Some infos about ROOT

What is ROOT?

- ROOT is an object oriented framework for data analysis
  - read data from some source
  - write data (persistent objects)
  - selected data with some criteria
  - produce results as plots, numbers, fits, ...
- Supports "interactive" (C/C++ like, Python) and "compiled" (C++) usage
- Integrates several tools like random number generations, fit methods (Minuit), Neural Network framework
- Developed and supported by High Energy Ph. community
  - homepage with documentation and tutorials: root.cern.ch

ROOT's user interface

- C++ in batch mode

```
root -b -q myMacro.C > myMacro.log
```

- C++ interpreted code with CINT – the C++ interpreter
  - in the command line:

```
root[0]  for (int i=0; i<10; i++) cout<<"hello "<<i<<endl;
```

  - loading a macro:

```
root[1] .L mySmallMacro.C;
root[2]  myFunction(1, 2, 3);
```

- C++ compiled code via CINT

```
root[]  .L myScript.C+
Creating shared library /home/.../MyScript_C.so
```

- Python:
  - Access to ROOT from Python

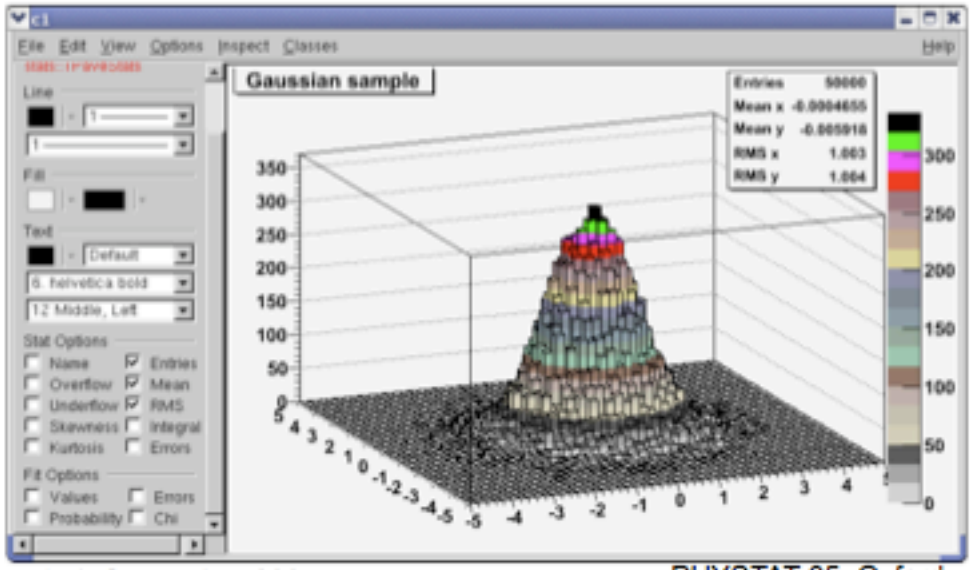
```
>>> from ROOT import TLorentzVector
>>> l = TLorentzVector
```

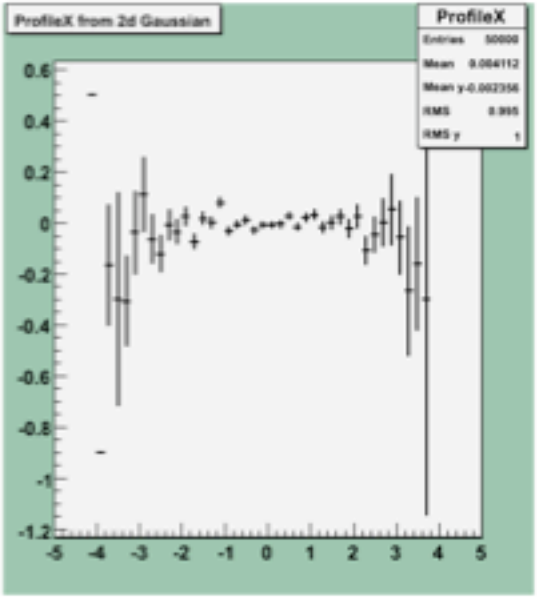
  - Access to Python from ROOT

```
root [0]  TPython::LoadMacro("MyPyClass.py");
root [1]  MyPyClass mpc;
```

Histograms

- 1-2-3 dimensional histograms
  - Errors for each bin can be computed:
    - Default: as sqrt(bin content)
    - As sqrt(sum of squares of weights of the bin)
- 1-2 dimensional profile histograms
  - Mean value of Y and its standard deviation for each bin in X





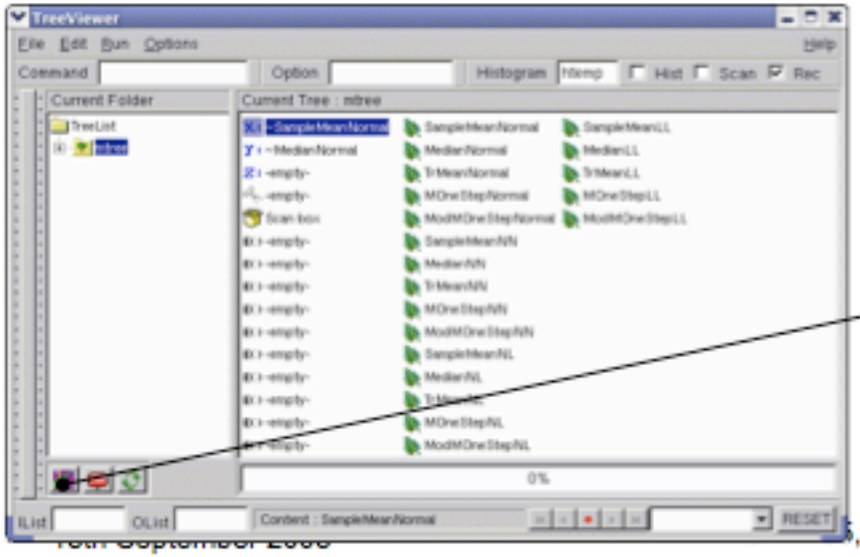
In questa esercitazione lavoreremo sugli istogrammi.
Come documentazione di riferimento trovate: ***ftp://root.cern.ch/root/doc/3Histograms.pdf***

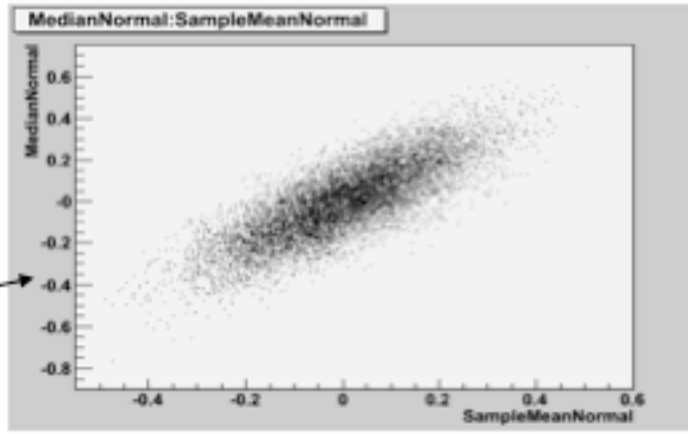
Analysis of TTrees

- TTree::Draw method and TTreeViewer - an easy way to examine the tree:
  - Producing histograms of user-defined expressions in up to 4 dimensions
  - Expressions – C++ formulas
  - Selections – expressions, user-defined macros or graphical cuts

Examples:

```
Tree.Draw("sqrt(x):y", "x>0 && y<1");
Tree.Draw("2*TMath::Log(x)", cut1 || cut2);
```

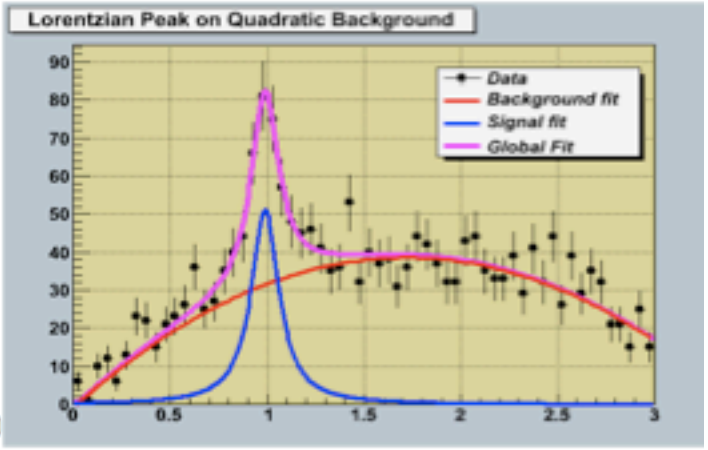




Fitting - interface

- Minimization packages: Minuit and Fumili
- Fitting can be done:
  - Directly in those packages with a user-defined function to minimize
  - Through the general interface of
    - TH1::Fit (binned data) – Chisquare and Loglikelihood methods
    - TGraph::Fit (unbinned data)
    - TGraphErrors::Fit (data with errors)
    - TGraphAsymmErrors::Fit (taking into account asymmetry of errors)
    - TTree::Fit and TTree::UnbinnedFit

- RooFit package for object-oriented data modeling. Distributed with ROOT starting from version 5.02-00



Graphs

- 1-d:
  - TGraph
  - TGraphErrors
  - TGraphAsymmErrors
  - TMultiGraph – a collection of graphs
- 2-d:
  - TGraph2D
  - TGraph2DErrors