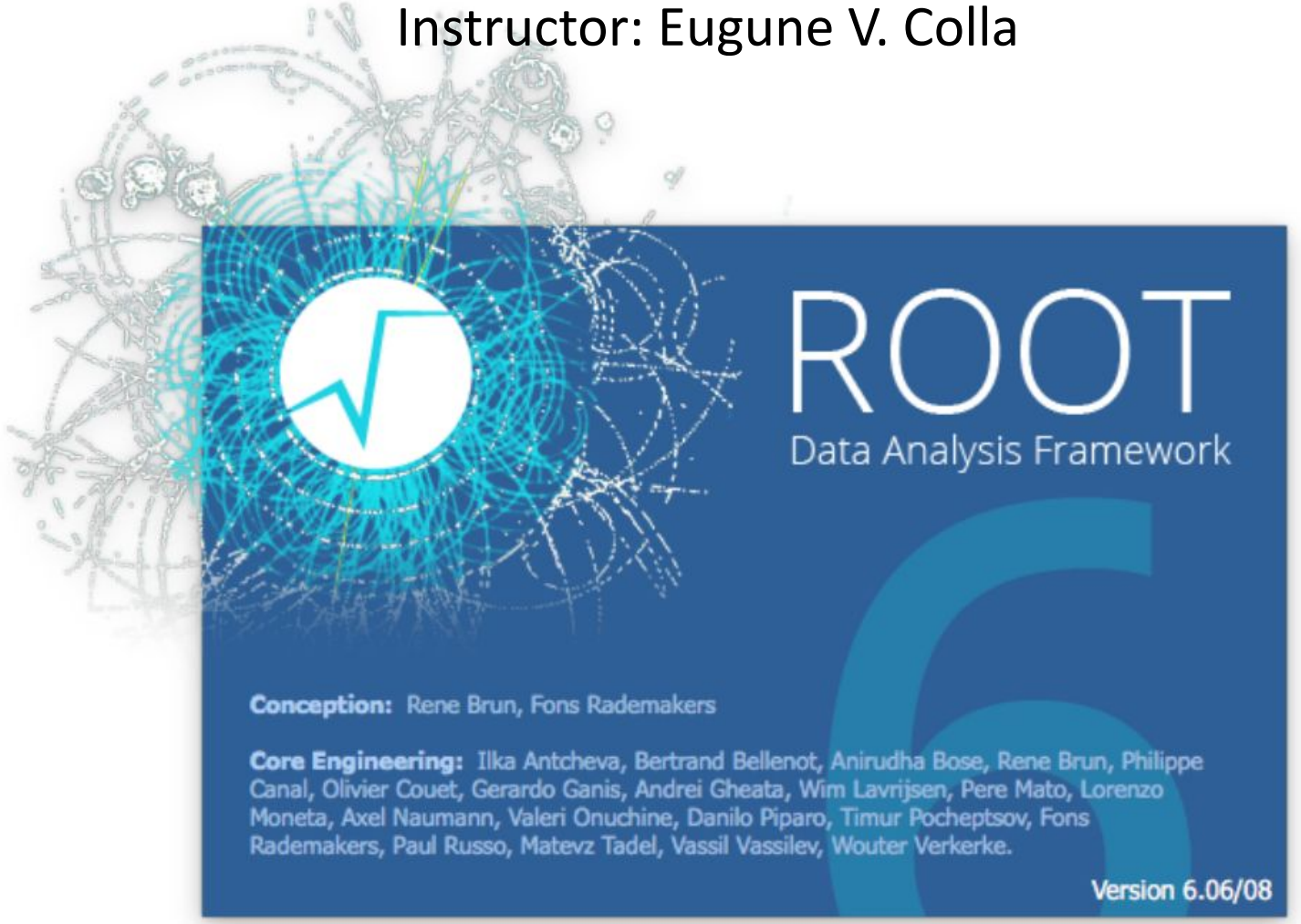
Introduction to ROOT

PHYS 403 2021

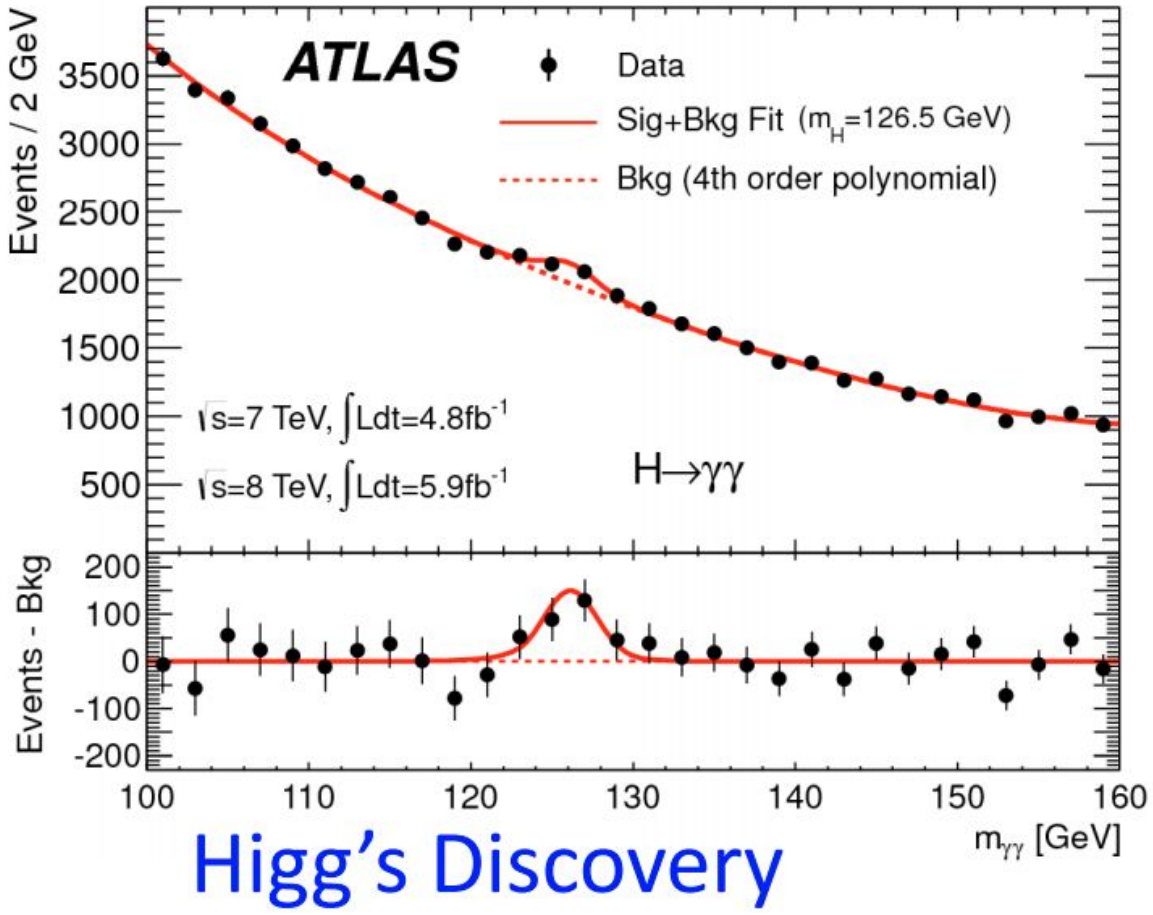
Instructor: Eugune V. Colla



ROOT@CERN: A Data Processing Framework

arxiv:1207.7214

- An extensively used computing and graphing tool in nuclear/particle physics.
- Developed at CERN, very useful for large scale data analysis (PetaByte!!).
- Features event-based tree structure data, fitting and bayesian unfolding tools, ML frameworks, and more!

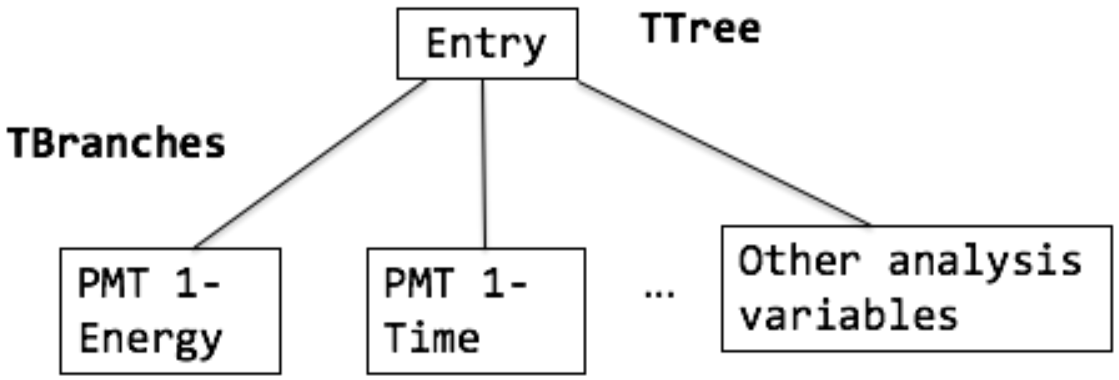


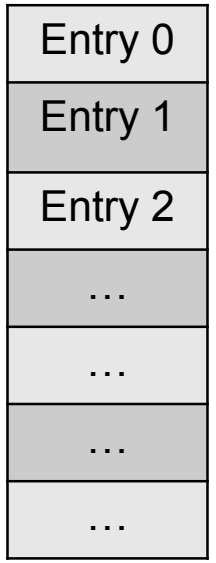
Higg's Discovery

Using ROOT

- https://root.cern/
- Tutorial for beginners: ROOT Primer https://root.cern/primer/
- **C++ based**
- **Can run macros, interactively with terminal and GUI**
- **Today: tree data-structure, histogram and fitting in ROOT**

Data Structure (Tree)





- TTree is one of the most commonly used structures in ROOT.
- One can store variables, arrays and any other C++ datatype in the tree ‘branches’
- Usually we “loop” over a TTree to obtain relevant information from each entry and make **plots**

Interactive ROOT with Terminal Commands

- Install ROOT (lab computers have ROOT installed)
- Open up a terminal window
- Make sure bin/ is added to PATH and lib/ is added to LD_LIBRARY_PATH, for sh shell, change to ROOT install folder and do,

```
$ . bin/thisroot.sh
```

- Start ROOT interactively

```
$ root
```

- The ROOT Prompt

```
root [0] 1+1
(int) 2
```

operation (syntax: c++ with ROOT classes)
results

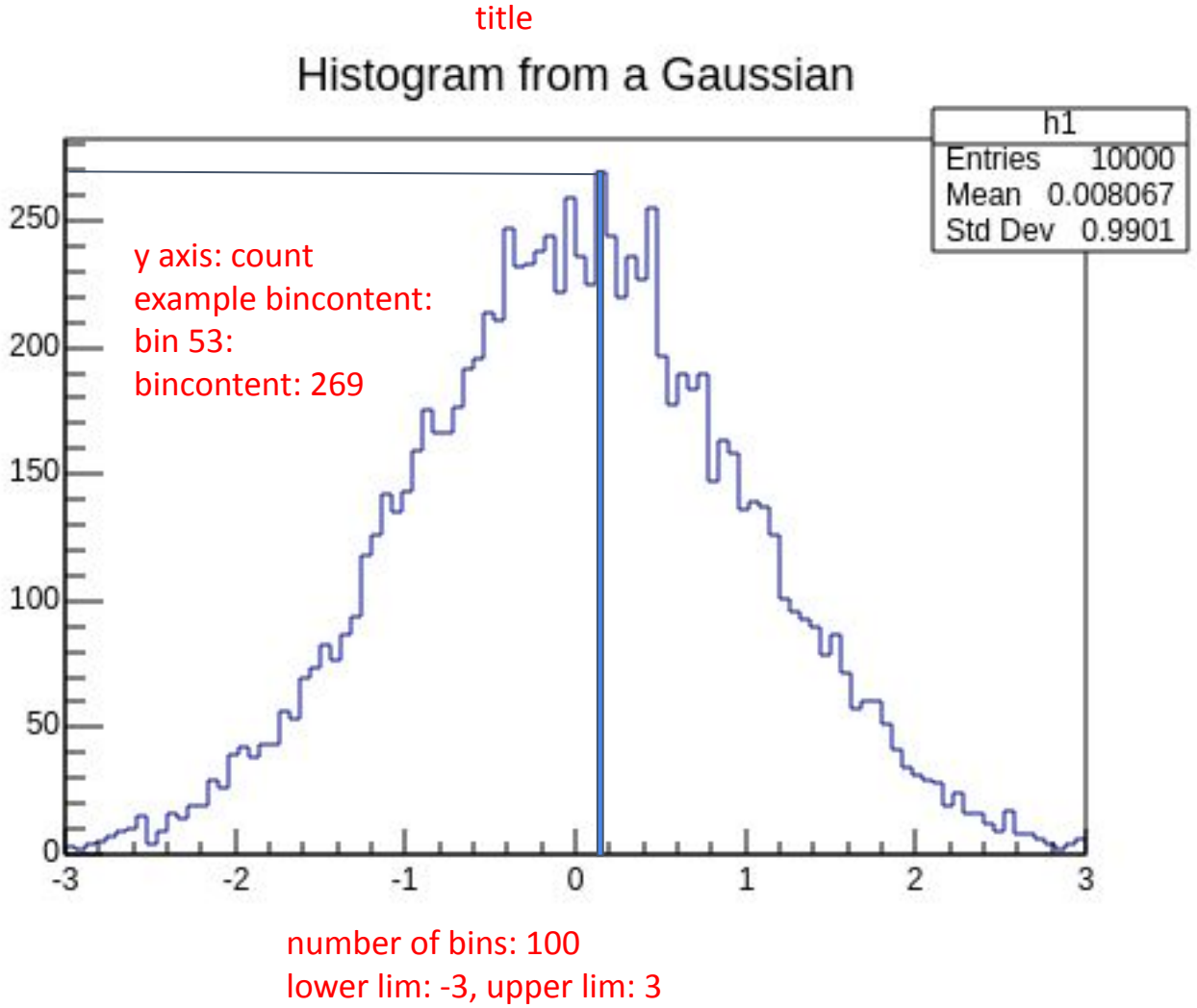
ROOT Macros

```
void trk_jet_ratio(std::string filename, std::string chain_name)
{
    TChain *fChain = new TChain(chain_name.c_str());
    fChain->Add(filename.c_str());
    std::cout << "Chain Entries: " << fChain->GetEntries() << std::endl;
    initBranches(fChain);
    int z_N = 21;
    Float_t zbins[z_N];
    for (int i = 0; i < z_N; i++)
```

- Advantages:
  - quickly rerun everything without the need to re-initialize everything
  - useful for more complicated tasks
- Disadvantages:
  - cannot change parameters in fitting interactively and seeing the results real-time
- Same syntax as interactive running
- save macros as “<function_name>.c”
- for example, save trk_jet_ratio.c (right hand screenshot)
- run in terminal by
- root -l ‘trk_jet_ratio.c(arguments)’

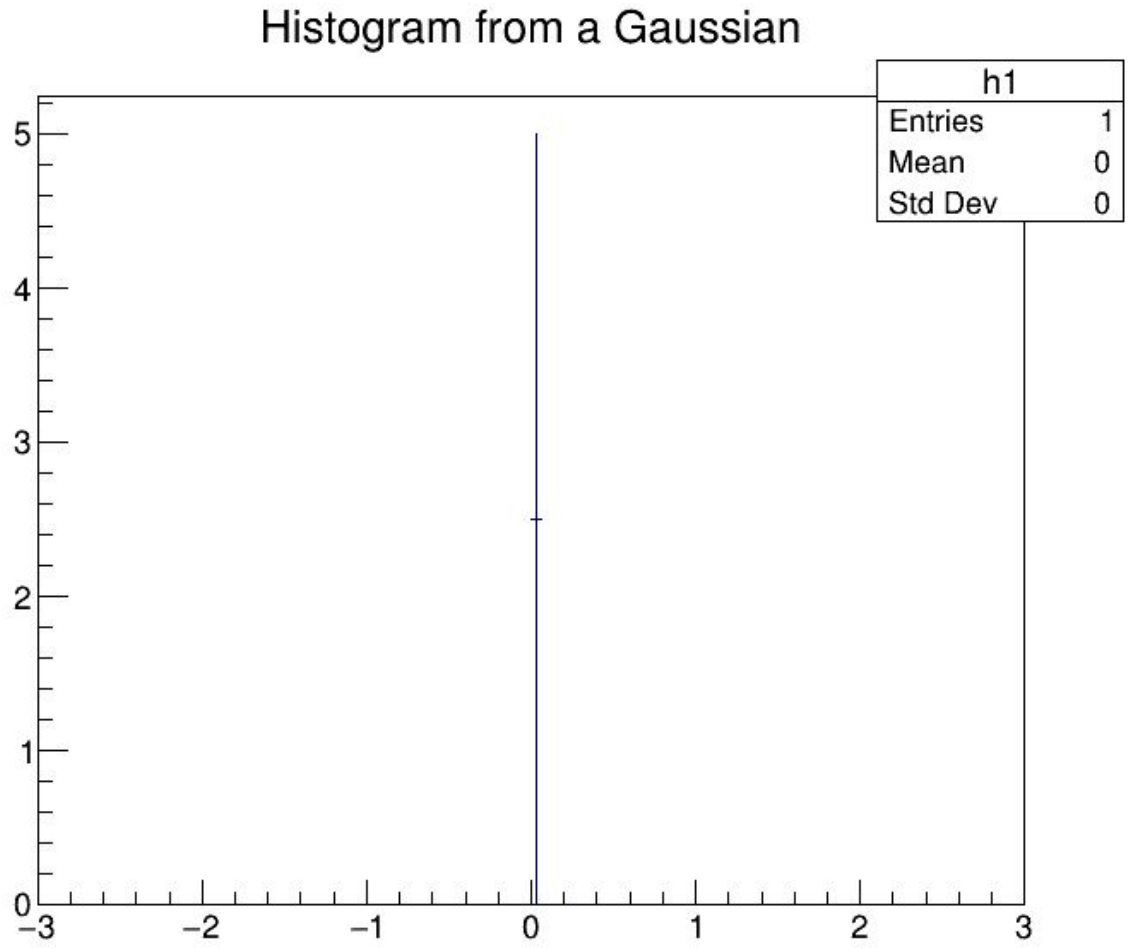
ROOT Histograms

- Histogram classes:
  - TH1, TH2, and TH3 (TH2 and TH3 inherit from TH1): for 1D, 2D, and 3D histograms. https://root.cern.ch/doc/master/classTH1.html
- Definitions:
  - number of bins
  - upper/lower limit
  - weight
  - y-axis: counts; x-axis: values of bins
  - BinContent: sum of weights in a bin
- Tutorial List:
- https://root.cern/doc/master/group__tutorial__hist.html

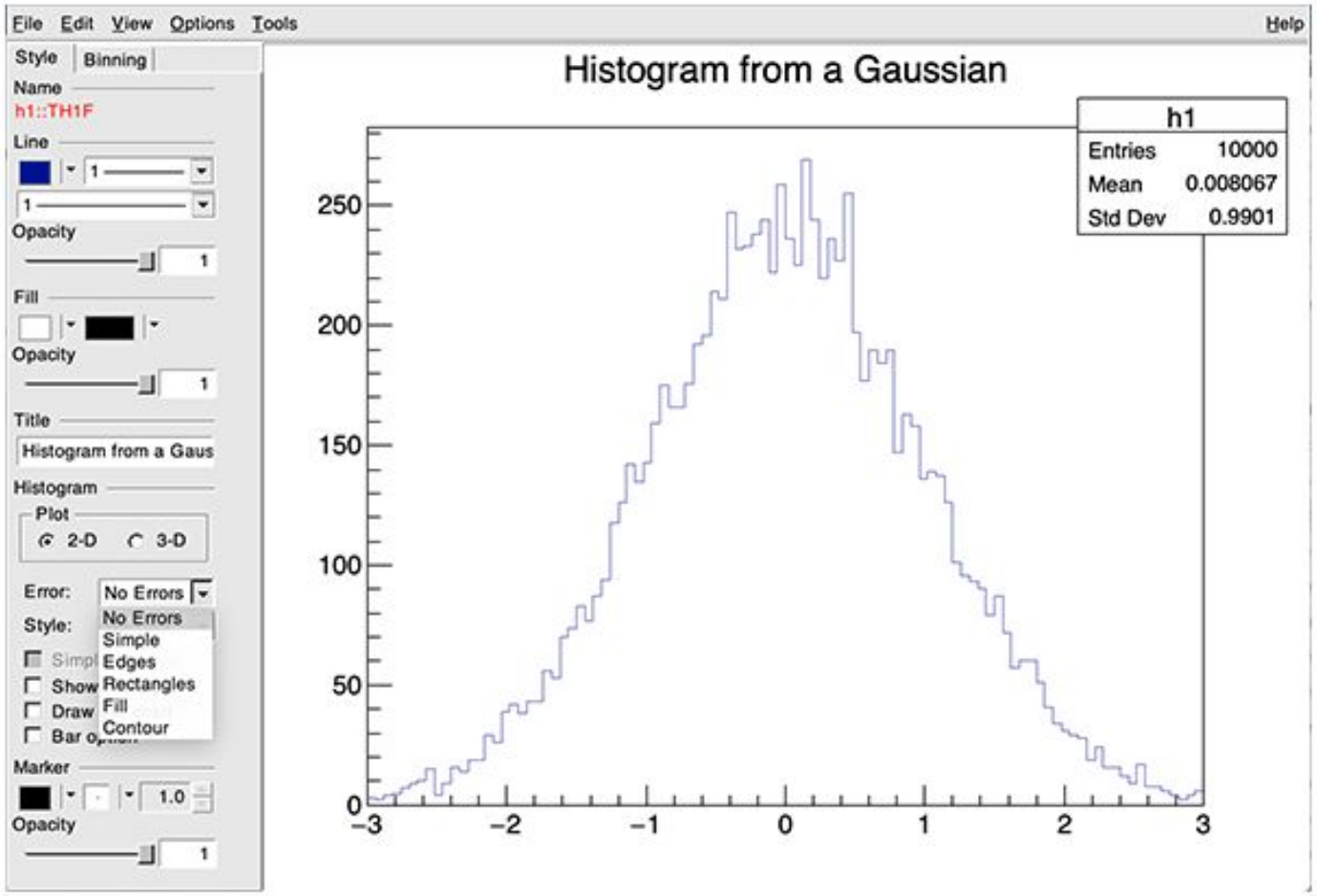


Histogram Syntax

- Using a histogram with type float (TH1F, TH1D uses double):
  - **instantiation:**
  - TH1F* h1 = new TH1F("h1","Histogram from a Gaussian",100,-3,3);
  - **Filling a single observation x with weight w**
  - h1->Fill(x,w);
  - for example: h1->Fill(0., 2.5); (if you put in constant, make sure their type matches with histogram)
  - **Draw histogram**
  - h1->Draw();
  - **Or: directly set bin content with "SetBinContent" (see online document)**



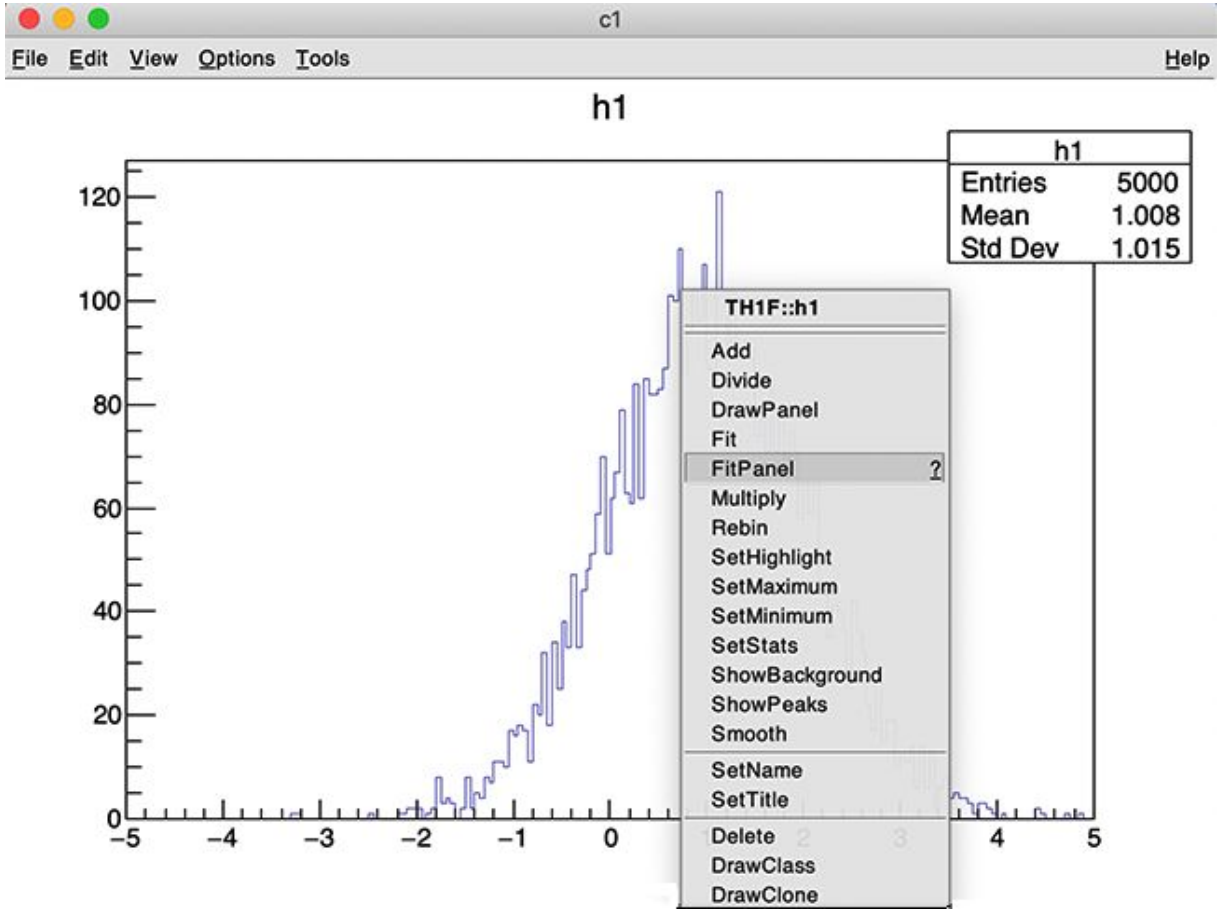
Adjust Histogram Style Interactively



- View (Editor)
- Adjust your style accordingly

Fit a Histogram using Interactive GUI

- Function Class: TF1
- Instantiation:
- TF1 *f1 = new TF1("f1", "[2]*TMath::Gaus(x,[0],[1])");
- Second argument can deal with many functions
- [0],[1],etc. are parameters to be fit
- After drawing your histogram, call out FitPanel



Fit a Histogram using Fit() Method

To fit a function f1 for histogram h1

h1->Fit(f1);

or h1->Fit(f1, “<fit options>”);

See:
https://root.cern.ch/root/htmldoc/guides/users-guide/FittingHistograms.html

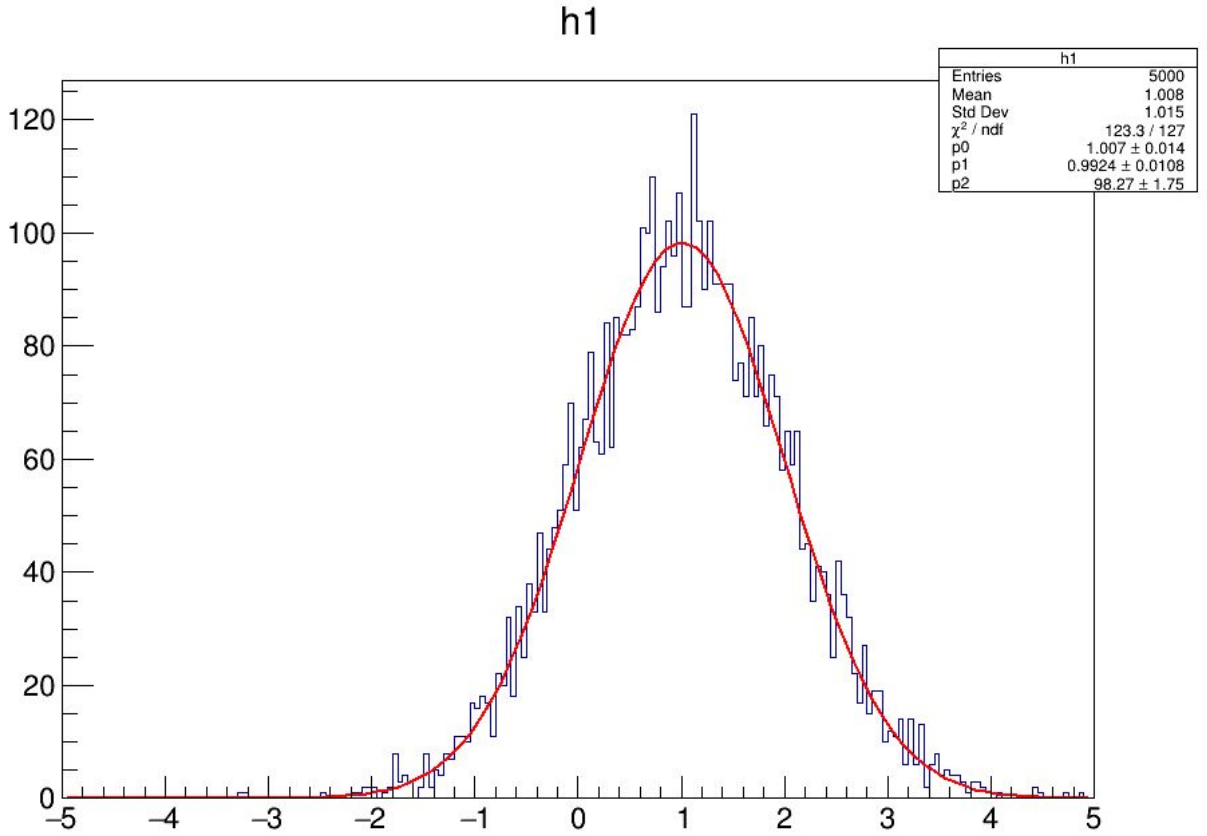
Retrieve Parameters after fitting:

f1->GetParameter(1); for parameter [0]

Or

gStyle->SetOptFit(1)

to **display everything on canvas.**



Where to find additional help (besides TA)

- very detailed tutorial/guides on http://root.cern.ch
- https://root.cern/manual/
- google your question with “root cern”
- root forum:
- https://root-forum.cern.ch/ for others discussion/asking for help