INTRODUCTION TO ROOT
&
BASIC APPLICATIONS

by **Alexis Pompili (pompili@ba.infn.it)**

Master course

Laboratorio di Analisi Dati

Esercitazione 0

How to access the Virtual Machine of this Course - I

ssh -Y pompili@212.189.205.223 … and I "land" in the directory **/home/pompili**

In your directory **/home/yourUserName** create the following script file **login_corso.sh**

```
#ROOT
source /cvmfs/cms.cern.ch/slc5_amd64_gcc481/lcg/root/5.34.18/bin/thisroot.sh
#GCC
source /cvmfs/cms.cern.ch/slc5_amd64_gcc481/external/gcc/4.8.1/etc/profile.d/init.sh

export PATH=$PATH:/afs/cern.ch/cms/slc5_amd64_gcc481/external/gcc/4.8.1/bin/
export PATH=$PATH:$ROOTSYS/bin:.

#LIBs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib:.
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cvmfs/cms.cern.ch/slc5_amd64_gcc481/external/gcc/4.8.1/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cvmfs/cms.cern.ch/slc5_amd64_gcc481/external/gcc/4.8.1/lib64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cvmfs/cms.cern.ch/slc5_amd64_gcc481/cms/cmssw/CMSSW_7_1_0/external/slc5_amd64_gcc481/lib
```

Dynamic libreries loaded at run time

Now execute this script file :

```
bash-3.2$ ssh -X pompili@212.189.205.223
pompili@212.189.205.223's password:
Last login: Thu Oct 12 17:46:11 2017 from dhcp147.ba.infn.it
-bash-3.2$ pwd
/home/pompili
-bash-3.2$ touch login_corso.sh
-bash-3.2$ vi login_corso.sh
-bash-3.2$ source login_corso.sh
-bash-3.2$
```

… this configure your environment to be able to use **ROOT** taken from some available CERN repository (without installing it locally !) [it could have been taken by **afs** but this is not possible for this VM]

How to access the Virtual Machine of this Course - II

Now you should be able to start **ROOT** :

```
-bash-3.2$ source login_corso.sh
-bash-3.2$ root
   ****************************************
   *                                      *
   *       W E L C O M E  to  R O O T      *
   *                                      *
   *   Version   5.34/18     14 March 2014  *
   *                                      *
   *  You are welcome to visit our Web site *
   *          http://root.cern.ch         *
   *                                      *
   ****************************************

ROOT 5.34/18 (heads/v5-34-00-patches@v5-34-17-156-g60f5d99, May 19 2014, 15:46:00 on linuxx8664gcc)

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

ROOT prompt

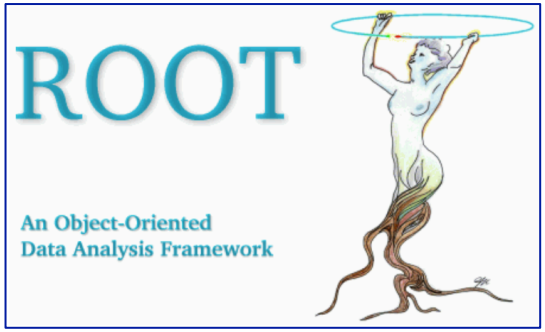
Introduction to ROOT/RooFit

In this course we are going to see some examples of usage of the ROOT system, a software framework for data (even *big-data*) analysis.

The ROOT system tool is today widely used in many High Energy Physics and Nuclear Physics applications, but also in many other fields of science, engineering and a growing use in many other domains. It has been continuously developed since 1995, and it has become a standard tool in HEP after 2000/1.

ROOT provides many libraries for visualization, maths & statistics, and also many interfaces to external systems. It includes an advanced I/O system to save and retrieve any object in an efficient way from local or network-wide data sets; it particularly supports reading experimental data objects evolving in time.

To start with have a look at: ROOT User Guides & Manuals



To start with have a look at: [ROOT User Guides & Manuals]

ROOT Guides

Title	Quick Link	All Links
Reference Guide	head / 6.08	all releases
User's Guide	6 Series (html)	all formats and series
ROOT Primer - Notebook Version (beta)	html pdf GitHub / SWAN	•
ROOT Primer	6 Series (html)	all formats and series
ROOT Primer 5	5 Series (pdf)	

Topical Manuals

Title	Quick Link	All Links
RooFit	Manual (pdf) / Quick Start Guide (pdf)	other formats
HTTP Server	6 Series (html)	all versions
JSROOT	6 Series (html)	all versions
CERNLib	(pdf)	-
Minuit	(pdf)	-
Minuit2	6 Series (html)	all formats and series
TSpectrum	6 Series (html)	all formats and series
TMVA	6 Series (external)	other links
PROOF	Drupal Book	other resources
VMC	Drupal Book	-

ROOT

"Diving Into ROOT"

Abstact:

ROOT is a software framework for data analysis, a powerful tool to cope with the demanding tasks typical of state of the art scientific data analysis. Among its prominent features are an advanced graphical user interface, ideal for interactive analysis, an interpreter for the C++ programming language, for rapid and efficient prototyping and a persistency mechanism for C++ objects, used also to write every year petabytes of data recorded by the Large Hadron Collider experiments. This introductory guide illustrates the main features of ROOT, relevant for the typical problems of data analysis: input and plotting of data from measurements and fitting of analytical functions.

RooFit

The Toolkit for Data Modeling with ROOT (RooFit) is a package that allows for modeling probability distributions in a compact and abstract way. It is distributed with ROOT

The RooFit library provides a toolkit for modeling the expected distribution of events in a physics analysis. Models can be used to perform unbinned maximum likelihood fits, produce plots, and generate "toy Monte Carlo" samples for various studies. RooFit was originally developed for the BaBar collaboration, a particle physics experiment at the Stanford Linear Accelerator Center. The software is primarily designed as a particle physics data analysis tool, but its general nature and open architecture make it useful for other types of data analysis also.

ROOT features

ROOT is a framework for data analysis.
It supports many operating systems.
It is a large set of libraries encoded in C++. It provides many ready classes.
The use of object oriented programming makes it easily extendible with external libraries.
It can be used interactively or as a library for external programs.

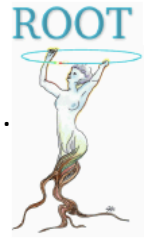
It has a C++ interpreter of the commands: **CINT** (it supports quite all the C++ commands)
Multi-row commands and macros (a set of C++ commands) must be included within {….}
It is possible to use the same code for compiled programs and interactive macros;
interactive macros can be tested interatctively and later integrated in a program.
Macros can be compiled automatically (**ACLIC: Automatic Compiler of Libraries for CINT**).
It is better to use compiled programs because they can be much more quickly executed.

The special commands of **CINT** start with the ".":
- to load macro : .L macro_name
- to execute macro : .x macro_name
- to exit **ROOT** : .q
- to print help : .h

ROOT basics: calculator - I

You can use ROOT interactive shell as a calculator!

ROOT offers you not only to type in C++ statements but also to use advanced mathematical functions living in the **TMath** namespace.
Some examples follow.



Start your **ROOT** session: `[pompili@ui03 ~]$ root -l` ----> avoids the appearance of the woman-tree .

At the **ROOT** prompt we proceed interactively:

```
root [0] sqrt(3)
(const double)1.73205080756887719e+00
root [1] TMath::Pi()
(Double_t)3.14159265358979312e+00
root [2] TMath::Erf(.2)
(Double_t)2.22702589210478447e-01
root [3] 5*898/(56-37)
(const int)236
```

ROOT basics: calculator - II

with $x=\frac{1}{2}$: $\sum_{i=0}^{+\infty}(1/2)^i=\frac{1}{1-1/2}=2$

A numerical example with a **geometrical series**: $\sum_{i=0}^{+\infty}x^i=\frac{1}{1-x}$ if $|x|<1$ [otherwise is irregular ($x<-1$) or diverges positively ($x\geq 1$)]

```
root [4] double x=.5
root [5] int N=30
root [6] double geom_series=0
root [7] for (int i=0;i<N;i++)geom_series+=TMath::Power(x,i)
root [8] geom_series
(double)1.99999999813735485e+00
```

$\cong 2$

$\sum_{i=0}^{30}x^i$

ROOT basics: calculator - II

with $x=\frac{1}{2}$: $\sum_{i=0}^{+\infty}(1/2)^i=\frac{1}{1-1/2}=2$

A numerical example with a **geometrical series**: $\sum_{i=0}^{+\infty}x^i=\frac{1}{1-x}$ if $|x|<1$ [otherwise is irregular ($x<-1$) or diverges positively ($x\geq 1$)]

```
root [4] double x=.5
root [5] int N=30
root [6] double geom_series=0
root [7] for (int i=0;i<N;i++)geom_series+=TMath::Power(x,i)
root [8] geom_series
(double)1.99999999813735485e+00
```

$\cong 2$ 

$\sum_{i=0}^{30}x^i$

This is a partial sum!
In general it holds for it :

$$\sum_{i=0}^{N}x^i=\frac{1-x^{N+1}}{1-x}$$

It can be verified :

ROOT basics: calculator - II

with $x=\frac{1}{2}$: $\sum_{i=0}^{+\infty}(1/2)^i=\frac{1}{1-1/2}=2$

A numerical example with a **geometrical series**: $\sum_{i=0}^{+\infty}x^i=\frac{1}{1-x}$ if $|x|<1$ [otherwise is irregular ($x<-1$) or diverges positively ($x\geq 1$)]

```
root [4] double x=.5
root [5] int N=30
root [6] double geom_series=0
root [7] for (int i=0;i<N;i++)geom_series+=TMath::Power(x,i)
root [8] geom_series
(double)1.9999999813735485e+00
```

$\cong 2$

$\sum_{i=0}^{30}x^i$

This is a partial sum!
In general it holds for it :

$$\sum_{i=0}^{N}x^i=\frac{1-x^{N+1}}{1-x}$$

It can be verified :

```
root [9] TMath::Abs(geom_series - (1-TMath::Power(x,N+1))/(1-x))
(Double_t)9.3132257461547851e-10
```

thus proving that $\left|\left(\sum_{i=0}^{N=30}x^i\right)-\left(\frac{1-x^{30+1}}{1-x}\right)\right|\to 0$

ROOT basics: calculator - II

with $x = \frac{1}{2}$: $\sum_{i=0}^{+\infty}(1/2)^i = \frac{1}{1-1/2} = 2$

A numerical example with a **geometrical series**: $\sum_{i=0}^{+\infty} x^i = \frac{1}{1-x}$ if $|x| < 1$ [otherwise is irregular ($x < -1$) or diverges positively ($x \geq 1$)]

```
root [4] double x=.5
root [5] int N=30
root [6] double geom_series=0
root [7] for (int i=0;i<N;i++)geom_series+=TMath::Power(x,i)
root [8] geom_series
(double)1.9999999813735485e+00
```

$\cong 2$ 

$\sum_{i=0}^{30} x^i$

This is a partial sum!
In general it holds for it :

$$\sum_{i=0}^{N} x^i = \frac{1-x^{N+1}}{1-x}$$

It can be verified :

```
root [9] TMath::Abs(geom_series - (1-TMath::Power(x,N+1))/(1-x))
(Double_t)9.3132257461547851e-10
```

thus proving that $\left| \left(\sum_{i=0}^{N=30} x^i \right) - \left(\frac{1-x^{30+1}}{1-x} \right) \right| \to 0$

Note that there are slight differences between **`Cint`** & the standard C++ language.
For instance you do not need the ";" at the end of a line in interactive mode.

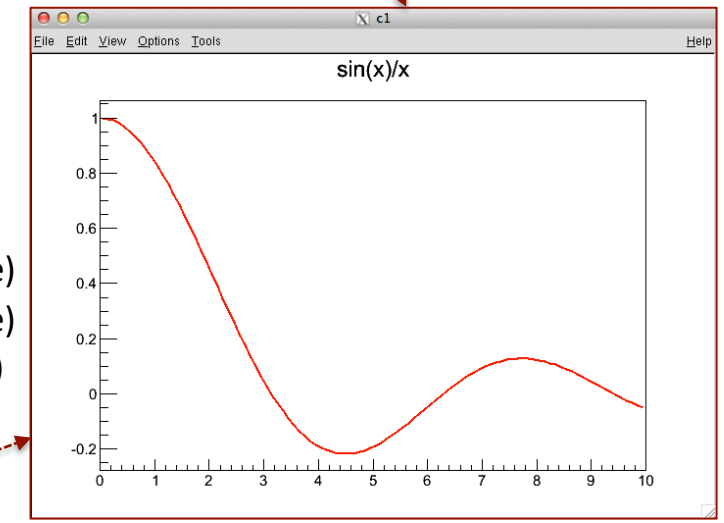
ROOT as a function plotter - I

For instance you can use the **ROOT**'s class **TF1** to display a function of one variable ***x*** :

```
root [0] TF1 *f1 = new TF1("f1","sin(x)/x",0.,10.)
root [1] f1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]
```

pointer to an instance of a TF1 class

arguments are used
in the constructor:
- name (of type string)
- function (of type string)
- parameter1 (of type double)
- parameter2 (of type double)
  (parameters define ***x*** range)



The **Draw()** method, here without any parameter, displays the function in a window which should pop up after typing!

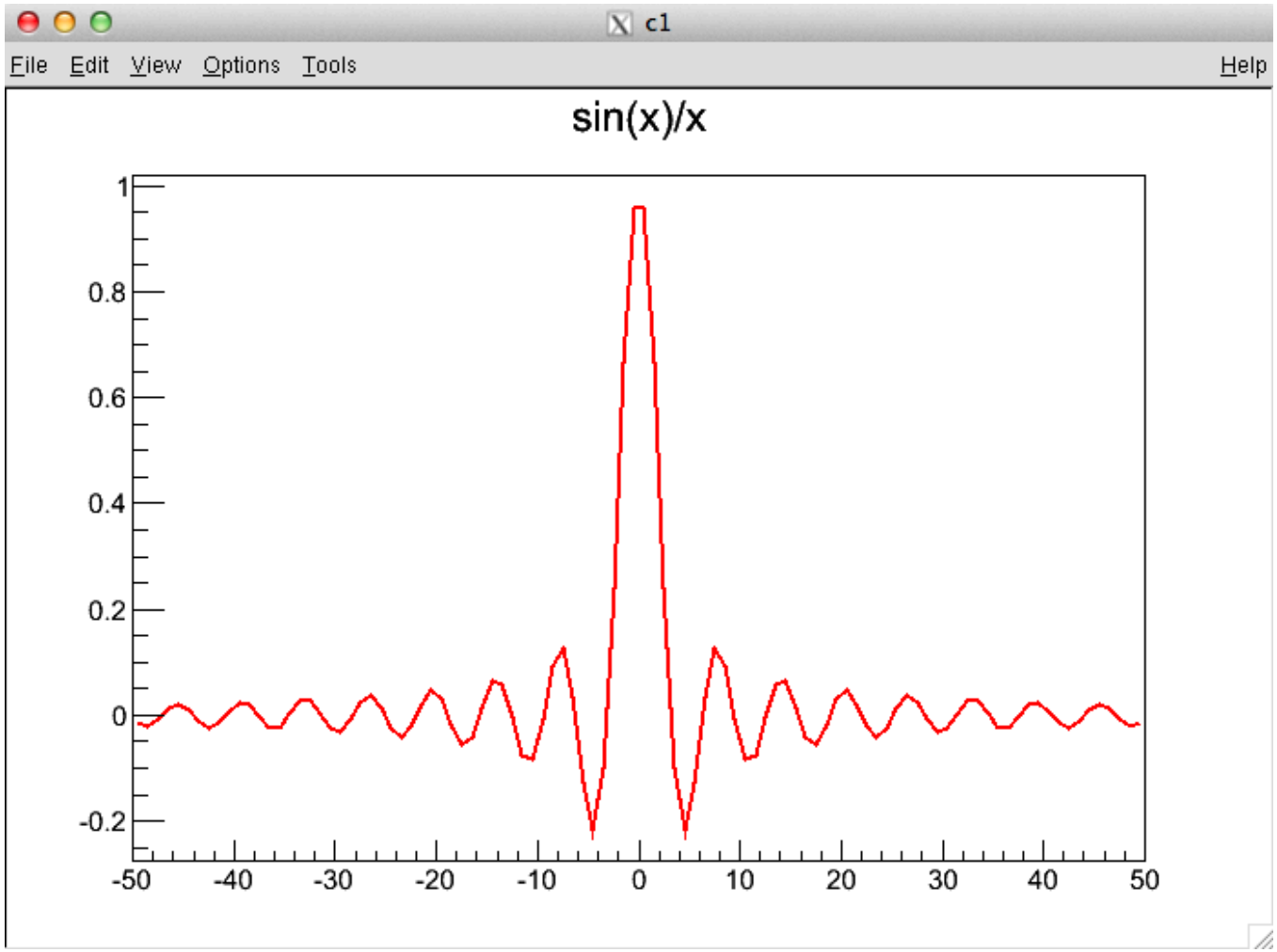
Note that all **ROOT** classes start with the letter **T** .

ROOT as a function plotter - II

Changing the range definition:

```
root [0] TF1 *f1 = new TF1("f1","sin(x)/x",-50.,50.)
root [1] f1->Draw()
```

… you get:

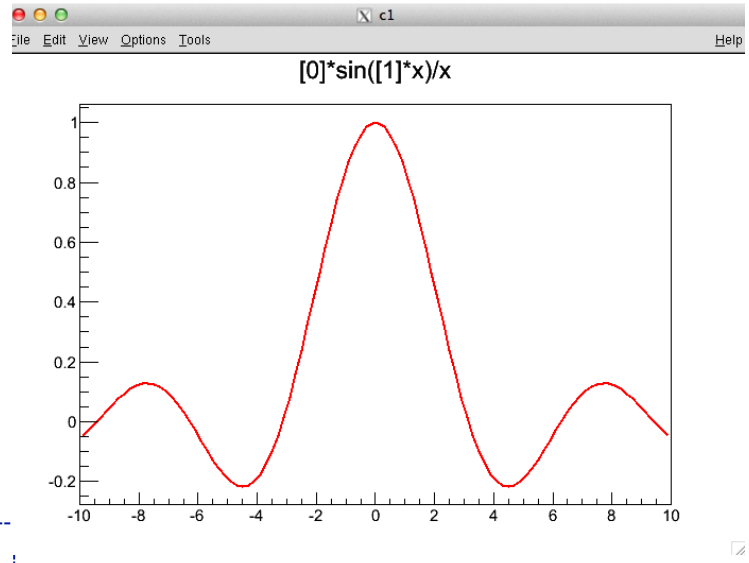


ROOT as a function plotter - II

Let's consider a slightly extended version of this example:

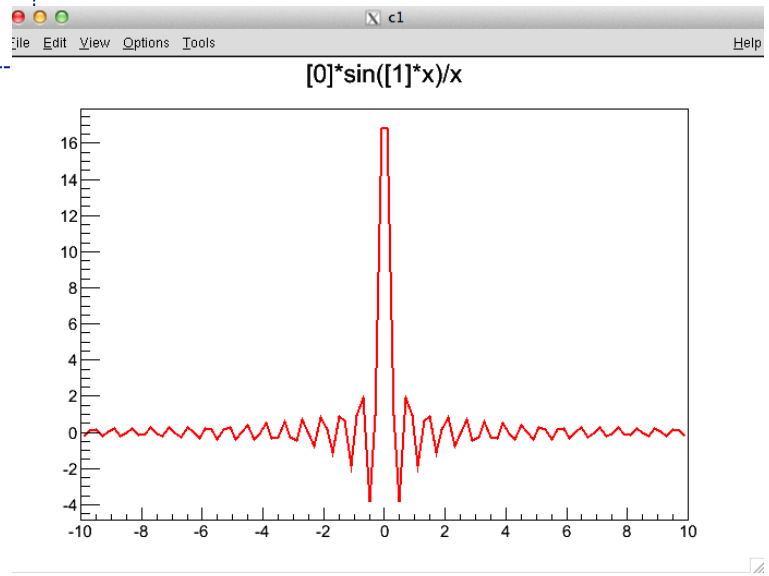
```
root [0] TF1 *f1 = new TF1("f1","[0]*sin([1]*x)/x",-10.,10.)
root [1] f1->SetParameter(0,1)
root [2] f1->SetParameter(1,1)
root [3] f1->Draw()
```

parameters



parameters's values assigned through the method
`SetParameter(<parameter_number>,<parameter_value>)`

```
root [1] f1->SetParameter(0,2)
root [2] f1->SetParameter(1,10)
root [3] f1->Draw()
```



Introduction to macros

If you have a number of lines which you were able to execute at the ROOT prompt, you can turn them into a ROOT macro, by giving them a name which corresponds to the file name without extension.

The general structure for a macro stored in filename myTestMacro.C is:

```
void myTestMacro() {
                        …
                        … my lines of C++ code …
                        …
}
```

It is easy to compile a macro as a stand-alone application by adding some include statements for header file or some "dressing code" to any macro.

Your test macro can be compiled by the ROOT C++ interpreter CINT: **`.L myTestMacro.C`**
`.L myTestMacro.C+` invokes **ACLIC (the Automatic Compiler of Libraries for CINT)** & compiles if the macro has been modified. Instead to force compilation one needs to do:
`.L myTestMacro.C++`

ROOT as a function plotter - exercise

- Simply move the previous code in a simple macro!

ROOT I/O – example - 1

Exercise 0a

ROOT macros typically start from two kind of data formats fed in input:

1) data files in ROOT format (data are organized in TTrees and/or simply histograms)
[we will see examples of this later]

2) text files (data are organized in rows & columns; read or written into sequential rows)

Let us examine an useful example in which we will read an external file (input) and provide an histogram in a ROOT file (output) and also the corresponding plot in a png file :

We start from a macro `CopiaHisto.C` taking as input file `histo.txt` and we execute it:

```
root [0] .x CopiaHisto.C
Info in <TCanvas::Print>: file histo.png has been created
```

… thus creating the output files `file.root` & `histo.png` :

```
-rw-r--r-- 1 pompili cms   316 Oct 18 01:07 histo.txt
-rw-r--r-- 1 pompili cms  1147 Oct 18 01:35 CopiaHisto.C
-rw-r--r-- 1 pompili cms 19911 Oct 18 01:37 histo.png
-rw-r--r-- 1 pompili cms  3943 Oct 18 01:37 file.root
```

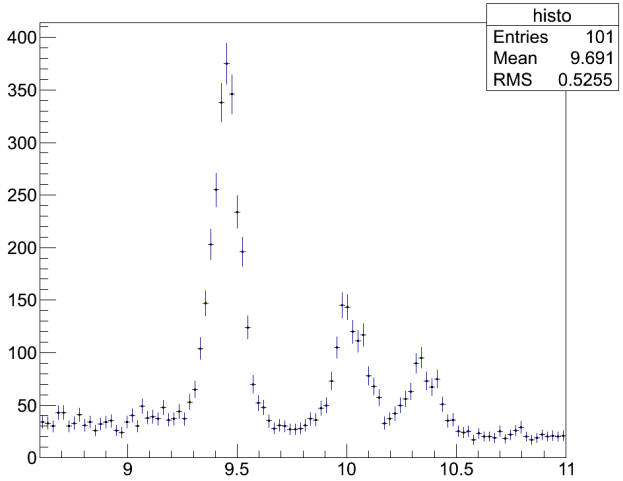
ROOT I/O – example - 2

Exercise 0a

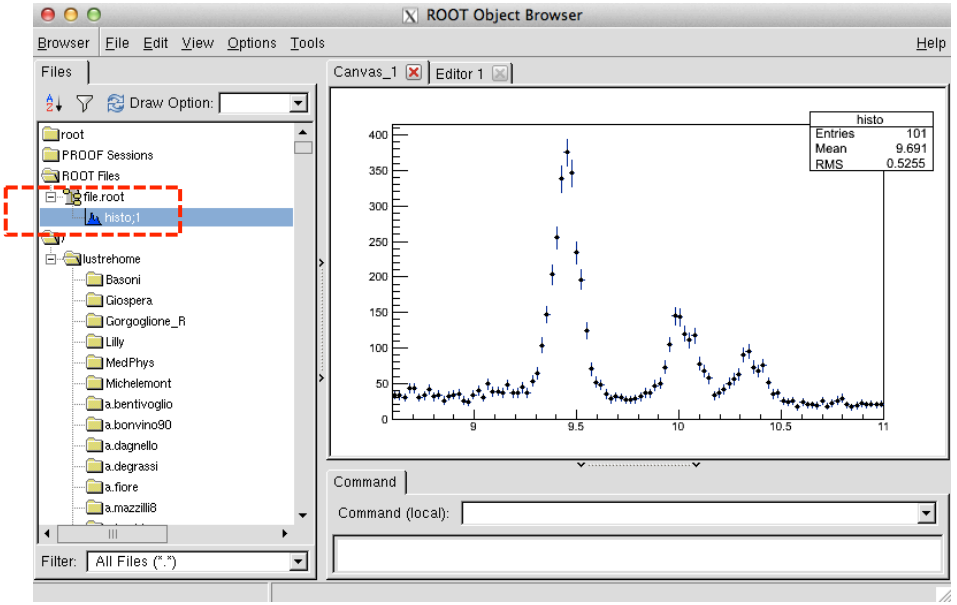
The text input file has the format:

```
34
33
30
43
43
30
33
41
31
34
26
32
34
35
26
24
34
40
30
49
38
39
37
48
36
37
44
37
53
65
104
147
203
255
338
375
346
234
196
124
70
52
48
35
28
31
30
27
27
28
31
37
36
47
50
73
105
145
143
120
111
117
78
68
57
33
37
42
50
56
63
90
95
73
67
75
51
35
36
```

The output graphical file looks like :



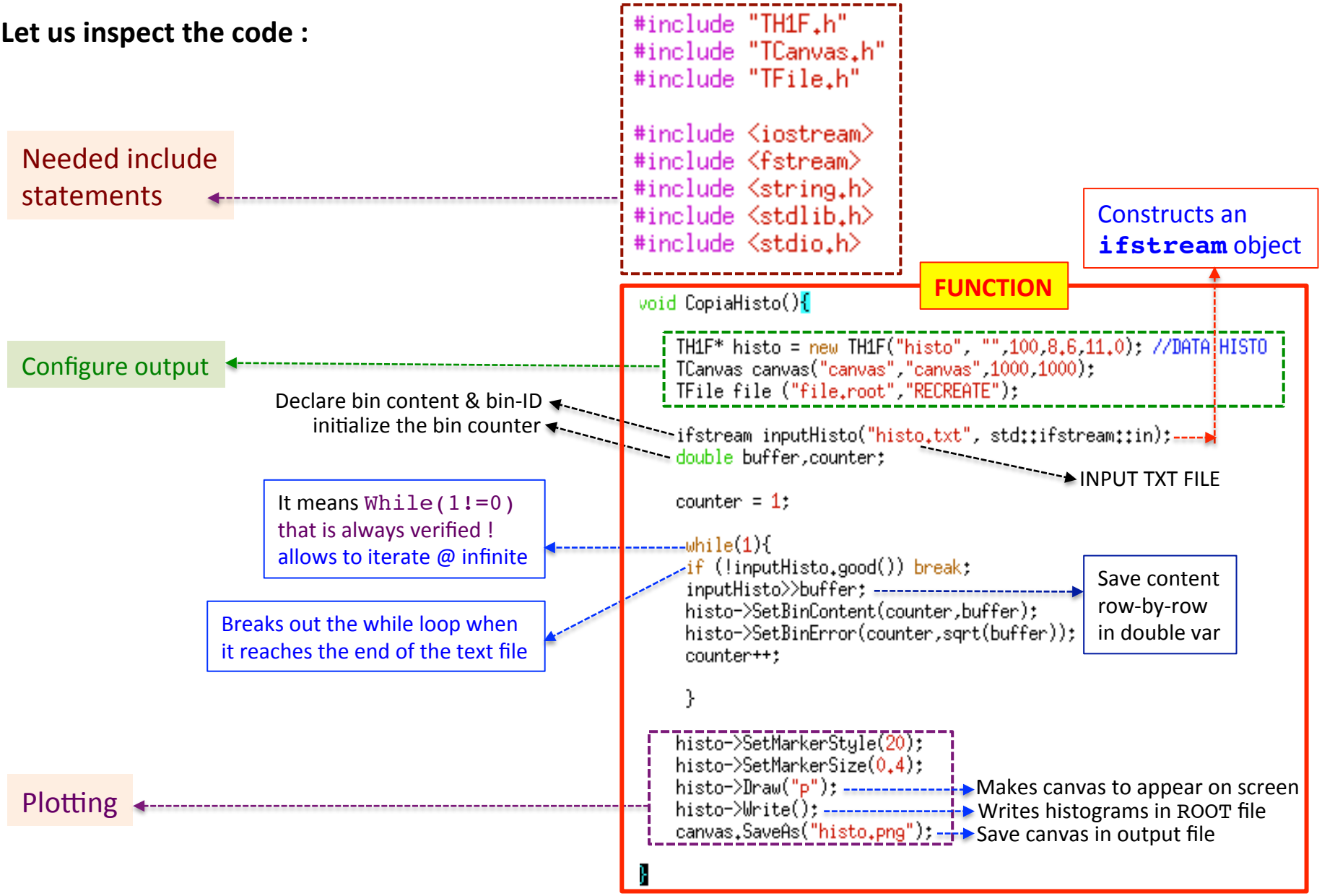
Inspecting the file.root with the **ROOT** Browser :



ROOT I/O – example - 3

Exercise 0a

Let us inspect the code :



ROOT : the most common classes

TH1, TH2 :	**Histograms 1D, 2D**	**(TH1F derived from the base class TH1)**
TProfile :	**Profile histograms**	
TF1, TF2 :	**Functions 1D, 2D**	
TLine :	**Line in plane**	
TMarker :	**Marker**	
TGraph :	**Vector of points in the plane**	
TLatex :	**Text writing as in LaTex**	
TLegend :	**Legend**	
TFile :	**File ROOT**	
TCanvas :	**Grafics/plotting window**	
TTree :	**Data Ntuple**	
TRandom :	**Random numbers**	

https://root.cern.ch/doc/v606/classTH1.html

An example of its usage will be given in the next exercise

ROOT Global Pointers

All ***global pointers*** in **ROOT** begin with a small "**g**"; the most important are:

- **gROOT**: the gROOT variable is the entry point to the ROOT system. Technically it is an instance of the TROOT class. Using the gROOT pointer one has access to basically every object created in a ROOT based program. The TROOT object is essentially a container of several lists pointing to the main ROOT objects.
- **gRandom**: the gRandom variable is a variable that points to a random number generator instance of the type TRandom3. Such a variable is useful to access in every point of a program the same random number generator, in order to achieve a good quality of the random sequence.
- **gStyle**: By default ROOT creates a default style that can be accessed via the gStyle pointer. This class includes functions to set some of the following object attributes.
  - Canvas
  - Pad
  - Histogram axis
  - Lines
  - Fill areas
  - Text
  - Markers
  - Functions
  - Histogram Statistics and Titles
- **gSystem**: An instance of a base class defining a generic interface to the underlying Operating System, in our case TUnixSystem.

TGraph

Exercise 0b

```
#include <TH1.h>
#include <TF1.h>
#include <TF2.h>
//#include <TFormula.h>
#include <TStyle.h>
#include <TCanvas.h>
//#include <TProfile.h>
#include <TGraph.h>
#include <TGraphErrors.h>
#include <TString.h>
#include <TLine.h>
//#include <TPad.h>

// esecuzione:
// .L main.C
// main("png")

void main(TString extens){
  //
  gROOT->Reset();
  gROOT->Clear();
  //
  ////////////////////////////////////// configuration of graphical style
  gStyle->SetCanvasColor(0);
  gStyle->SetPadColor(0);
  //gStyle->SetHistFillColor(0);
  //gStyle->SetHistLineStyle(1);
  //gStyle->SetHistLineWidth(1);
  //gStyle->SetHistLineColor(1);
  gStyle->SetTitleXOffset(0.9);
  gStyle->SetTitleYOffset(1.15);
  //gStyle->SetOptStat(1110);
  gStyle->SetOptStat(kFALSE);
  gStyle->SetOptFit(0111);
  gStyle->SetStatH(0.1);
  gStyle->SetPadTopMargin(0.09);
  gStyle->SetPadBottomMargin(0.13);
  gStyle->SetPadLeftMargin(0.12);
  gStyle->SetPadRightMargin(0.10);
  gStyle->SetPadTickX(1);  // To get tick marks on the opposite side of the frame
  gStyle->SetPadTickY(1);
  gStyle->SetOptTitle(1);
  gStyle->SetStatFont(42);
  gStyle->SetTitleFont(42);
  gStyle->SetTitleSize(1);
  //
  gROOT->SetStyle("Plain"); // change to default/plain graphical style
  gStyle->SetNdivisions(10);
  gStyle->SetCanvasBorderMode(0);
  gStyle->SetPadBorderMode(0);
  //
  gStyle->SetOptTitle(1);
  gStyle->SetStatFont(42);
  gStyle->SetTitleFont(42);
  gStyle->SetTitleSize(1);
  gStyle->SetPadColor(0);
  //
  ////////////////////////////////////////////////////////////////////////
```

TGraph

Exercise 0b

```
.....................................................................
//
TCanvas *MyC = new TCanvas("MyC","A simple Graph with errors bars",900,700);
MyC->SetBorderSize(2);
MyC->SetFrameFillColor(0);
MyC->SetGridx(1);
MyC->SetGridy(0);
MyC->cd();
//
// vector of maximum Temperatures in July 2017
Float_t myMaxTemp[31] = {26.,28.,31.,29.,27.,34.,33.,31.,32.,29.,28.,26.,27.,27.,30.,31.,30.,32.,33.,34.,33.,31.,31.,30.,32.,29.,28.,29.,27., 28.,26.};
//
Float_t yErr[31] = {1.,1.,1.,1.,1.,1.,1.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.,2.};
//
// vector of days in July 2017
Float_t x[31] = {1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.,12.,13.,14.,15.,16.,17.,18.,19.,20.,21.,22.,23.,24.,25.,26.,27.,28.,29.,30.,31.};
//
Float_t xErr[31] = {0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5};
//
/////////////////////////////////////////////////////////////////////////
//
TGraphErrors *myTempGraph = new TGraphErrors(31, x, myMaxTemp, xErr, yErr);
//
myTempGraph->SetTitle("Maximum Temperatures per day in Bari in July 2017 ");
myTempGraph->SetMarkerColor(4);
myTempGraph->SetMarkerStyle(21);
myTempGraph->SetMaximum(40.);
myTempGraph->SetMinimum(10.);
myTempGraph->Draw("AP");    //ALP
//
//////////////
//
//MyC->Update();
MyC->SaveAs("./Plots/myMaxTempJuly2017."+extens);
//MyC->Clear();
// //
gROOT->Reset();
gROOT->Clear();
//delete MyC;
//
}
```