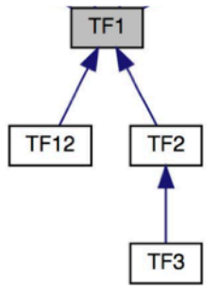
Aggiungo al plot il box di testo con le le informazioni sul risultato del fit

FUNCTIONS

Functions

- Functions are represented by the **TF1** class
- They have names, formulas, line properties, can be evaluated as well as their integrals and derivatives
  - Numerical techniques for the time being

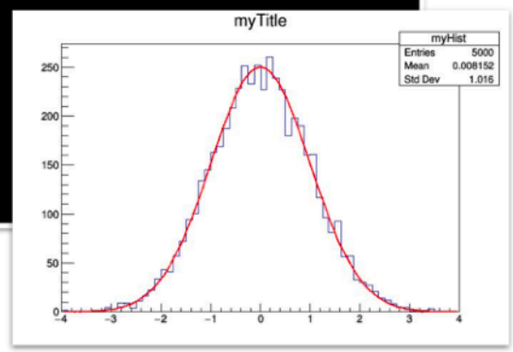
option	description
"SAME"	superimpose on top of existing picture
"L"	connect all computed points with a straight line
"C"	connect all computed points with a smooth curve
"FC"	draw a fill area below a smooth curve



84

My First Function

```
root [0] TH1F h("myHist", "myTitle", 64, -4, 4)
root [1] h.FillRandom("gaus")
root [2] h.Draw()
root [3] TF1 f("g", "gaus", -8, 8)
root [4] f.SetParameters(250, 0, 1)
root [5] f.Draw("Same")
```



86

Functions

Can describe functions as:

- Formulas (strings)
- C++ functions/functors/lambdas
  - Implement your highly performant custom function
- With and without parameters
  - Crucial for fits and parameter estimation

85

ROOT as a Function Plotter

- The class TF1 represents one-dimensional functions (e.g. *f(x)*):

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

- An extended version of this example is the definition of a function with parameters:

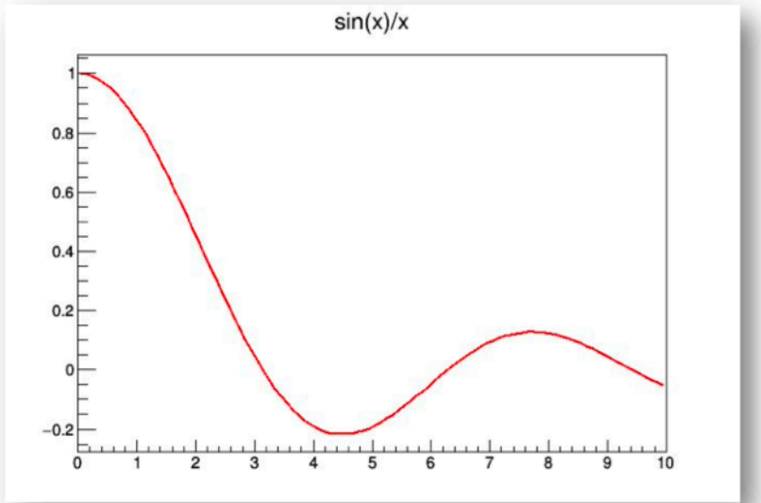
```
root [2] TF1 f2("f2","[0]*sin([1]*x)/x",0.,10.);
root [3] f2.SetParameters(1,1);
root [4] f2.Draw();
```

Try it!

87

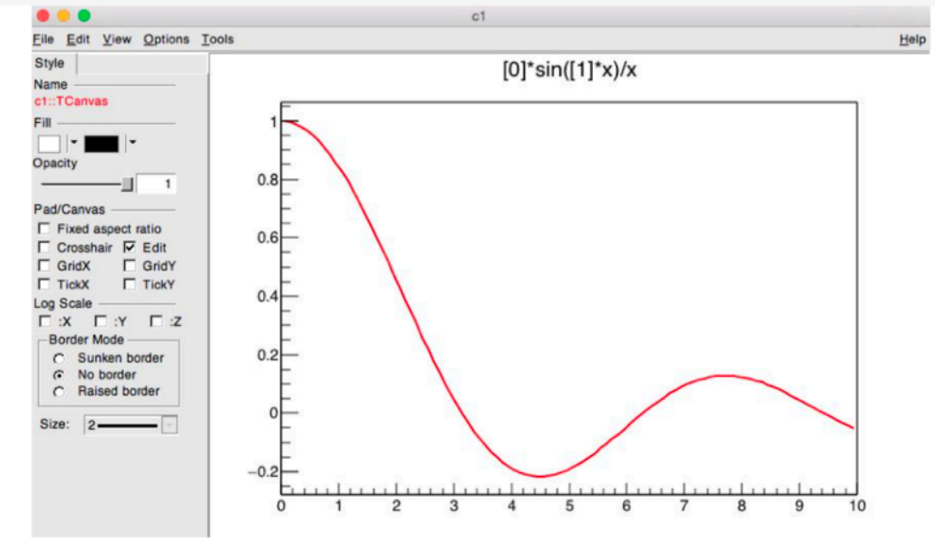
FUNCTIONS

ROOT as a Function Plotter



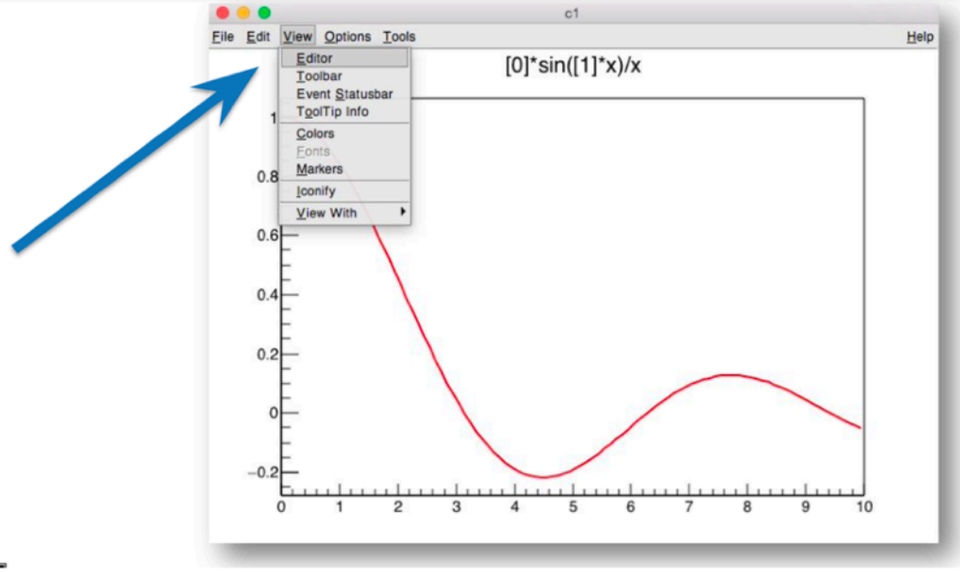
88

ROOT as a Function Plotter



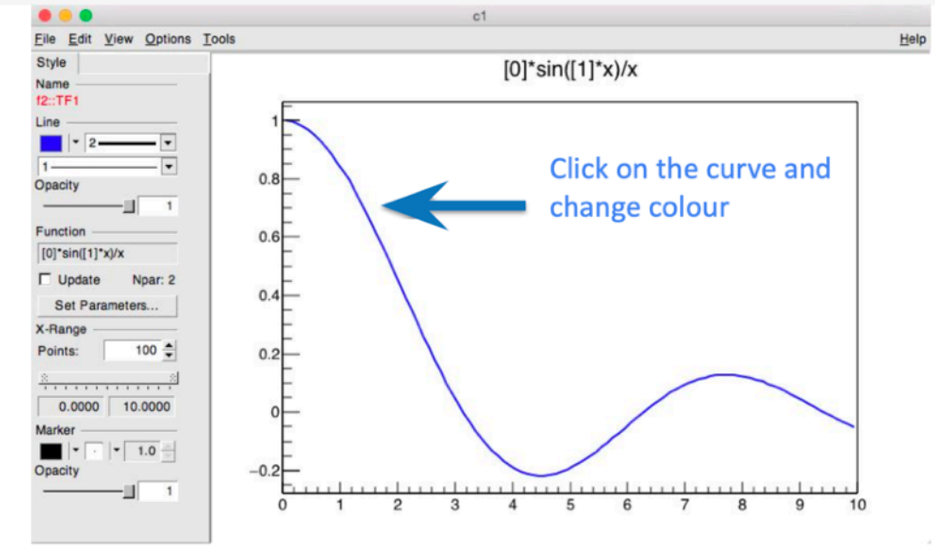
90

ROOT as a Function Plotter



89

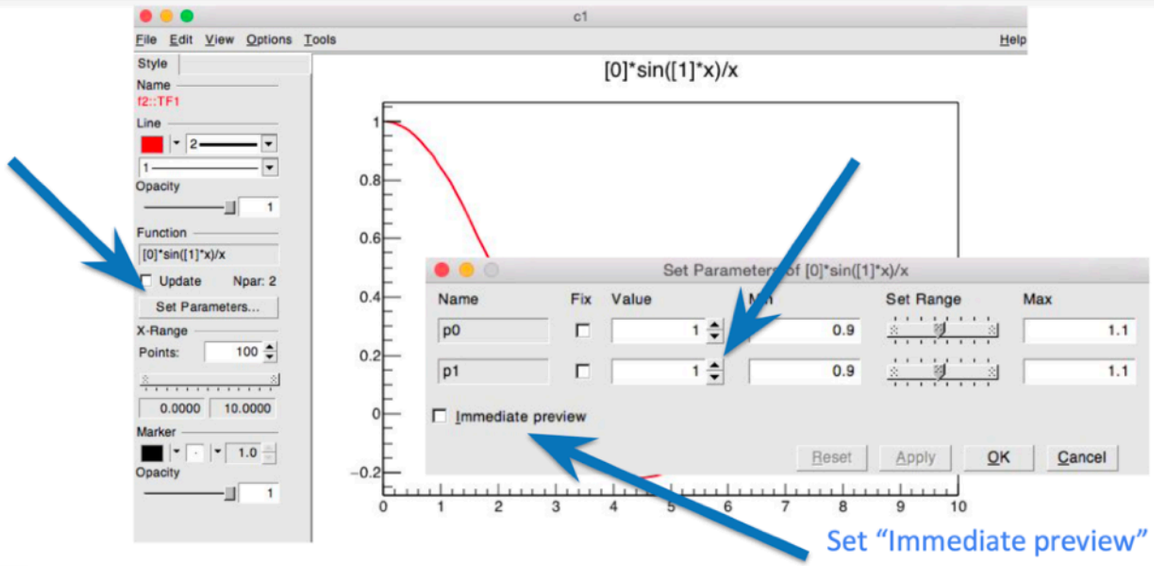
ROOT as a Function Plotter



91

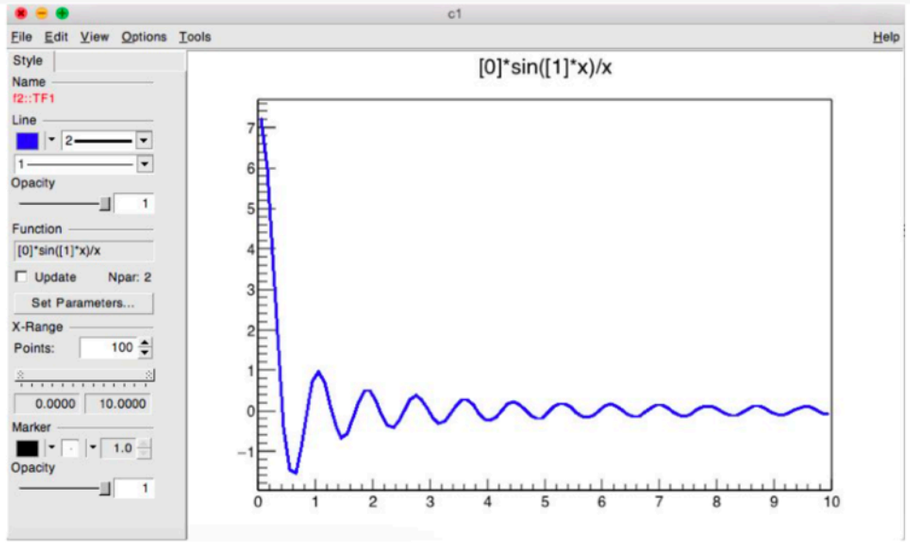
FUNCTIONS

ROOT as a Function Plotter



92

ROOT as a Function Plotter

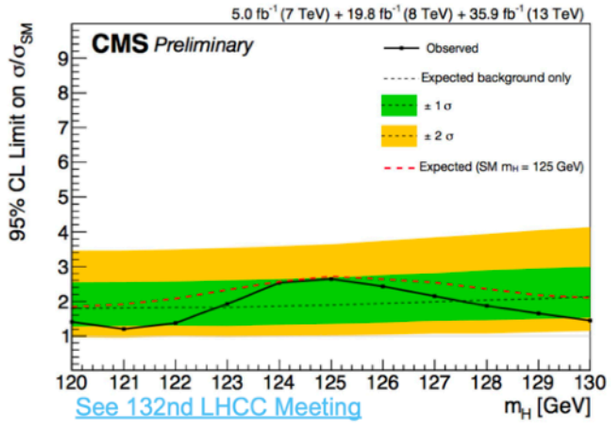


93

GRAPHS

Graphs

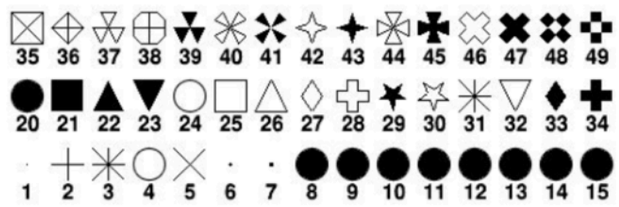
- Display points and errors
- Not possible to calculate momenta
- Not a data reduction mechanism
- **Fundamental to display trends**
- Focus on TGraph and TGraphErrors classes in this course



See 132nd LHCC Meeting

97

The Markers



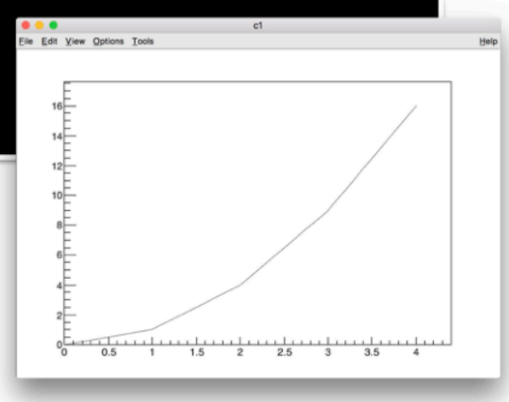

```
kDot=1, kPlus, kStar, kCircle=4, kMultiply=5,
kFullDotSmall=6, kFullDotMedium=7, kFullDotLarge=8,
kFullCircle=20, kFullSquare=21, kFullTriangleUp=22,
kFullTriangleDown=23, kOpenCircle=24, kOpenSquare=25,
kOpenTriangleUp=26, kOpenDiamond=27, kOpenCross=28,
kFullStar=29, kOpenStar=30, kOpenTriangleDown=32,
kFullDiamond=33, kFullCross=34 etc…
```

Also available through more friendly names ☺️

100

My First Graph

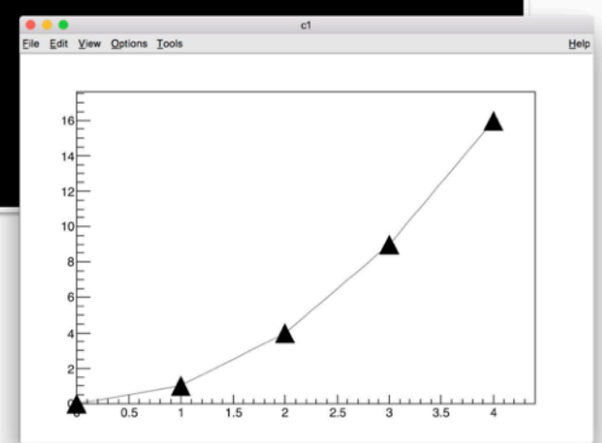
```
root [0] TGraph g;
root [1] for (auto i : {0,1,2,3,4}) g.SetPoint(i,i,i*i)
root [2] g.Draw("APL")
```



98

My First Graph

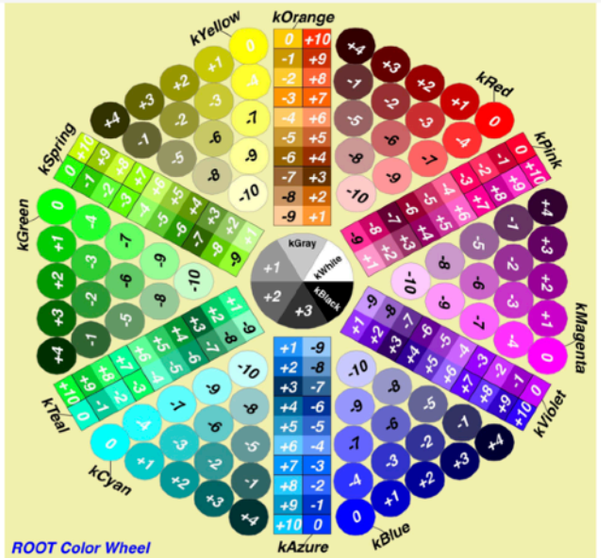
```
root [3] g.SetMarkerStyle(kTriangleUp)
root [4] g.SetMarkerSize(3)
```



102

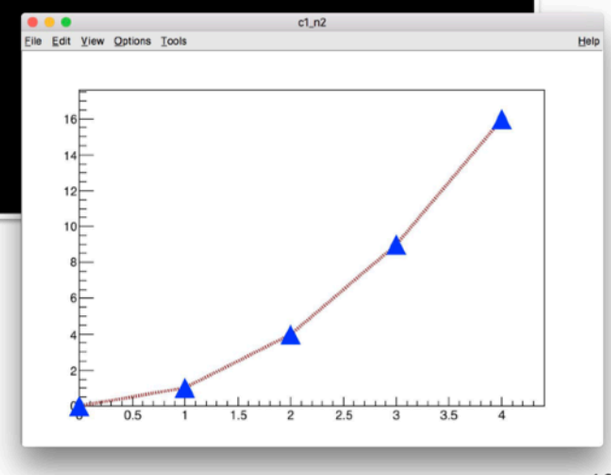
GRAPHS

The Colors (TColorWheel)



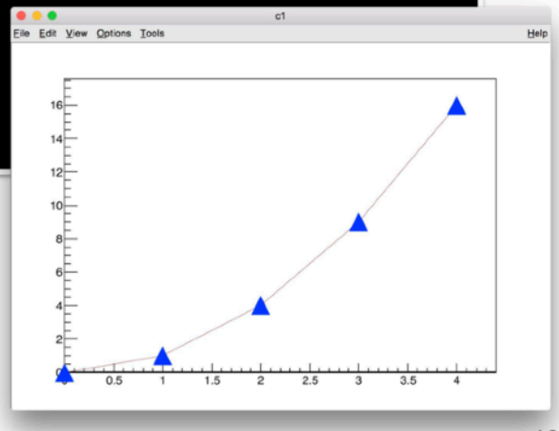
My First Graph

```
root [7] g.SetLineWidth(2)
root [8] g.SetLineStyle(3)
```



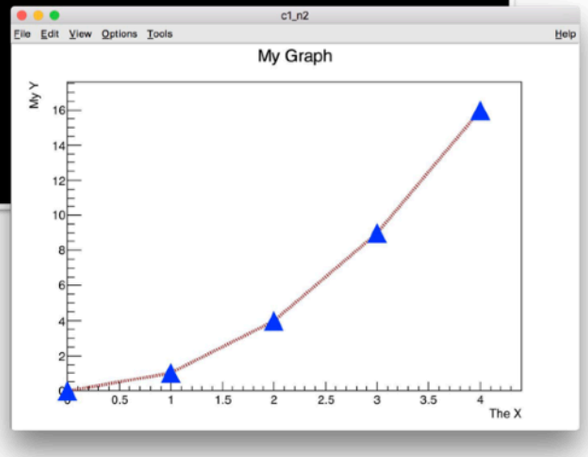
My First Graph

```
root [5] g.SetMarkerColor(kAzure)
root [6] g.SetLineColor(kRed - 2)
```



My First Graph

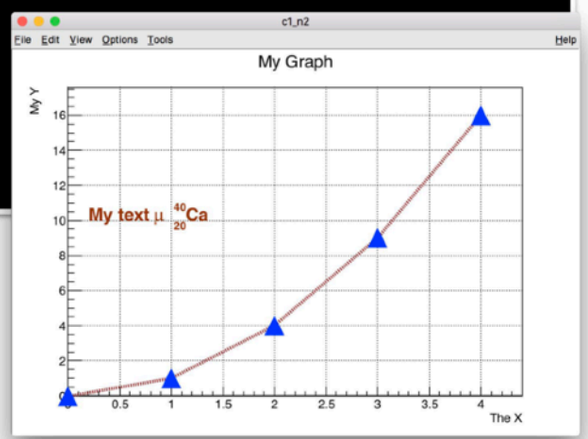
```
root [9] g.SetTitle("My Graph;The X;My Y")
```



GRAPHS

My First Graph

```
root [10] auto txt = "#color[804]{My text #mu {}^{40}_{20}Ca}"
root [11] TLatex l(.2, 10, txt)
root [12] l.Draw()
```



Features of a Good Plot

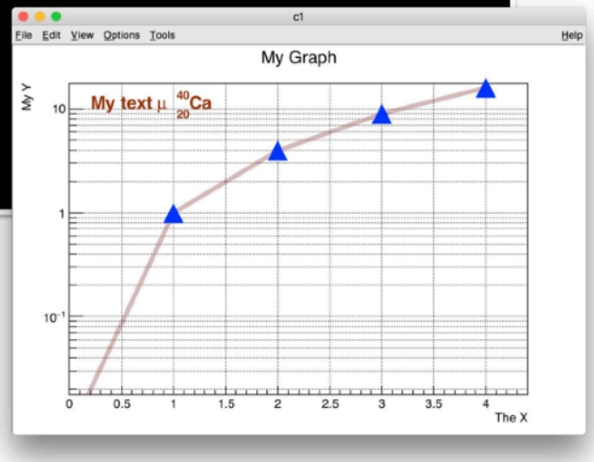
Every plot should be **self-contained** and deliver a **clear message**, even if extracted from the publication in which it's shown.

To achieve this goal the mandatory pieces of a plot are:

1. **The data**, of course. The best representation should be chosen to avoid any misinterpretation.
2. **The plot title** which should summarize clearly what the data are.
3. **The axis titles**. The X and Y titles should be properly titled with the variable name and its unit.
4. **The axis** themselves. They should be clearly labelled to avoid any ambiguity.
5. **The legend**. It should explain clearly the various curves on the plot.
6. **Annotations** highlighting specific details on the plot.

My First Graph

```
root [13] gPad->SetLogy();
```



myFirstGraph.C

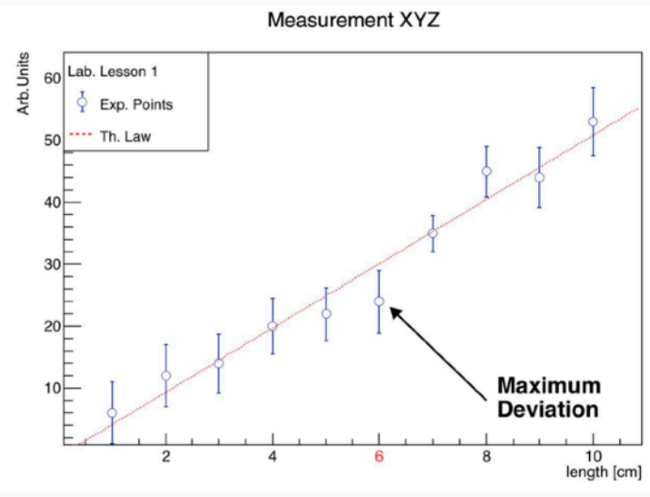
https://github.com/root-project/training/tree/master/BasicCourse/Exercises/Graphics

"Good Plot" example

The next slides will detailed how to build this plot step by step.

Each exercise will be marked as:

⇒ Something to do



GRAPHS & FUNCTIONS

The data set

```
void macro1(){
   // The number of points in the data set
   const int n_points = 10;
   // The values along X and Y axis
   double x_vals[n_points] = {1,2,3,4,5,6,7,8,9,10};
   double y_vals[n_points] = {6,12,14,20,22,24,35,45,44,53};
   // The errors on the Y axis
   double y_errs[n_points] = {5,5,4.7,4.5,4.2,5.1,2.9,4.1,4.8,5.43};

   // Instance of the graph
   auto graph = new TGraphErrors(n_points,x_vals,y_vals,nullptr,y_errs);
}
```

This code creates the **data set**.

⇒ Create a macro called "macro1.C" and execute it.

114

The data drawing

As this graph has error bars along the Y axis an obvious choice will be to draw it as an **error bars plot**. The command to do it is:

```
graph->Draw("APE");
```

The Draw() method is invoked with three options:

- "A" the **axis** coordinates are automatically computed to fit the graph data
- "P" **points** are drawn as marker
- "E" the **error bars** are drawn

⇒ Add this command to the macro and execute it again.

⇒ Try to change the options (e.g. “APEL”, “APEC”, “APE4”).

115

Add a Function

The data set we built up looks very linear. It could be interesting to **compare it with a line** to see what the linear law behind this data set could be.

```
// Define a linear function
auto f = new TF1("Linear law","[0]+x*[1]",.5,10.5);
// Let's make the function line nicer
f->SetLineColor(kRed);
f->SetLineStyle(2);
// Set parameters
f->SetParameters(-1,5);
f->Draw("Same")
```

The function "**f**" graphical aspect is customized the same way the graph was before.

⇒ Add this code to the macro. Execute it again.

⇒ Play with the graphical attributes for the "**f**" function.

118

Plot Titles

The graph was created without any titles. It is time to define them. The following command **define the three titles** (separated by a ";") in one go. The format is:

"Main title ; x axis title ; y axis title"

```
graph->SetTitle("Measurement XYZ;length [cm];Arb.Units");
```

The axis titles can be also set individually:

```
graph->GetXaxis()->SetTitle("length [cm]");
graph->GetYaxis()->SetTitle("Arb.Units");
```

⇒ Add this code to the macro. You can choose one or the other way.

⇒ Play with the text. You can even try to add TLatex special characters.

119

2D FUNCTIONS

2D plots with color map (1)

A very common way to represent 2D histograms is the **color plot**. We will use the following macro to illustrate this kind of plot.

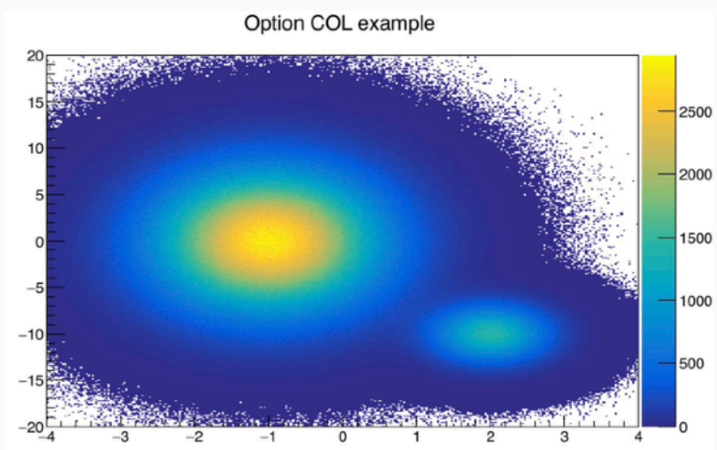
```
void macro2(){
   TH2F *h = new TH2F("h","Option COL example ",300,-4,4,300,-20,20);
   h->SetStats(0);
   h->SetContour(200);
   float px, py;
   for (int i = 0; i < 25000000; i++) {
      gRandom->Rannor(px,py);
      h->Fill(px-1,5*py);
      h->Fill(2+0.5*px,2*py-10.,0.1);
   }
   h->Draw("colz");
}
```

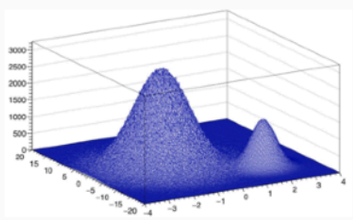
https://github.com/root-project/training/tree/master/BasicCourse/Exercises/Graphics
macro2.C

133

2D plots with color map (2)

The previous macro generates the following output which is a plot with two smooth 2D gaussian:





The default ROOT color map (kBird) render perfectly the smoothness of the dataset.

134

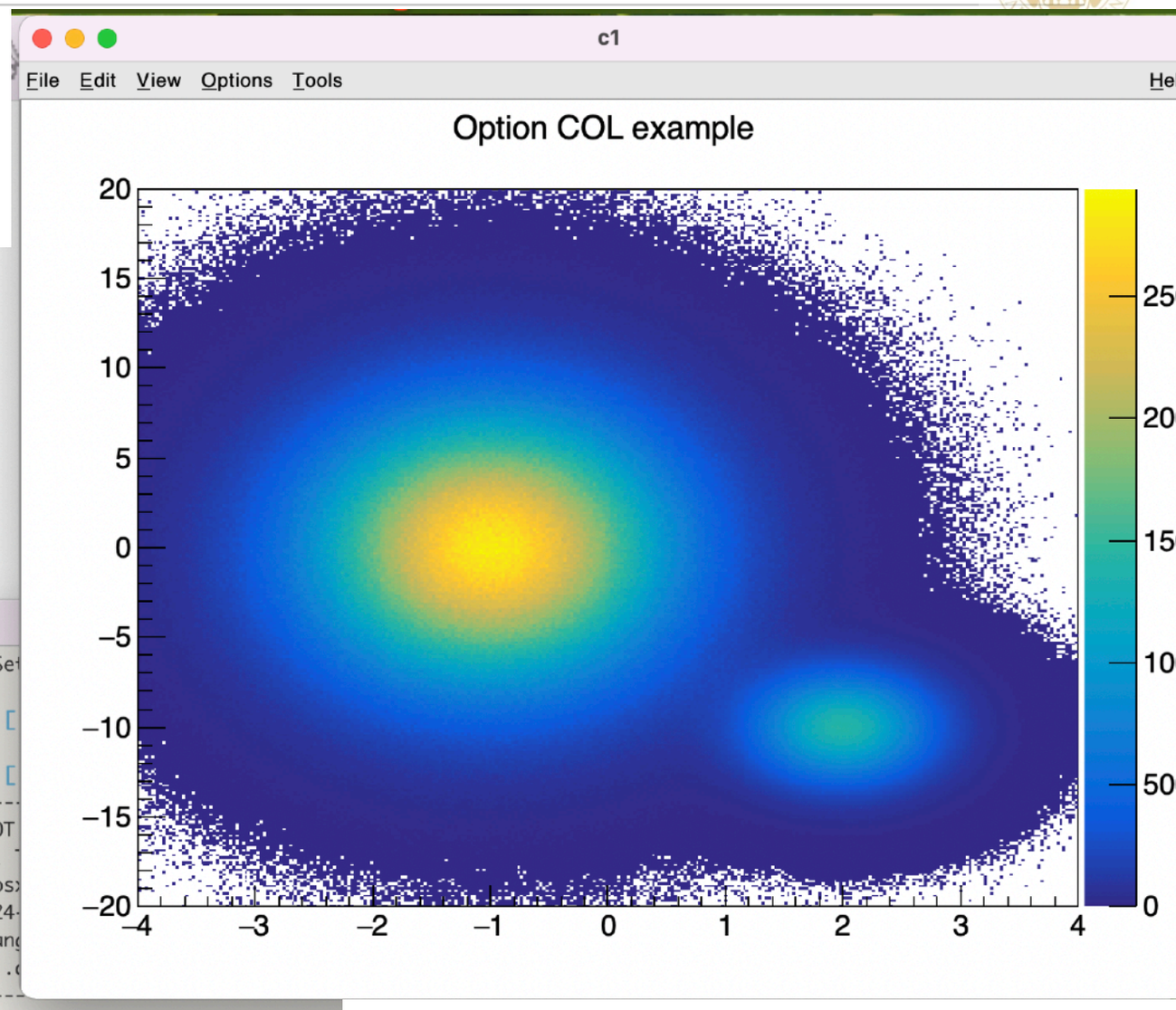
2D FUNCTIONS

editor di testo:
notepad, wordpad (Windows)
emacs su sistemi Linux
(compreso Mac)

macroCollection.C

```
// Make the plot esthetically better
graph->SetMarkerStyle(kOpenCircle);
graph->SetMarkerColor(kBlue);
graph->SetLineColor(kBlue);

// The canvas on which we'll draw the gr
auto mycanvas = new TCanvas();
// Draw the graph !
graph->Draw("APE");

// Define a linear function
auto f = new TF1("Linear law","[0]+x*[1]",.5,10.5);
// Let's make the function line nicer
f->SetLineColor(kRed);
f->SetLineStyle(2);

// Fit it to the graph and draw it
graph->Fit(f);
// Build and Draw a legend
auto legend = new TLegend(.1,.7,.3,.9,"Lab. Lesson 1");
legend->AddEntry(graph,"Exp. Points","PE");
legend->AddEntry(f,"Th. Law", "L");
legend->Draw();

// Draw an arrow on the canvas
auto arrow = new TArrow(8,8,6.2,23,0.02,"|>");
arrow->SetLineWidth(2);
arrow->Draw();

// Add some text to the plot and highlight the 3rd label
auto text = new TLatex(8.2,7.5,"#splitline{Maximum}{Deviation}");
text->Draw();
graph->GetXaxis()->ChangeLabel(3,-1,-1,-1,kRed);
}


void macro2(){
  TH2F *h = new TH2F("h","Option COL example ",300,-4,4,300,-20,20);
  h->SetStats(0);
  h->SetContour(200);
  float px, py;
  for (int i = 0; i < 25000000; i++) {
    gRandom->Rannor(px,py);
    h->Fill(px-1,5*py);
    h->Fill(2+0.5*px,2*py-10.,0.1);
  }
  h->Draw("colz");
}
```

-:--- macroCollection.C Bot L56 (C/*l Abbrev)




```
root [2] gStyle->Se
root [3] .q
mb-spagnolo:09:28 [
[1] 21231
mb-spagnolo:09:28 [
   ----------------
  | Welcome to ROOT
  | (c) 1995-2021,
  | Built for macosx
  | From tags/v6-24-
  | With Apple clang
  | Try '.help', '.
   ----------------

root [0] .L macroCollection.C
root [1] macro1()
 FCN=3.84883 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                     EDM=5.96982e-22    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0          -1.01604e+00   3.33409e+00   1.48321e-03  -8.98235e-12
   2  p1           5.18756e+00   5.30717e-01   2.36095e-04   9.40487e-12
root [2] gStyle->SetOptFit(1111)
root [3] macro2()
root [4]
```

Avvio root da **terminale (prompt di comandi)**
Carico in memoria le funzioni definite nel file

Eseguo la funzione macro2()

TTree

Da pag. 320 del tutorial

The TTree

A columnar dataset in ROOT is represented by **TTree**:

- Also called *tree*, columns also called *branches*
- An object type per column, **any type of object**
- One row per *entry* (or, in collider physics, *event*)

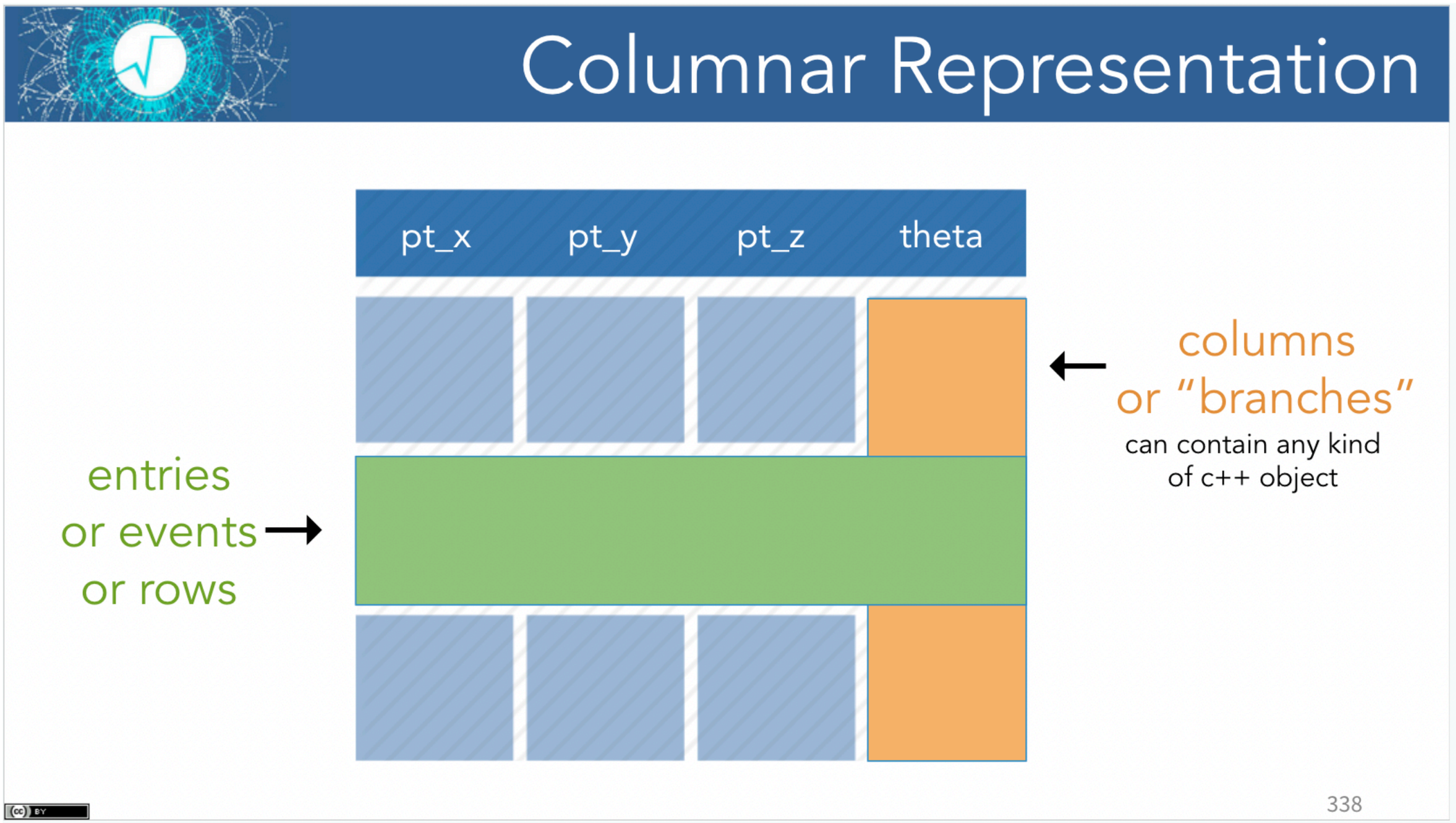
If just a **single number** per column is required, the simpler **TNtuple** *can* be used.

320

(cc) BY

TTree

Da pag. 320 del tutorial



TTree

Da pag. 320 del tutorial

Filling a TNtuple

```
TFile f("myfile.root", "RECREATE");
TNtuple myntuple("n", "n", "x:y:z:t");

// We assume to have 4 arrays of values:
// x_val, y_val, z_val, t_val

for (auto ievt: ROOT::TSeqI(128)) {
   myntuple.Fill(x_val[ievt], y_val[ievt],
                 z_val[ievt], t_val[ievt]);
}
myntuple.Write();
f.Close();
```

Names of the columns

Works only if columns are simple numbers

321

TTree

```
// determiniamo la soluzione dell'equazione
// differenziale y''=-0.000196y'^2+0.98
// con la consizione iniziale y(0)=0 e y'(0)=0 usando
// l'algoritmo di Eulero nell'intervallo
// 0<t<40
//
double fa(double t, double ya, double yb) {
   double f=-0.00196*ya*ya+9.8;
   return f;
}
double fb(double t, double ya, double yb) {
   double f=ya;
   return f;
}
void EuleroMult1() {
   // TTree che conterra il risultato della risoluzione numerica dell'eq. diff.
   // Apro file
    TFile * f=new TFile("EuleroMult1.root","RECREATE");
   // Creo il TTree
    TTree * T= new TTree("T","Equazioni Differenziali");
   // Le variabili del TTree
    double ya=0,yb=0,t=0;
    T->Branch("t",&t,"t/D");
    T->Branch("V",&ya,"ya/D");
    T->Branch("Y",&yb,"yb/D");
//step usato per la discretizzazione
    int NPoint=100000;
    double h;
// step dato l'intervallo e il numero di punti.
    h=50./NPoint;
// condizioni iniziali
    t=0;
    ya=0;
    yb=0;
    double ya_old=ya,yb_old=yb,t_old=t;
```

```
// Loop principale
    for (int i=0; i<NPoint; i++)  {
      t=t+h;
// formula di Eulero
      ya=ya_old+h*fa(t_old,ya_old,yb_old);
      yb=yb_old+h*fb(t_old,ya_old,yb_old);
// riempio TTree
      T->Fill();
// mi preparo a passo successivo
      ya_old=ya;
      yb_old=yb;
      t_old=t;
    }
//
    T->Write();
    f->Close();

}
```

TTree

Da pag. 320 del tutorial

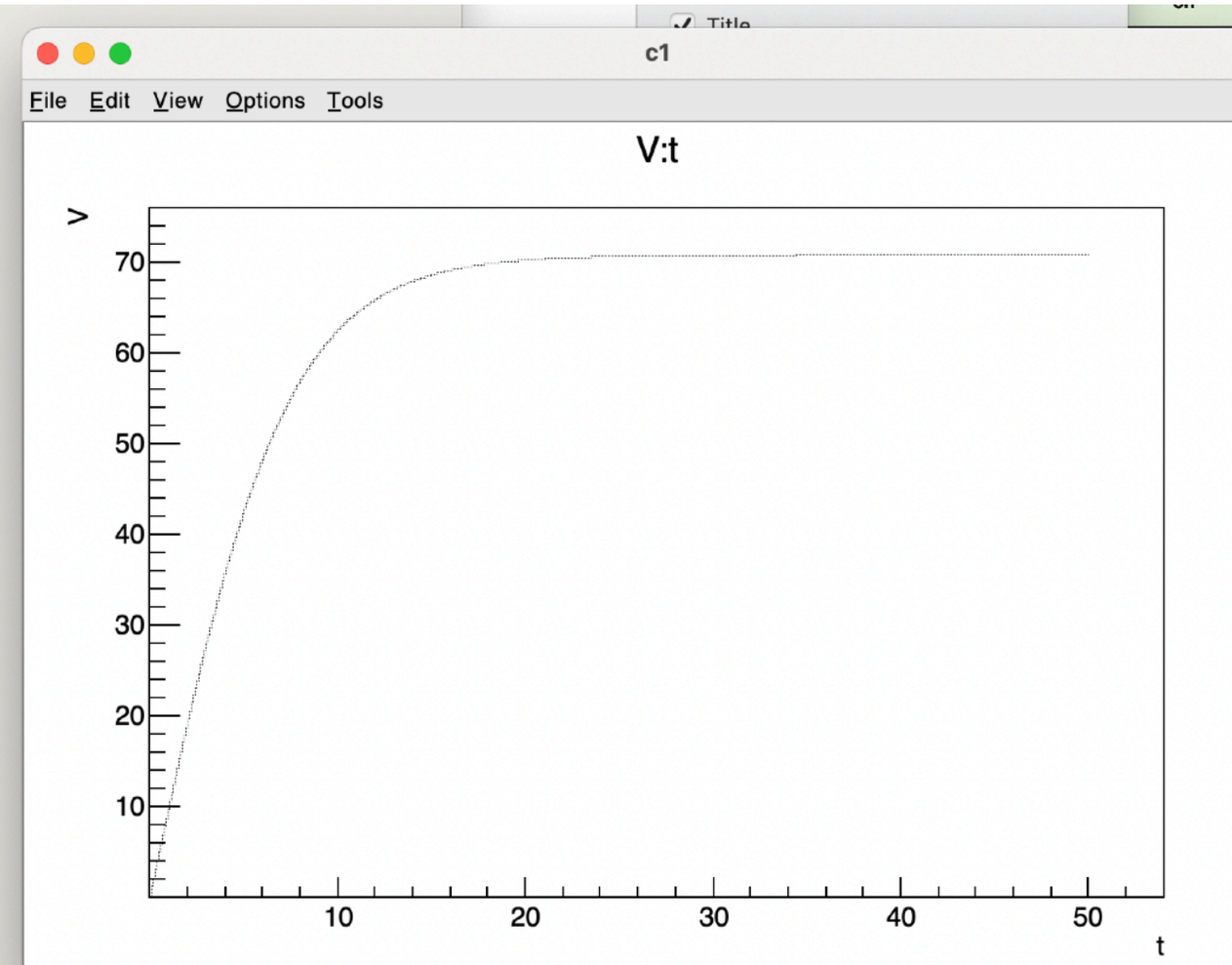
Reading a TNtuple

```
TFile f("hsimple.root");
TNtuple *myntuple;
f.GetObject("hsimple", myntuple);
TH1F h("h", "h", 64, -10, 10);
for (auto ievt: ROOT::TSeqI(myntuple->GetEntries()) {
   myntuple->GetEntry(ievt);
   auto xyzt = myntuple->GetArgs(); // Get a row
   if (xyzt[2] > 0) h.Fill(xyzt[0] * xyzt[1]);
}
h.Draw();
```

Works only if columns are simple numbers

322

(cc) BY

X

```
mb-spagnolo:08:52 [ ~/Documents/Didattica/MetodiStatComp/esempi/l21 ] root -l EuleroMult1.root
root [0]
Attaching file EuleroMult1.root as _file0...
(TFile *) 0x13397ef00
root [1] .ls
TFile**         EuleroMult1.root
 TFile*         EuleroMult1.root
  KEY: TTree    T;1     Equazioni Differenziali
root [2] T->Print()
******************************************************************************
*Tree    :T         : Equazioni Differenziali                                 *
*Entries :   100000 : Total =         2407906 bytes  File  Size =    1631033 *
*        :          : Tree compression factor =   1.48                       *
******************************************************************************
*Br    0 :t         : t/D                                                     *
*Entries :   100000 : Total  Size=     802507 bytes  File Size  =     381905 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   2.10     *
*.............................................................................*
*Br    1 :V         : ya/D                                                    *
*Entries :   100000 : Total  Size=     802510 bytes  File Size  =     582299 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   1.38     *
*.............................................................................*
*Br    2 :Y         : yb/D                                                    *
*Entries :   100000 : Total  Size=     802510 bytes  File Size  =     665669 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   1.20     *
*.............................................................................*
root [3] T->Draw("V:t")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [4]
root [4]
root [4]
root [4]
```