# ROOT : A quick introduction

# ROOT in a nutshell

- ROOT is a software toolkit which provides building blocks for
  - Data processing
  - Data analysis
  - Data visualization
  - Data storage
- Simulation (Event Generators, Detectors, Tracking)
- ROOT is written mainly in C++ (C++11 standard)
  - Bindings for Python and other languages provided
- Adopted in High Energy Physics and other sciences (but also industry)
  - ~250 PetaBytes of data in ROOT format on the LHC Computing Grid
  - Fits and parameters' estimations for discoveries (e.g. the Higgs)
  - Thousands of ROOT plots in scientific publications

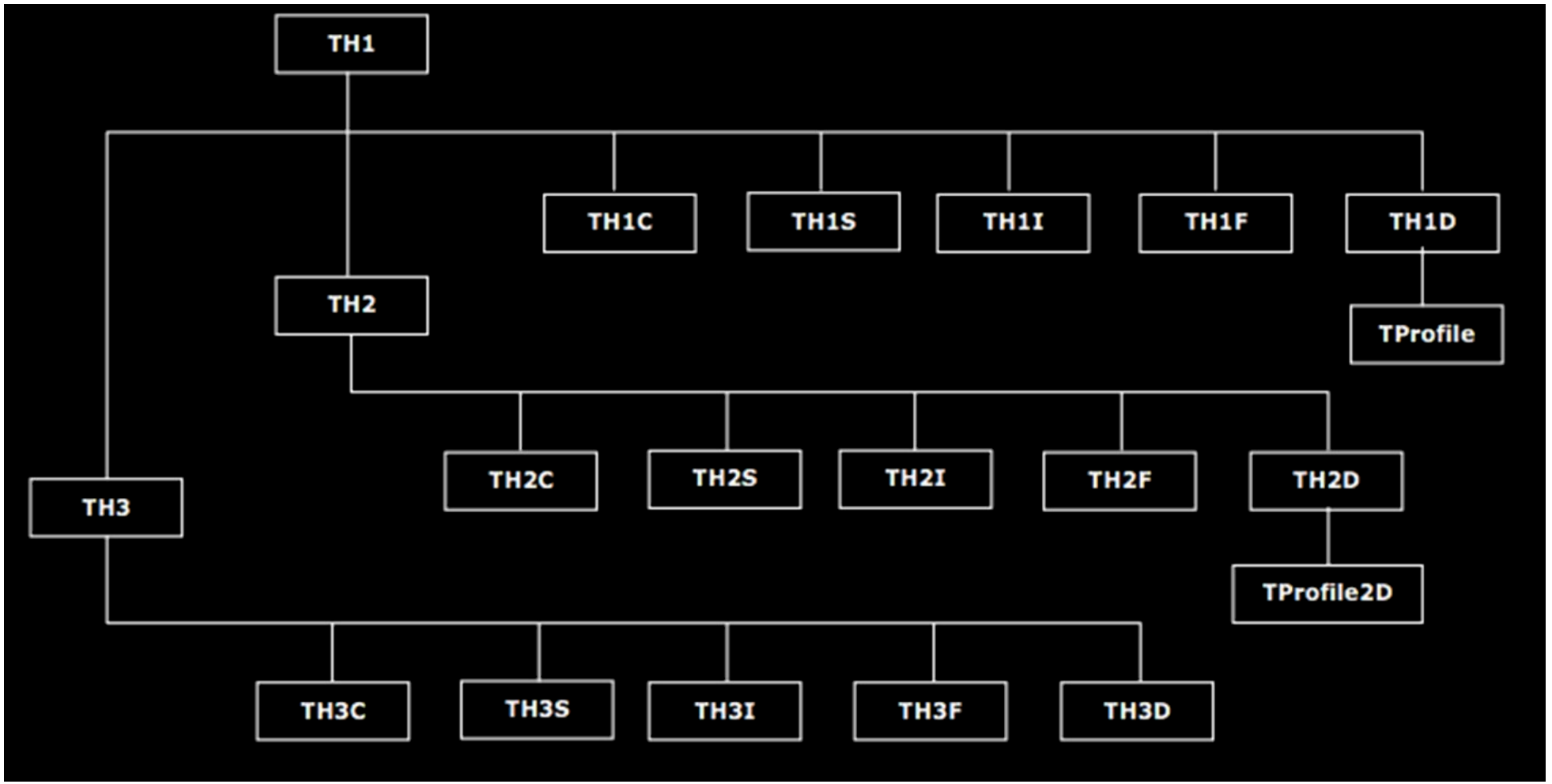
# ROOT in a Nutshell

- ROOT can be imagined as a family of building blocks for a variety of activities, for example:
  - Data analysis: histograms, graphs, trees
  - I/O: storage of any C++ object
  - Statistical tools (RooFit/RooStats): rich modeling and statistical inference
  - Math: non trivial functions (e.g. Erf, Bessel), optimized math functions (VDT)
  - C++ interpretation: fully C++11 compliant
  - Multivariate Analysis (TMVA): e.g. Boosted decision trees, neural networks
  - And more: HTTP severing, JavaScript visualization, advanced graphics (2D, 3D,event display)
  - PROOF: parallel analysis facility

# Persistency (I/O)

- ROOT offers the possibility to write C++ objects into files
  - Exceptional: impossible with C++ alone!
  - Used for petabytes/year rates of LHC detectors
  - 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  for all ROOT objects: TObject::Write

# Histograms in 1D, 2D and 3D



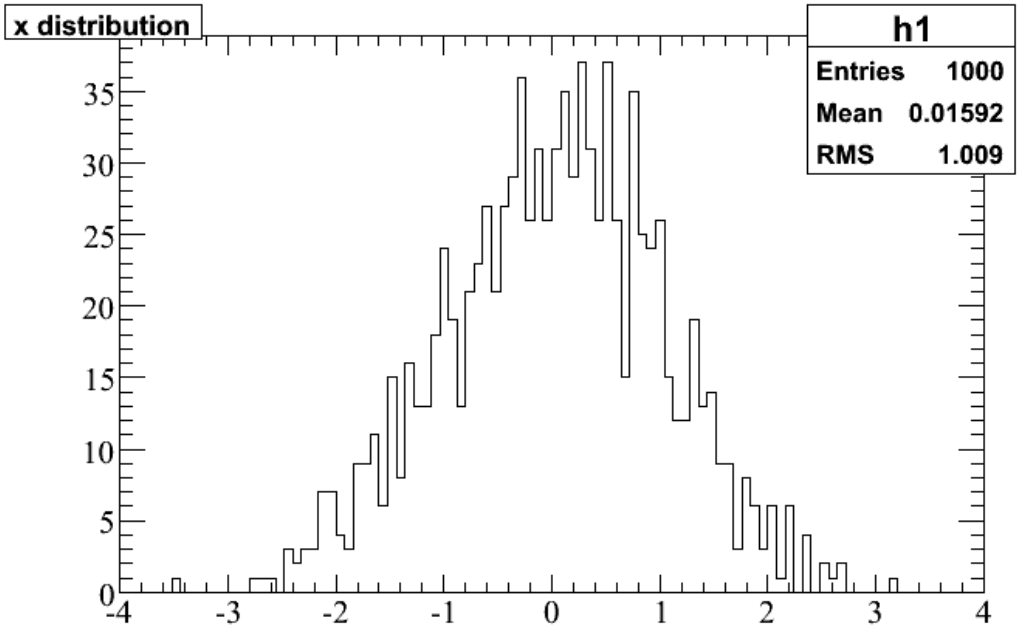

TH?F (Floats) : Max bin content – 7 Digits
TH?D (Double) : Max bin content – 14 Digits

# 1-Dimensional Histograms : TH1I,TH1F,TH1D

- `TH1F* name = new TH1F("name","Title", Bins, lowest bin, highest bin);`

- **Example**:

```
TH1F* h1 = new TH1F("h1","x distribution",100,-4,4);
Float_t x = 0;
for(Int_t i=0; i<1000; i++){
     x=gRandom->Gaus(0,1);
     h1->Fill(x);
}
h1->Draw();
```




The complete list of Methods can be found in:
http://root.cern.ch/root/html/TH1F.html

# Histograms

```
// using different constructors


TH1I* h1 = new TH1I("h1", "h1 title", 100, 0.0, 4.0);
TH2F* h2 = new TH2F("h2", "h2 title", 40, 0.0, 2.0,30, -1.5, 3.5);
TH3D* h3 = new TH3D("h3", "h3 title", 80, 0.0, 1.0,100, -2.0, 2.0,50, 0.0, 3.0);


// cloning a histogram
TH1F* hc = (TH1F*)h1->Clone("title of the cloned histogram");


// projecting histograms
// the projections always contain double values !


TH1D* hx = h2->ProjectionX(); // ! TH1D, not TH1F
TH1D* hy = h2->ProjectionY(); // ! TH1D, not TH1F
```

# Histograms : A complex example

```
// histograms filled and drawn in a loop
void hsum() {

TCanvas *c1 = new TCanvas("c1","The HSUM example",200,10,600,400);
c1->SetGrid();


// Create some histograms.
TH1F *total  = new TH1F("total","This is the total distribution",100,4,4);
TH1F *main   = new TH1F("main","Main contributor",100,-4,4);
TH1F *s1     = new TH1F("s1","This is the first signal",100,-4,4);
TH1F *s2     = new TH1F("s2","This is the second signal",100,-4,4);

total->Sumw2();  // store the sum of squares of weights
total->SetMarkerStyle(21);
total->SetMarkerSize(0.7);
main->SetFillColor(16);
s1->SetFillColor(42);
s2->SetFillColor(46);
```

Macro: hsum.C

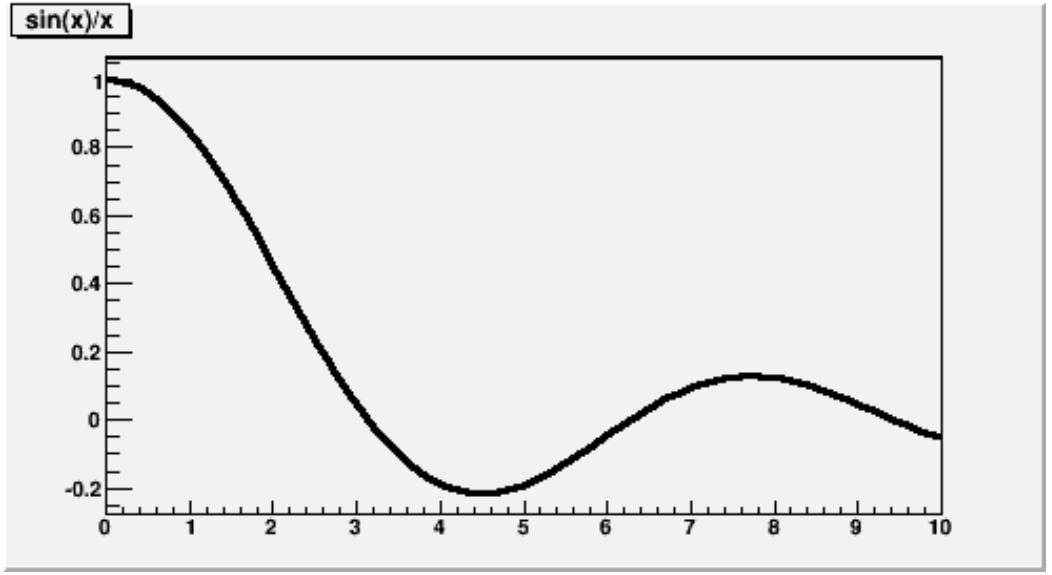
# Histograms : A complex example

```
// Fill histograms randomly

    gRandom->SetSeed();

    Float_t xs1, xs2, xmain;
    for (Int_t i=0; i<10000; i++) {
        xmain = gRandom->Gaus(-1,1.5);
        xs1   = gRandom->Gaus(-0.5,0.5);
        xs2   = gRandom->Landau(1,0.15);
        main->Fill(xmain);
        s1->Fill(xs1,0.3);
        s2->Fill(xs2,0.2);
        total->Fill(xmain);
        total->Fill(xs1,0.3);
        total->Fill(xs2,0.2);
     }
  }
  total->Draw("sameaxis"); // to redraw axis hidden by the fill area
```

Macro: hSum.C

# Formulas and Functions

```
void formula2(){
    TCanvas *c1 = new TCanvas("c1","Example with Formula",500,500);
    // Create a one dimensional function and draw it //
    TF1 * fun1 = new TF1("fun1","sin(x)/x",0,10);
    c1->cd(1);
    fun1->Draw();
    cout << " Deriv " << fun1->Derivative(2.) << endl;
    cout << " Integral " << fun1->Integral(0.,3.) << endl;
    cout << " Func val " << fun1->Eval(1.2456789) << endl;
  }
```




Macro: formula2.C


10

# Creating a TF1 with Parameters

The second way to construct a **TF1** is to add parameters to the expression. Here we use two parameters:

```
root[] TF1 *f1 = new TF1("f1","[0]*x*sin([1]*x)",-3,3);
```

The parameter index is enclosed in square brackets. To set the initial parameters explicitly you can use:

```
root[] f1->SetParameter(0,10);
```

This sets parameter 0 to 10.
You can also use SetParameters to set multiple parameters at once.

```
root[] f1->SetParameters(10,5);
```

This sets parameter 0 to 10 and parameter 1 to 5.
We can now draw the **TF1**:

```
root[] f1->Draw();
```

# Creating a TF1 with a User Function

```
// define a function with 3 parameters
Double_t fitf(Double_t *x,Double_t *par){
   Double_t arg = 0;
   if (par[2] != 0) arg = (x[0] - par[1])/par[2];
   Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
   return fitval;
}


void userfunctexample() {
   // Create a TF1 object using the function defined above. The last
   //  parameter specify the number of parameters for the function.
   TF1 * func = new TF1("fit",fitf,-3,3,3);
   // set the parameters to the mean and RMS of the histogram
   func->SetParameters(500,0.,0.5);
   // give the parameters meaningful names
   func->SetParNames ("Constant","Mean_value","Sigma");
   // call TH1::Fit with the name of the TF1 object
   func->Draw();
}
```



Macro: userfunctexample.C

# Write and Read a ROOT file

- **Write (in a macro):**

```
TFile *myfile = new TFile("fillrandom.root","RECREATE");
myfile->cd();
form1->Write();
sqroot->Write();
h1f->Write();
myfile->Close();
```

- **Read (in a macro):**

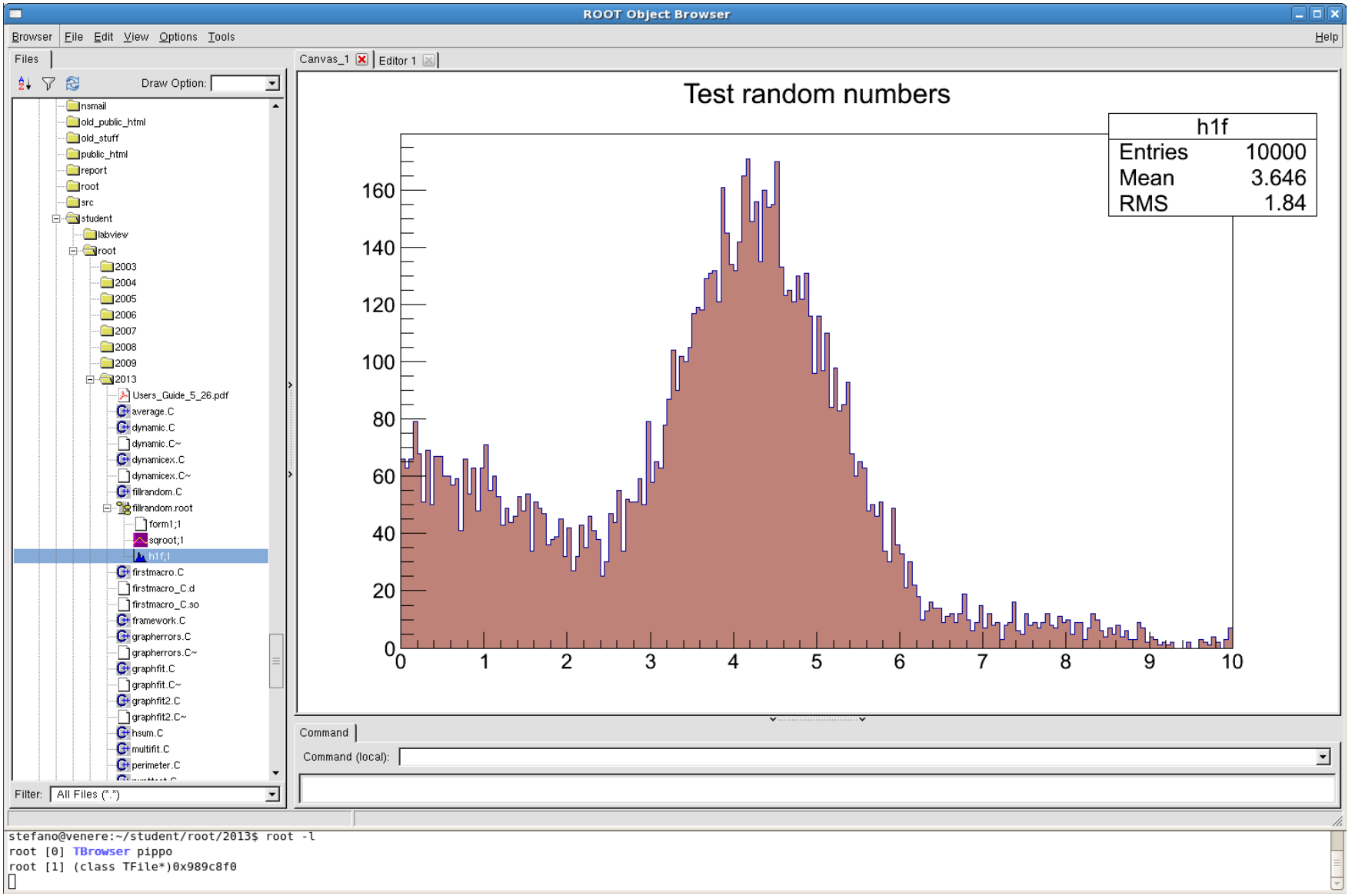
```
TFile *pfile = new TFile("fillrandom.root");
TH1F * ph1 = (TH1F *) pfile->Get("h1f"); // take the object from the TFile
```

- **Read (in the terminal):**

```
root [0] TFile *pfile = new TFile("fillrandom.root");
root [1] pfile->ls(); // it is useful only to know what is stored inside your
TFile if you have no idea of how it has been created.
root [2] TH1F * ph1 = (TH1F *) pfile->Get("h1f"); // take the object from the
TFile
```

# TBrowser : A GUI to analyze the TFile

```
root [0] new TBrowser()
```



# Passing arguments to a macro

```
// define a function with 3 parameters
Double_t fitf(Double_t *x,Double_t *par){
   Double_t arg = 0;
   if (par[2] != 0) arg = (x[0] - par[1])/par[2];
   Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
   return fitval;
}

void userfunctexample(Int_t min = -3, Int_t max = 3) {
   // Create a TF1 object using the function defined above. The last
   //  parameter specify the number of parameters for the function.
   TF1 * func = new TF1("fit",fitf,min,max,3);
   // set the parameters to the mean and RMS of the histogram
   func->SetParameters(500,0.,0.5);
   // give the parameters meaningful names
   func->SetParNames ("Constant","Mean_value","Sigma");
   // call TH1::Fit with the name of the TF1 object
   func->Draw();
   TString nameout = "Funz.";
   nameout+=min;
   nameout+=".";
   nameout+=".root";
   TFile *fo = new Tfile(nameout,"RECREATE");
   fo->cd();
   func->Write();
   fo->Close();
}
```

# Documentation

- https://wwwusers.ts.infn.it/~lea/lacd2016.html
- root.cern.ch
- In any case

# Documentation

- https://wwwusers.ts.in
- root.cern.ch
- In any case

# Esercitazione 10

Esercizio 1

- Fill a histogram randomly (n=10,000) with a Landau distribution with a most probable value at 20 and a “width” of 5 (use the ROOT website to find out about the Landau function)
- Fill the same histogram randomly (n=5,000) with a Gaussian distribution centered at 5 with a “width” of 3
- Write a compiled script with a fit function that describes the total histogram nicely (it might be a good idea to fit both peaks individually first and use the fit parameters for a combined fit)
- Add titles to x- and y-axis
- Include a legend of the histogram with number of entries, mean, and RMS values
- Add text to the canvas with the fitted function parameters
- Draw everything on a square-size canvas (histogram in blue, fit in red)
- Save as png, eps, and root file

# Esercitazione 10