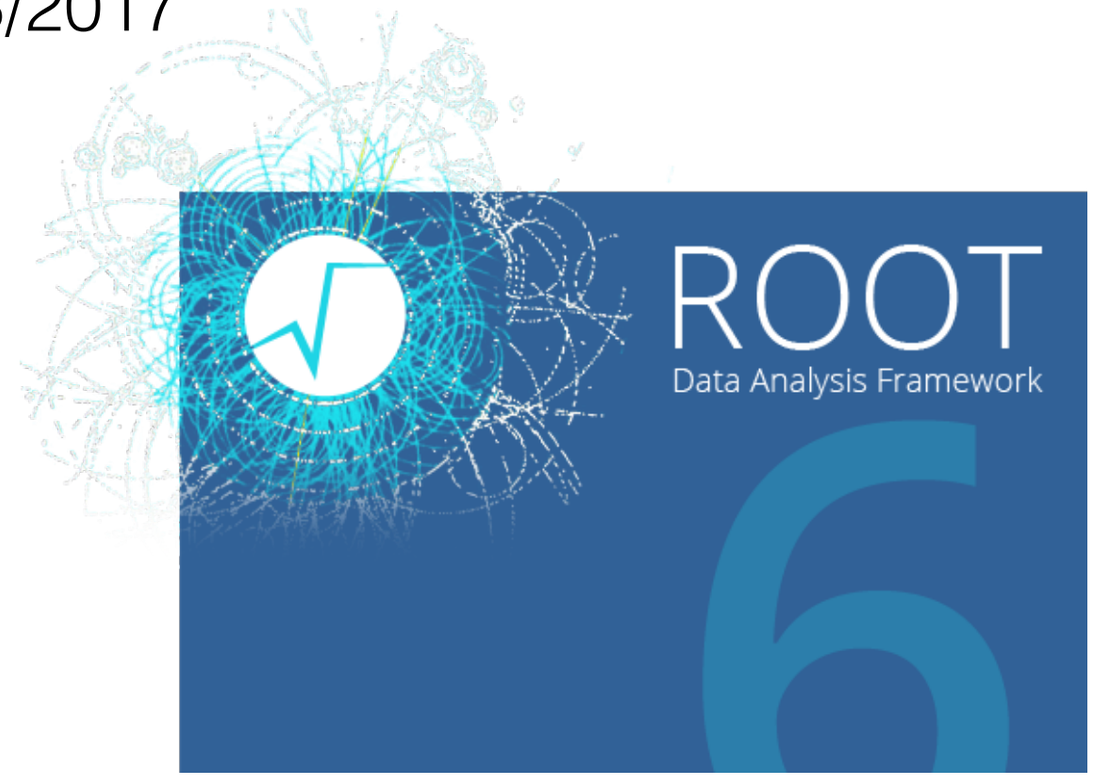
An introduction to the ROOT framework

Computational Physics M.Sc.

13/03/2017



Contents

Why ROOT?

Because ROOT is ...

A modular scientific software framework. It provides all the functionalities needed to deal with **big data processing**, **statistical analysis**, visualisation and storage. It is mainly written in C++ but integrated with other languages such as Python and R (…from the ROOT page).

But also because it is …

An object-oriented, **HEP-related** data analysis framework! This means that custom Physics objects (electrons, muons, jets, MET) may be created, stored and accessed with ease. Finally, Lorentz vectors and boosts, along with all the necessary methods are already defined.

ROOT framework

Q: What is a framework?

A: In computer programming, a software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.

ROOT framework:

+ Less code required from the user for several functionalities
+ Coherent and modular code
+ Assiduous tests from many users ensuring stable releases
+ I/O, utilities and services, graphics
+ Object-oriented: encapsulation and inheritance

- Framework conventions, reduced flexibility
- Learning curve may not be so steep

CINT

CINT is a C/C++ interpreter, with several extensions to C++
(but also with quite a few differences)

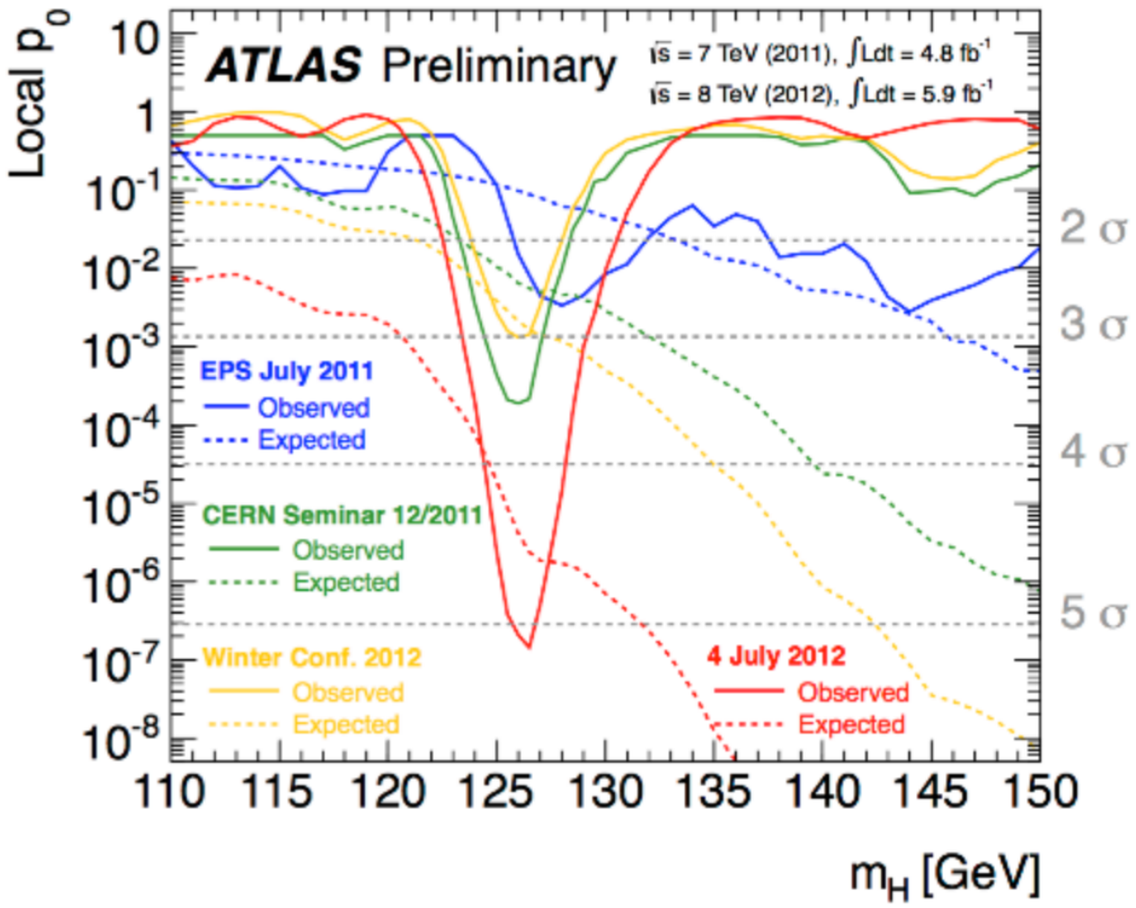
In computer science, an interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program (wikipedia)

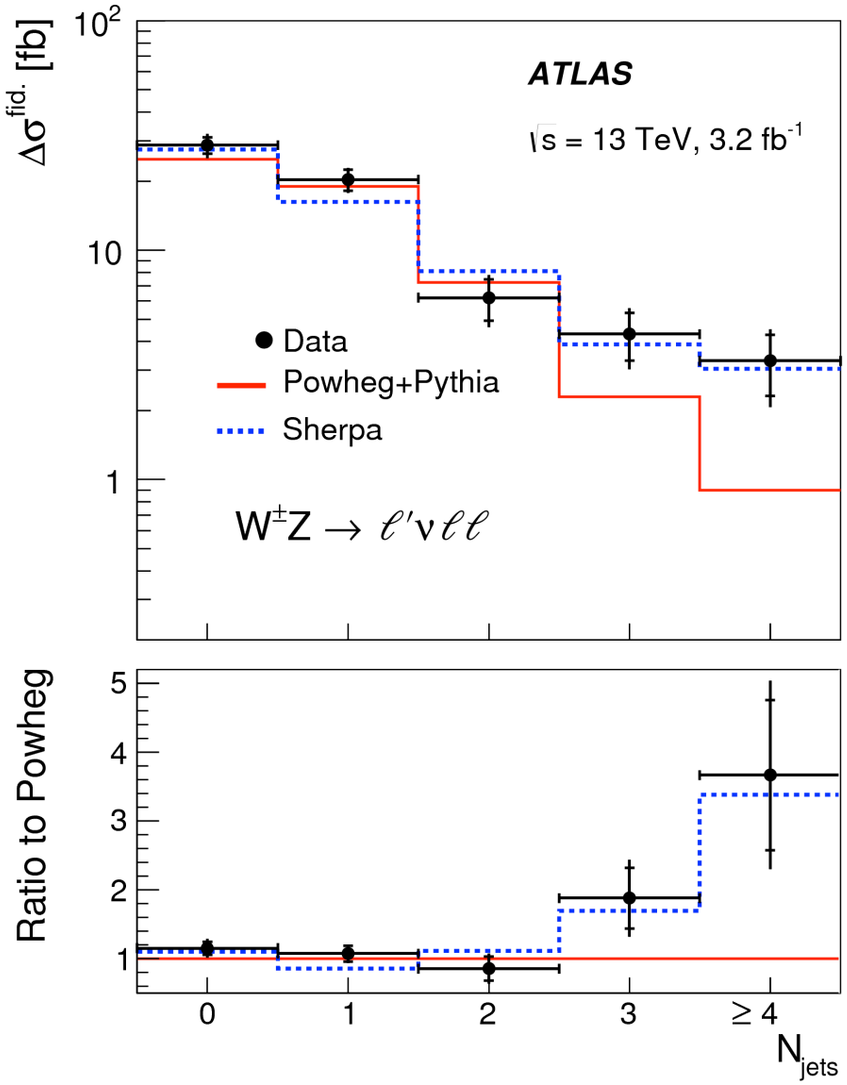
CINT commands always start with a “.”

More on CINT:
https://root.cern.ch/cint
http://www.hanno.jp/gotom/Cint.html

What can ROOT do

- Analyse very large datasets: (Ldt=35/fb, σ_{tot} = ~ 5 10^7 nb)
- Offer implementation of statistical methods to extract confidence intervals, unfolded distributions, ..



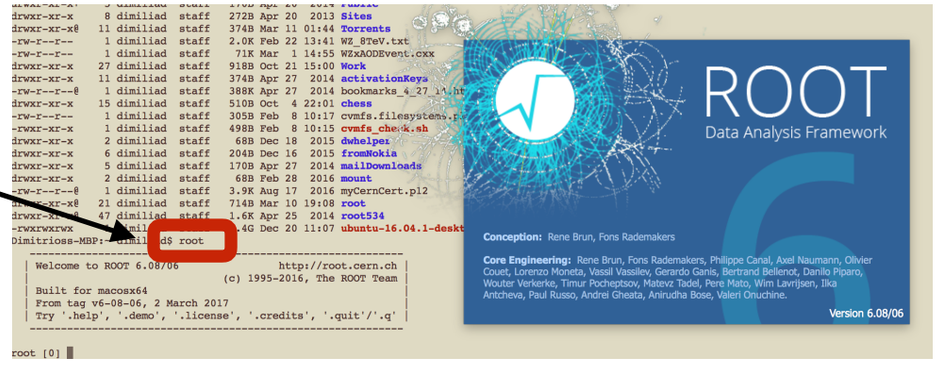


- Save all statistical analysis for future reference and/or comparisons and checks
- Beautify result plot!

ROOT basics

Start ROOT:

```
dimiliad$ root
```



Skip splash screen: `dimiliad$ root -l`

Get help: `root [0] .?`

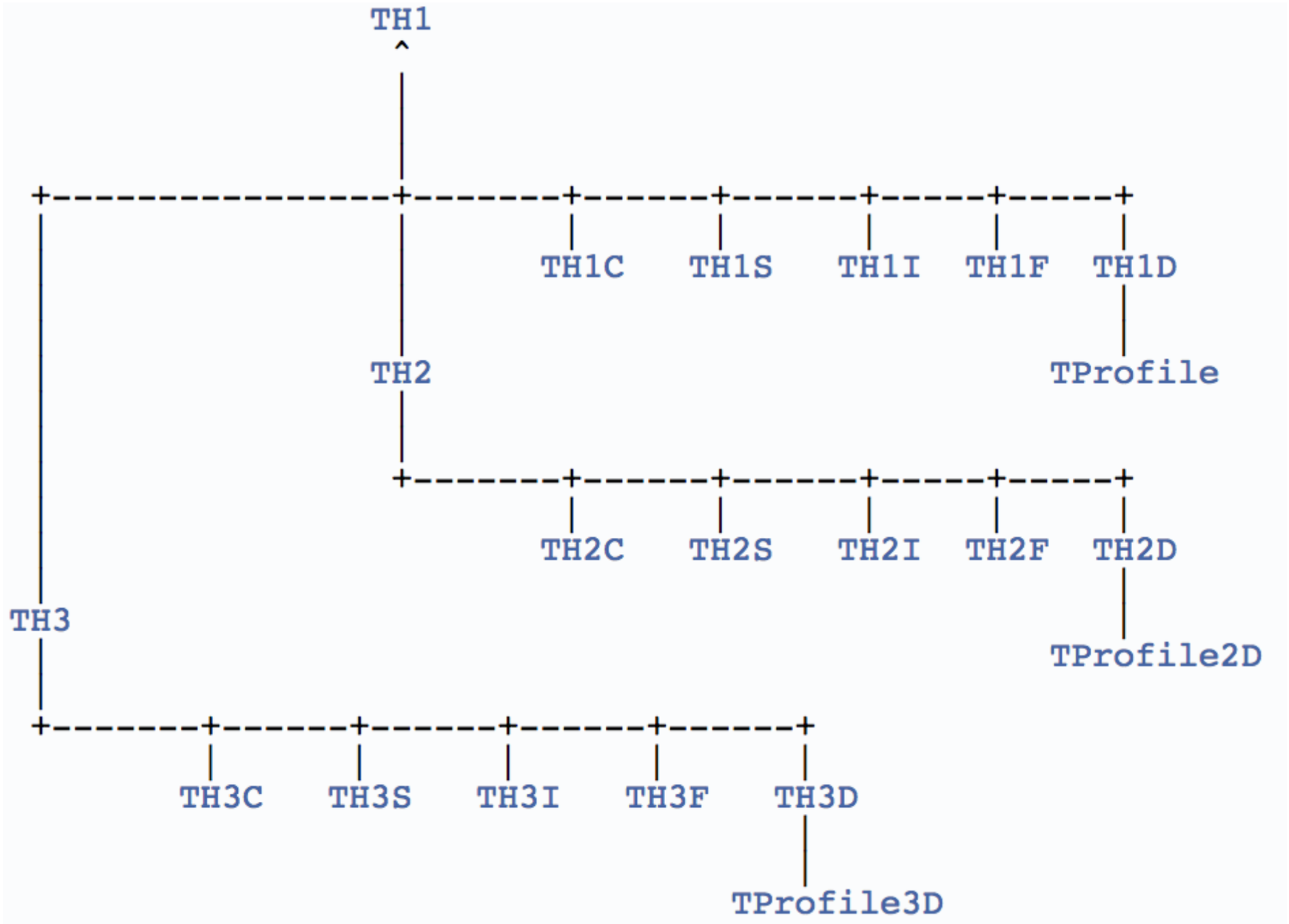
Quit ROOT: `root [0] .q`

Web page: http://root.cern.ch/

Documentation: https://root.cern.ch/documentation

ROOT tutorials: http://root.cern.ch/root/Tutorials.html

Histograms


TH1
TH1C
TH1S
TH1I
TH1F
TH1D
TProfile
TH2
TH2C
TH2S
TH2I
TH2F
TH2D
TProfile2D
TH3
TH3C
TH3S
TH3I
TH3F
TH3D
TProfile3D

Histograms::Create

Declaring a TH1F histogram:

```
root [4] TH1F h1("hist1","myHisto",100,-3.,3.);
```

name, bins, type, title, range

Get all methods for h1 from the ROOT prompt by typing: h1. and then hit TAB

object name

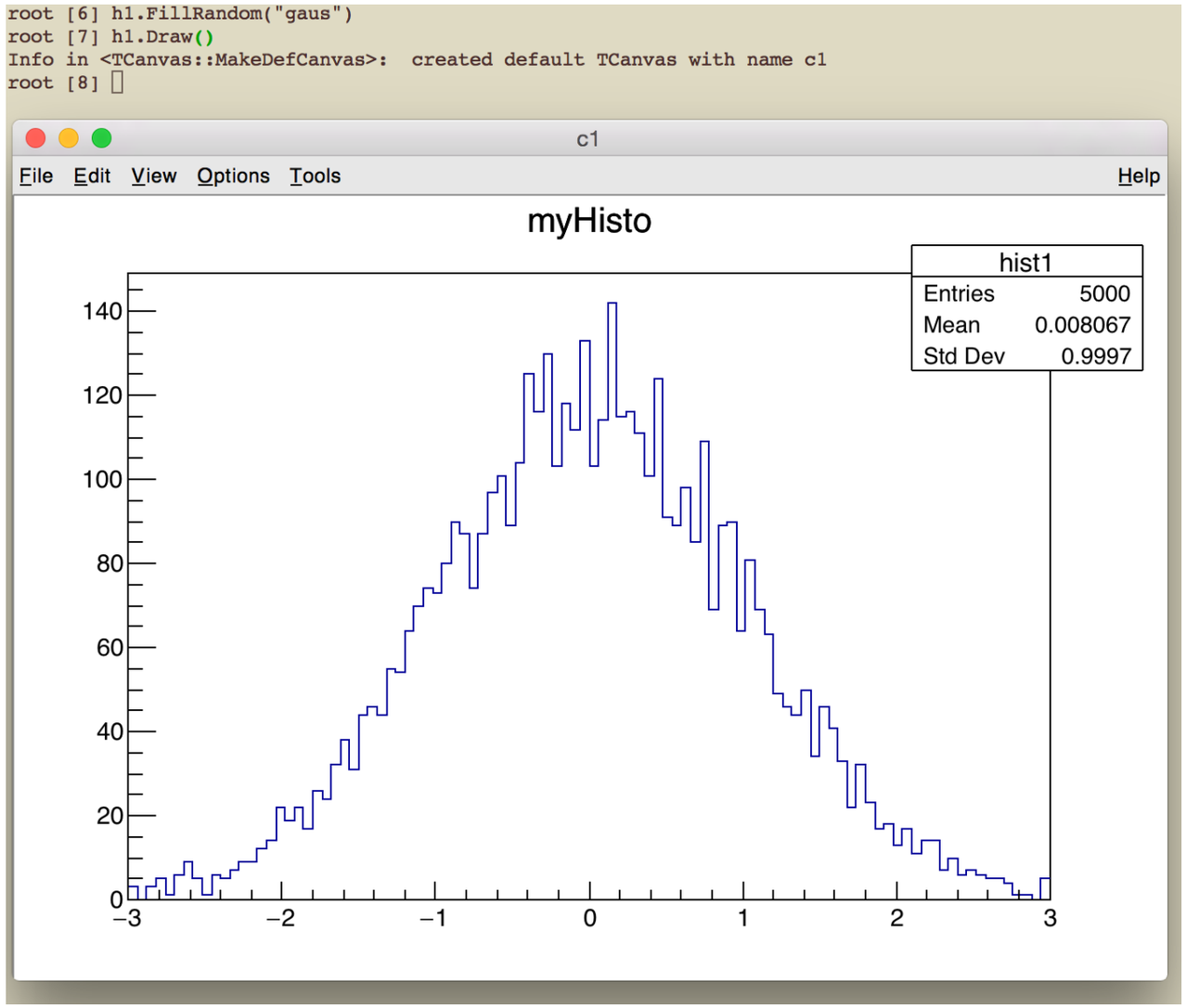
```
root [13] h1.GetBin
GetBin
GetBinCenter
GetBinContent
GetBinError
GetBinErrorLow
GetBinErrorOption
GetBinErrorUp
GetBinLowEdge
GetBinWidth
GetBinWithContent
GetBinXYZ
root [13] h1.GetBinWidth(
Double_t GetBinWidth(Int_t bin) const
root [13] h1.GetBinWidth(0)
(Double_t) 0.0600000
root [14] h1.GetName
ROOT_prompt_14:1:4: error: reference to non-static member function must be called; did you mean to call it with no arguments?
h1.GetName
~~~^~~~~~
          ()
root [15] h1.GetName()
(const char *) "hist1"
root [16] h1.GetTitle()
(const char *) "myHisto"
root [17]
```

I hit TAB here!

GetName is a function; ROOT complains!

Histograms::Filling random

Fill it with random numbers taken from a Gaussian:



Statistics box

Add/remove information in the stats box by using:

gStyle>SetOptStat(xxxxxxxxx)

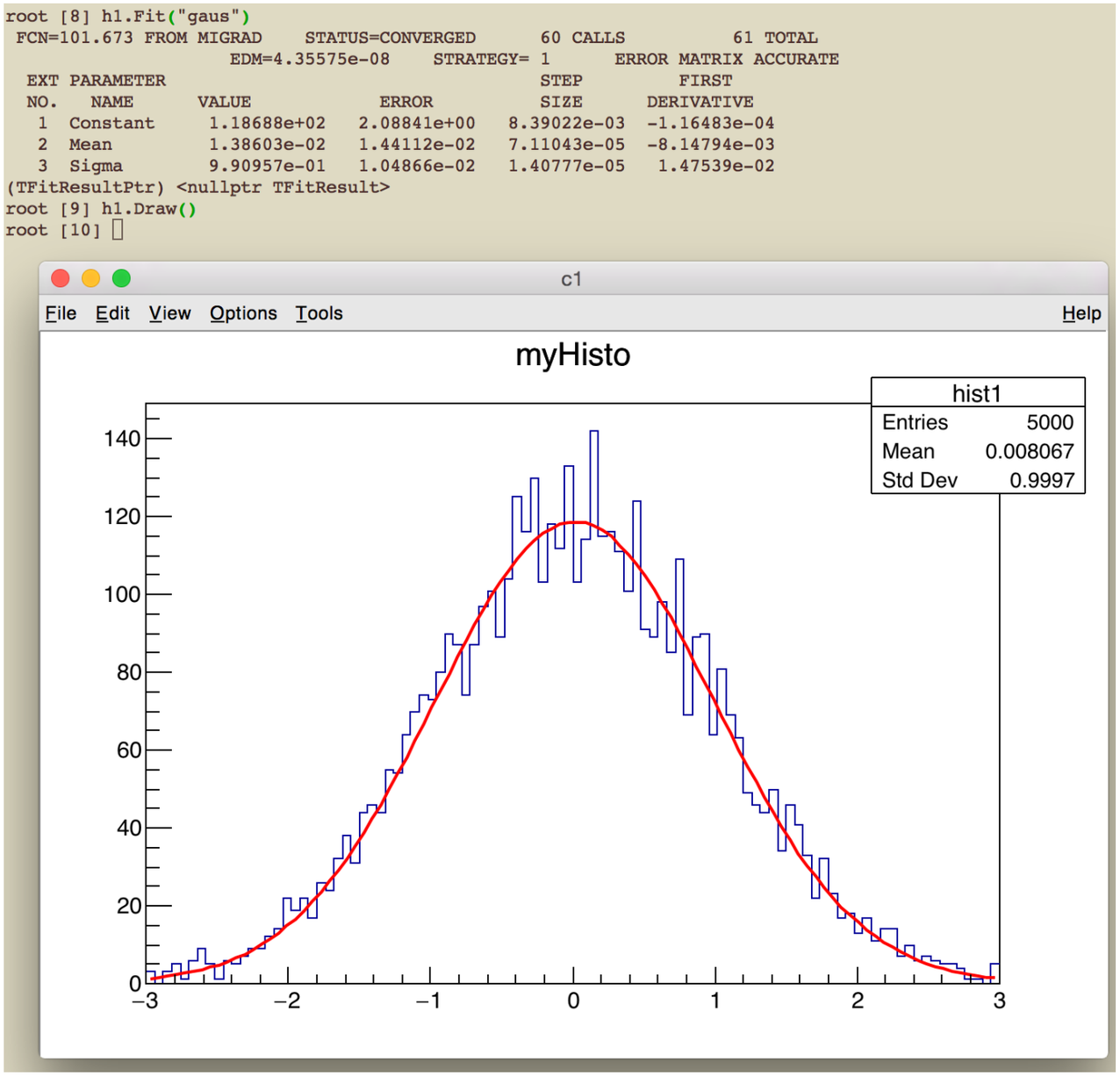
Disable statistics box:
h1.SetStats(0)

https://root.cern.ch/doc/master/classTStyle.html#a906e5f9060357a95f893701b3bed57a2

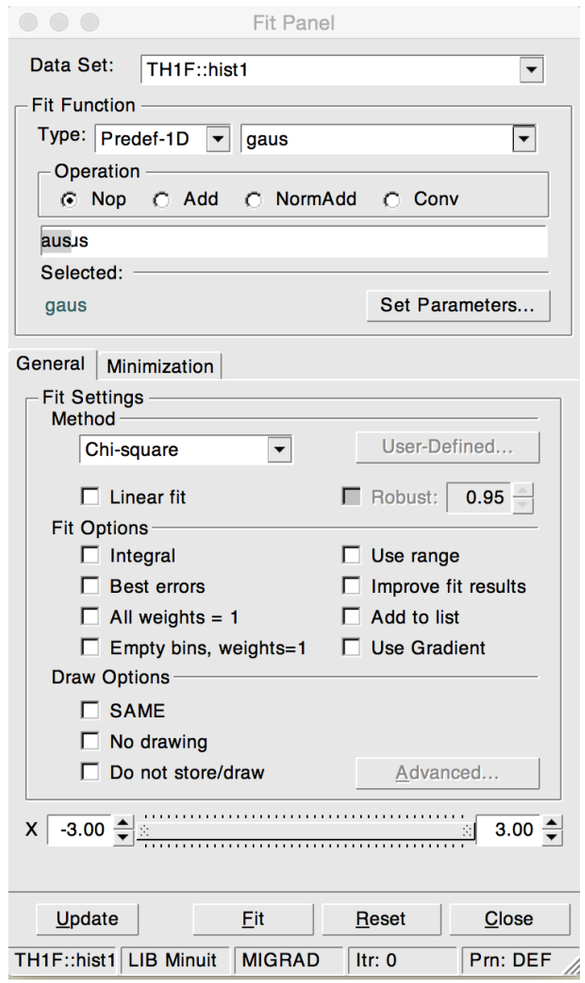
Histograms::Fitting data

Fit histogram with a Gaussian:

```
root [8] h1.Fit("gaus")
 FCN=101.673 FROM MIGRAD    STATUS=CONVERGED      60 CALLS          61 TOTAL
                     EDM=4.35575e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     1.18688e+02   2.08841e+00   8.39022e-03  -1.16483e-04
   2  Mean         1.38603e-02   1.44112e-02   7.11043e-05  -8.14794e-03
   3  Sigma        9.90957e-01   1.04866e-02   1.40777e-05   1.47539e-02
(TFitResultPtr) <nullptr TFitResult>
root [9] h1.Draw()
root [10]
```

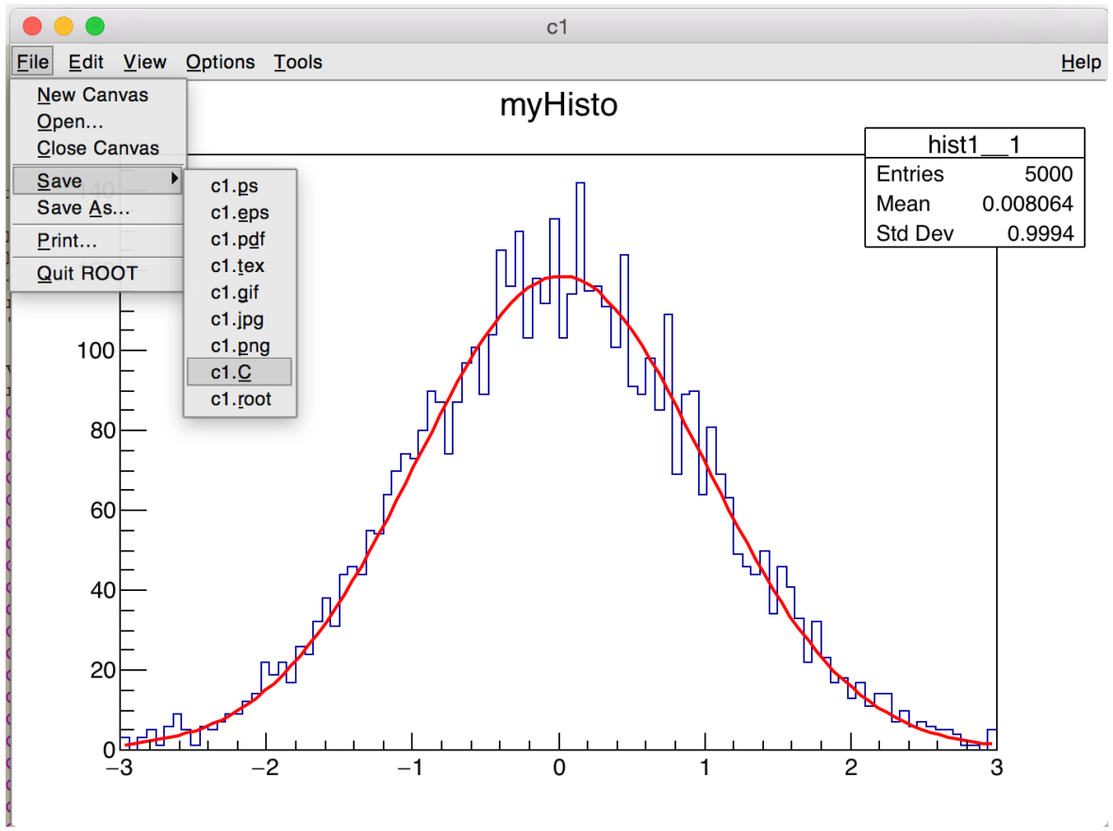


- Can right-click on the histogram and then select FitPanel
- Provides a variety of functions to fit and minimisation methods



Histograms::Operators, saving

- Operations performed with histograms: +, -, *, /
- MUST call Sumw2() before!!!
- Can save everything to a .C file! (or a ROOT file, but let's leave that for later on…)



Hands-on: Create a 2-D histogram of p_T (up to 1 TeV) vs **η** (from -2.5 up to 2.5). Set the momentum resolution to 10 GeV and the pseudorapidity resolution to 0.1. Put labels in the axes and a plot title (p_T vs η). Can you fill it with a random?

Histograms::Help

Histograms documentation (User's guide):
https://root.cern.ch/root/htmldoc/guides/users-guide/Histograms.html#histograms

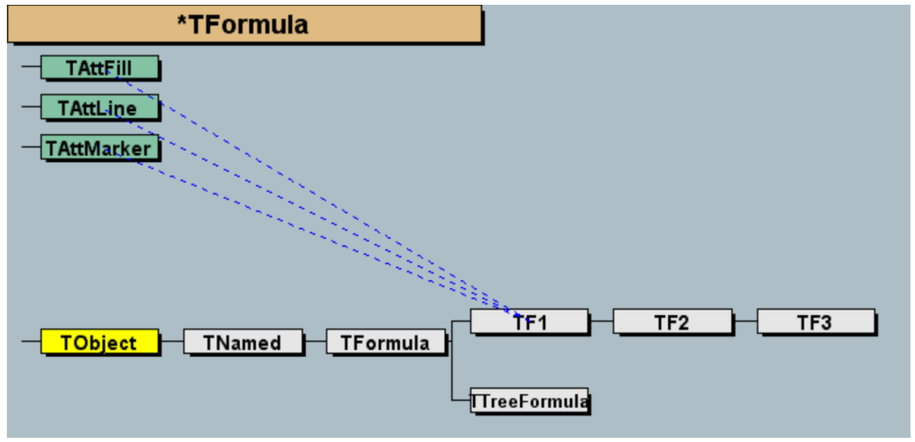
Web histogram tutorials and examples:
https://root.cern.ch/doc/v608/group__tutorial__hist.html

Always available in your root directory:
https://root.cern.ch/root/html/tutorials/hist/index.html

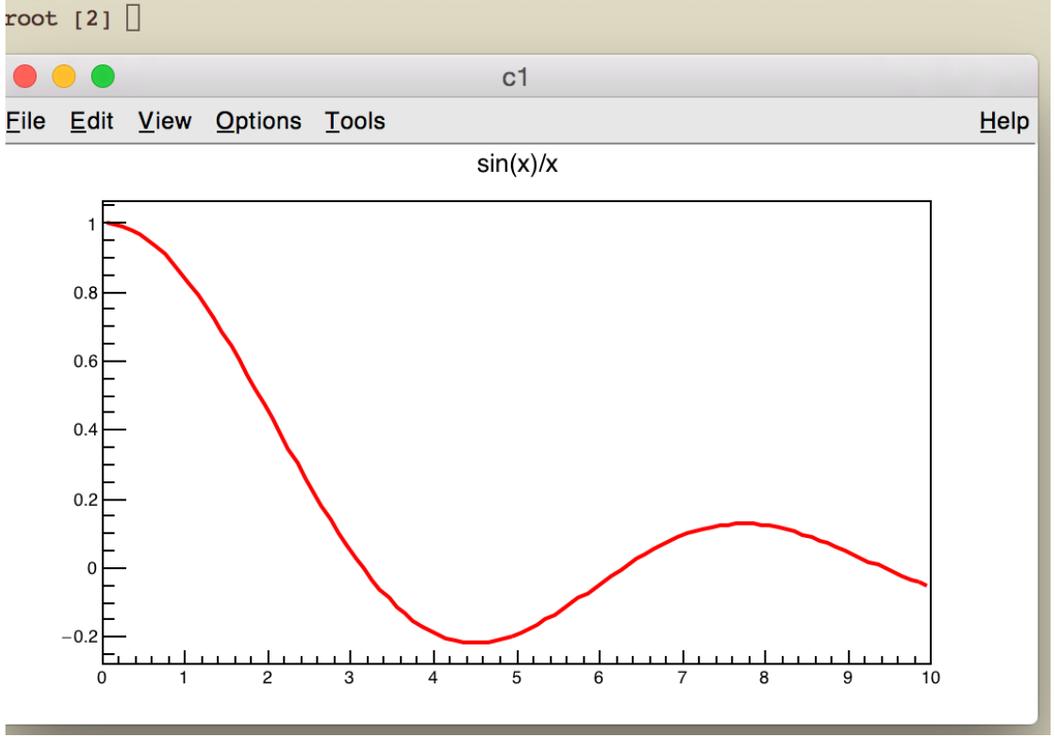
TH1 Class Reference:
https://root.cern.ch/doc/master/classTH1.html

Functions::Definition

- Several functions implemented in ROOT (gauss, landau, polynomial)
- User-defined functions feasible, in 1, 2 or 3 dimensions (TF1,2,3)




```
root [0] TF1 f0("f0", "sin(x)/x", 0., 10.)
(TF1 &) Name: f0 Title: sin(x)/x
root [1] f0.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]
```



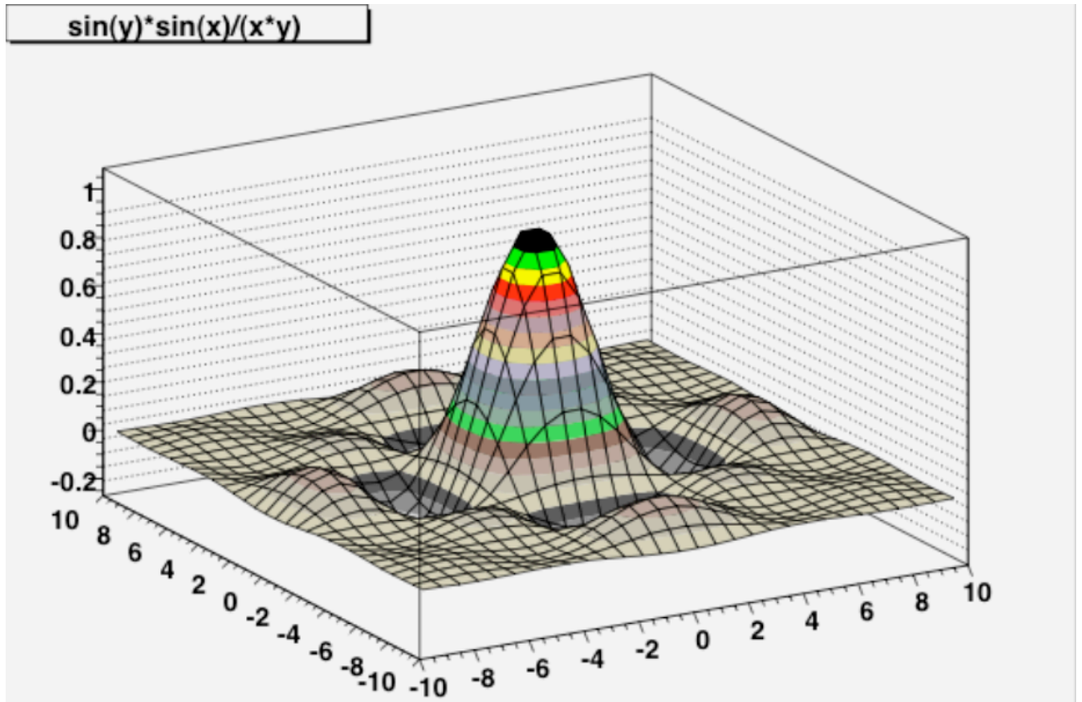
- Can combine functions:
- TF1 myF(“myF”,”3+f0”,0.,10.)

- Parameters: A function may be written in the form:
  TF1 func(“func”,”[0]*x+sin([1]*x)”,0.,10.)

- [0], [1]: parameters, accessed by:
  func->SetParameters(3,0.5);

Functions::Hands-on

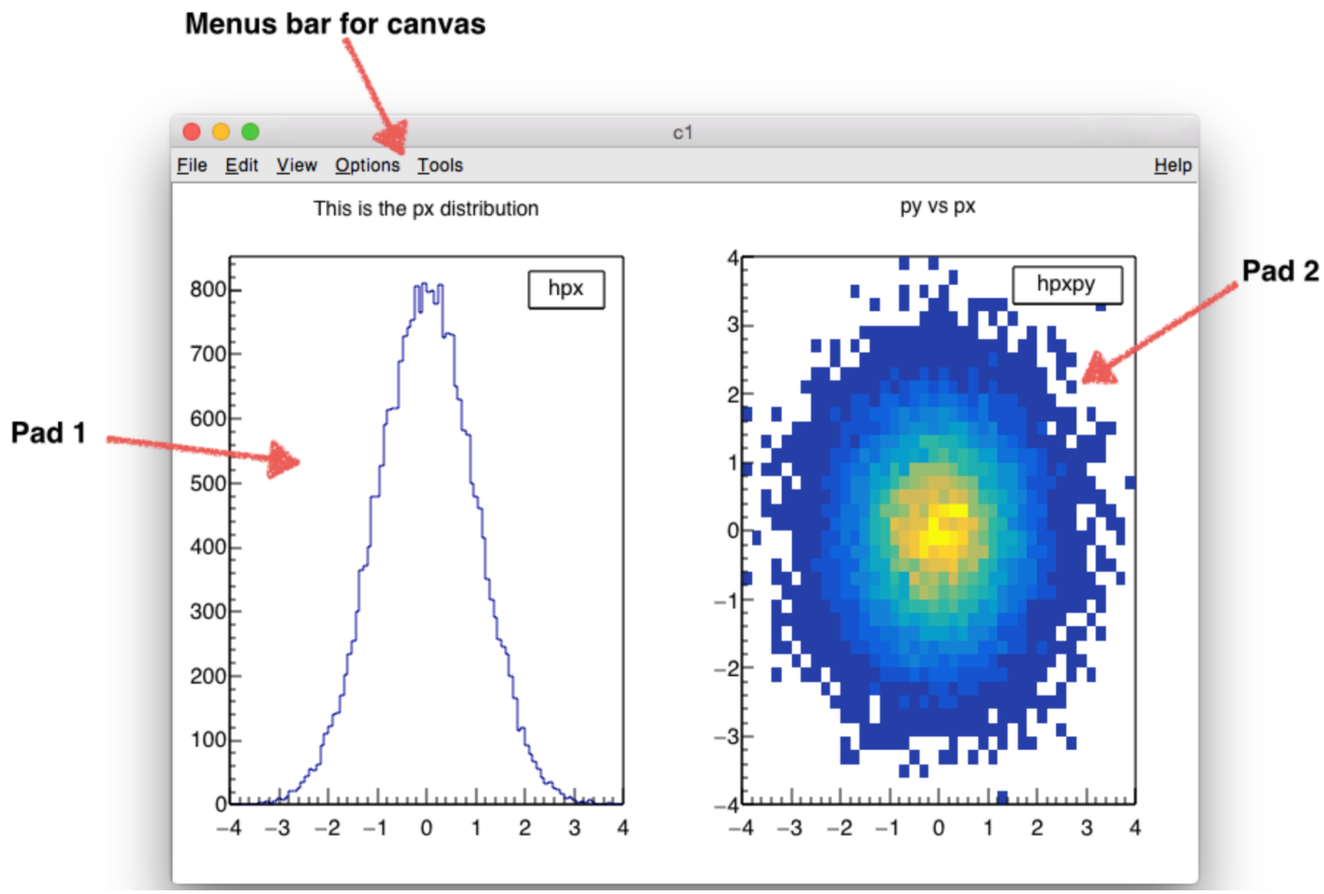
- Produce the following plot:
- You have the formula and range
- You need the drawing option: surf1
  function.Draw("surf1")
- Explore other drawing options



https://root.cern.ch/doc/master/classTF1.html

Canvases and pads

- ROOT draws everything on a canvas
- When hist.Draw() is called, a Canvas is created (c1)
- Can have individual graphic areas within a canvas: pads




```
void TPad::Divide ( Int_t     nx = 1,
                    Int_t     ny = 1,
                    Float_t   xmargin = 0.01,
                    Float_t   ymargin = 0.01,
                    Int_t     color = 0
                  )
```

https://root.cern.ch/doc/master/classTPad.html
https://root.cern.ch/doc/master/classTCanvas.html

More…

Identifier	σύμβαση	παράδειγμα
Classes	αρχίζει με T	THashTable
Non-class types	τελειώνει με _t	Simple_t, Int_t
Enumeration types	αρχίζει με E	EColorLevel
Data members	αρχίζει με f	fViewList
Member functions	αρχίζει με κεφαλαίο	Draw()
Static variables	αρχίζει με g	gSystem
Static data members	αρχίζει με fg	fgTokenClient
Locals and parameters	αρχίζει με μικρό γράμμα	seed, thePad
Constants	αρχίζει με k	kInitialSize, kRed
Template arguments	αρχίζει με A	AType
Getters and setters	αρχίζει με Get, Set, ή Is (boolean)	SetLast(), Get-First(), IsDone()

More…

Built-in variables instead of
C++ assure code portability

· Char_t Signed	1 byte
· UChar_t Unsigned	1 byte
· Short_t Signed short integer	2 bytes
· UShort_t Unsigned short integer	2 bytes
· Int_t Signed integer	4 bytes
· UInt_t Unsigned integer	4 bytes
· Long_t Signed long integer	8 bytes
· ULong_t Unsigned long integer	8 bytes
· Float_t Float	4 bytes
· Double_t Float	8 bytes
· Bool_t Boolean	(kFALSE, kTRUE)

Next

- ROOT files: opening, exploring, creating
- Macros: named and unnamed
- Compiling ROOT code