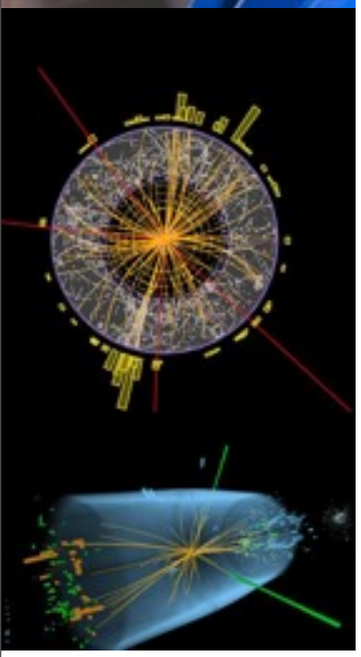


Introduction

Start Using ROOT

ROOT Training at IRMM

25th February 2013

Dr Lorenzo Moneta
CERN PH-SFT
CH-1211 Geneva 23
sftweb.cern.ch
root.cern.ch

Outline

- Starting to work from the ROOT prompt
  - ROOT as a calculator
- Creating and plotting functions
- Plotting data measurements
- Introduction to histograms
- Plotting one-dimensional histograms

We will have interleaving lectures and exercises

Documentation is all available at the ROOT Web site

http://root.cern.ch

ROOT Download & Installation

- Download from ROOT Web site http://root.cern.ch
- Binaries for Linux, MacOS and Windows
- Source files can be built with:
  - `./configure`
    `make`
  - CMake
  - see the instructions on the Web site for building from sources

- *For the training ROOT should have been already installed !*

LET'S FIRE UP ROOT !

Assuming ROOT is now properly installed in your system

Starting Up ROOT

ROOT is prompt-based. Launch ROOT:

```
$ root
```

The ROOT Prompt will appear:

```
root [0] _
```

It “speaks” C++:

```
root [0] gROOT->GetVersion();
(const char* 0x5ef7e8)"5.27/04"
```

ROOT As Pocket Calculator

Calculations:

```
root [0] sqrt(42)
(const double)6.48074069840786038e+00
root [1] double val = 0.17;
root [2] sin(val)
(const double)1.69182349066996029e-01
root [2] TMath::Erf(1.)
(Double_t)8.42700792949714783e-01
```

Uses C++ Interpreter CINT

ROOT Prompt

? Why C++ and not a scripting language?!

! You'll write your code in C++, too. Support for python, ruby,… exists.

? Why a prompt instead of a GUI?

! ROOT is a programming framework, not an office suite. Use GUIs where needed.

Running Code

Macro: a file that is interpreted by CINT

```
int mymacro(int value)
{
  int ret = 42;
  ret += value;
  return ret;
}
```

- Create a new file, `mymacro.C`
- Edit the file and include these above lines.
- Execute from the root prompt:

```
root [0] .x mymacro.C(42)
```

Compiling Code: ACLic

- Load code as shared lib, much faster:

```
root [0] .x mymacro.C+(42)
```

- Uses the system's compiler, takes seconds
- Subsequent **.x mymacro.C+(42)** check for changes, only rebuild if needed
- Exactly as fast as *Makefile* based stand-alone binary!
- CINT knows types and functions in the file
  - e.g. call

```
root [1] mymacro(43)
```

Compiled versus Interpreted

? Why compile?

! Faster execution, CINT has limitations, validate code.

? Why interpret?

! Faster Edit → Run → Check result → Edit cycles ("rapid prototyping"). Scripting is sometimes just easier.

? Are Makefiles dead?

! Yes! ACLiC is even platform independent!

Controlling ROOT

- Useful CINT commands from the ROOT prompt:
  - quit ROOT

```
root [1] .q
```

  - to get the list of available commands

```
root [1] .?
```

  - to access the shell of the OS (e.g UNIX or MS/DOS)

```
root [1] .!  <OS_command>
```

e.g.: .! pwd

  - to execute a macro (add a + at the end for compiling with ACLIC)

```
root [1] .x <file_name>
```

e.g.: .x mymacro.C
.x mymacro.C+

  - to load a macro

```
root [1] .L <file_name>
```

e.g.: .L mymacro.C
.L mymacro.C+

Time for Exercises!

Put in practice the concepts to which you were just exposed: read the instructions here

https://twiki.cern.ch/twiki/bin/view/Main/RootIRMMTutorial2013StartingROOTExercises

and solve exercise 1

Plotting a function

- Start using one of basics ROOT classes - the function class TF1:

```
root [0] TF1 * f1 = new TF1("f1","sin(x)/x",0,10);
```

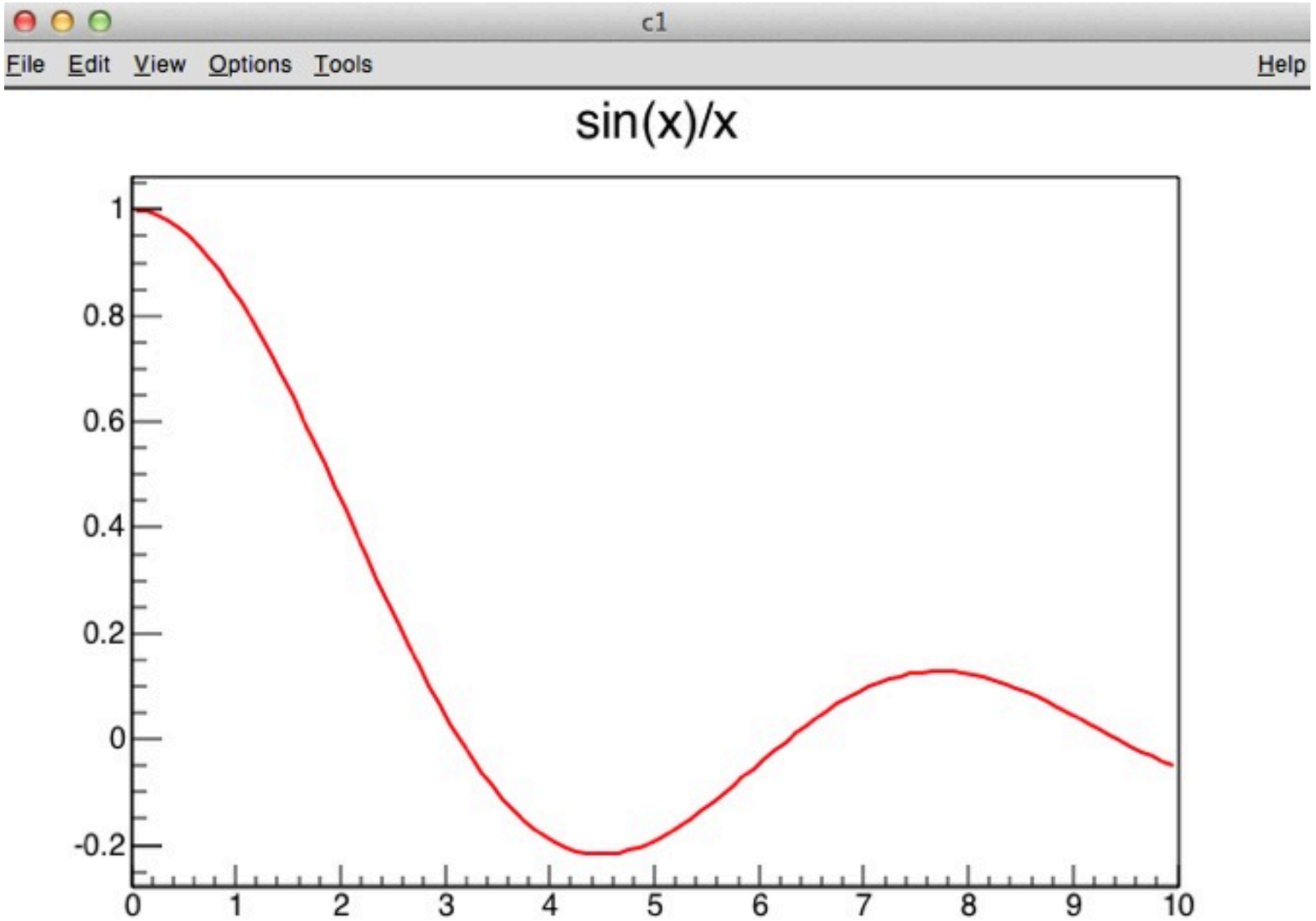
pointer — object name — function type (formula) — function range

- Draw the function:

```
root [1] f1->Draw();
```

ROOT Canvas

- The Function will be drawn in a ROOT Canvas
  - The Canvas has a GUI Menu



Function With Parameters

- Use the ROOT formula syntax to create a function with parameters, e.g. 2 parameters called [0] and [1]:

```
root [2] TF1 * f2 = new TF1("f2","[0]*sin([1]*x)/x",0,10);
```

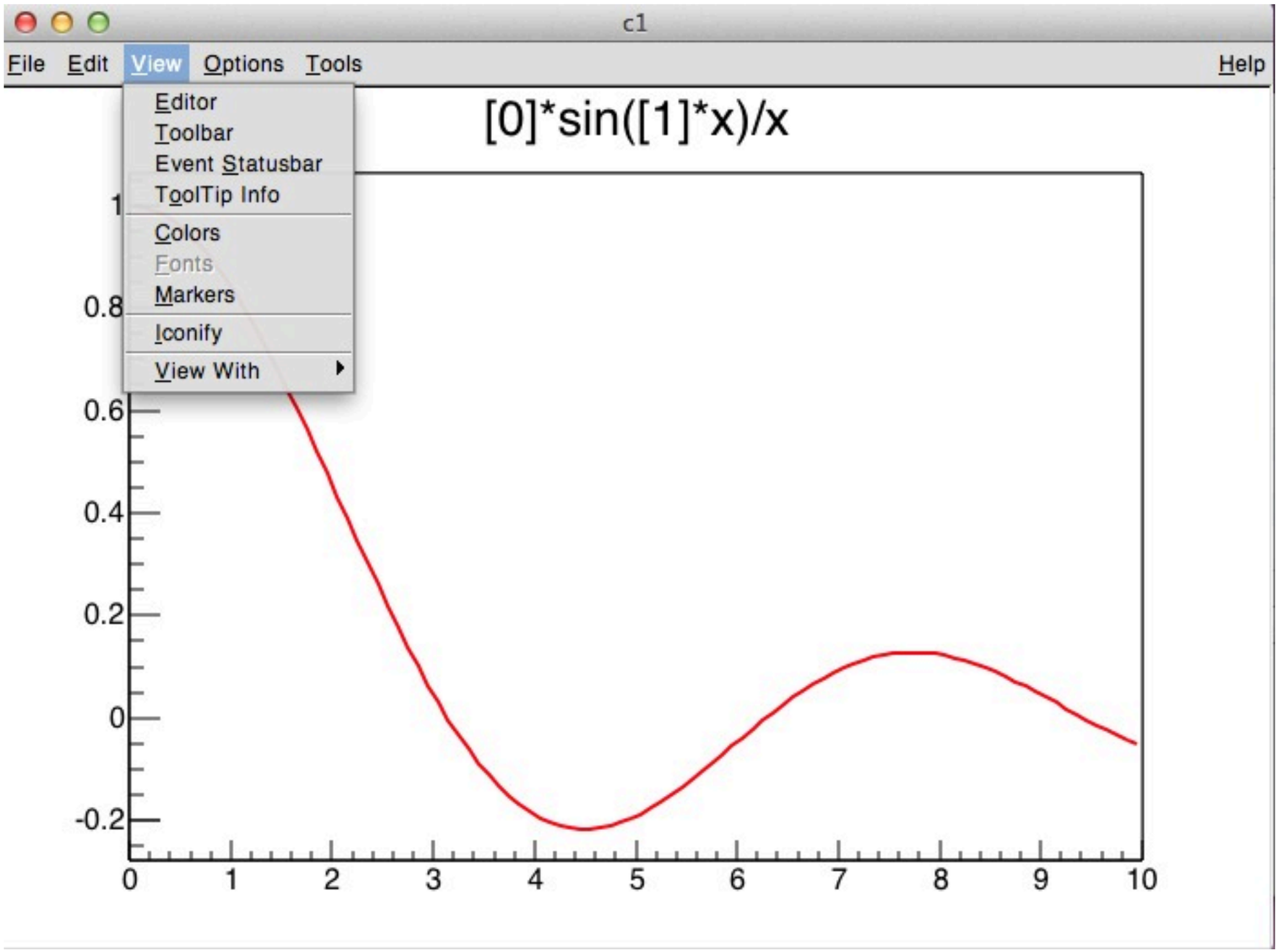
- Need to set the parameter values (by defaults the parameters have zero initial values):

```
root [3] f2->SetParameter(0,1);
root [4] f2->SetParameter(1,1);
root [5] f2->Draw();
```

- Can also use the GUI (function editor) to change the function parameters.

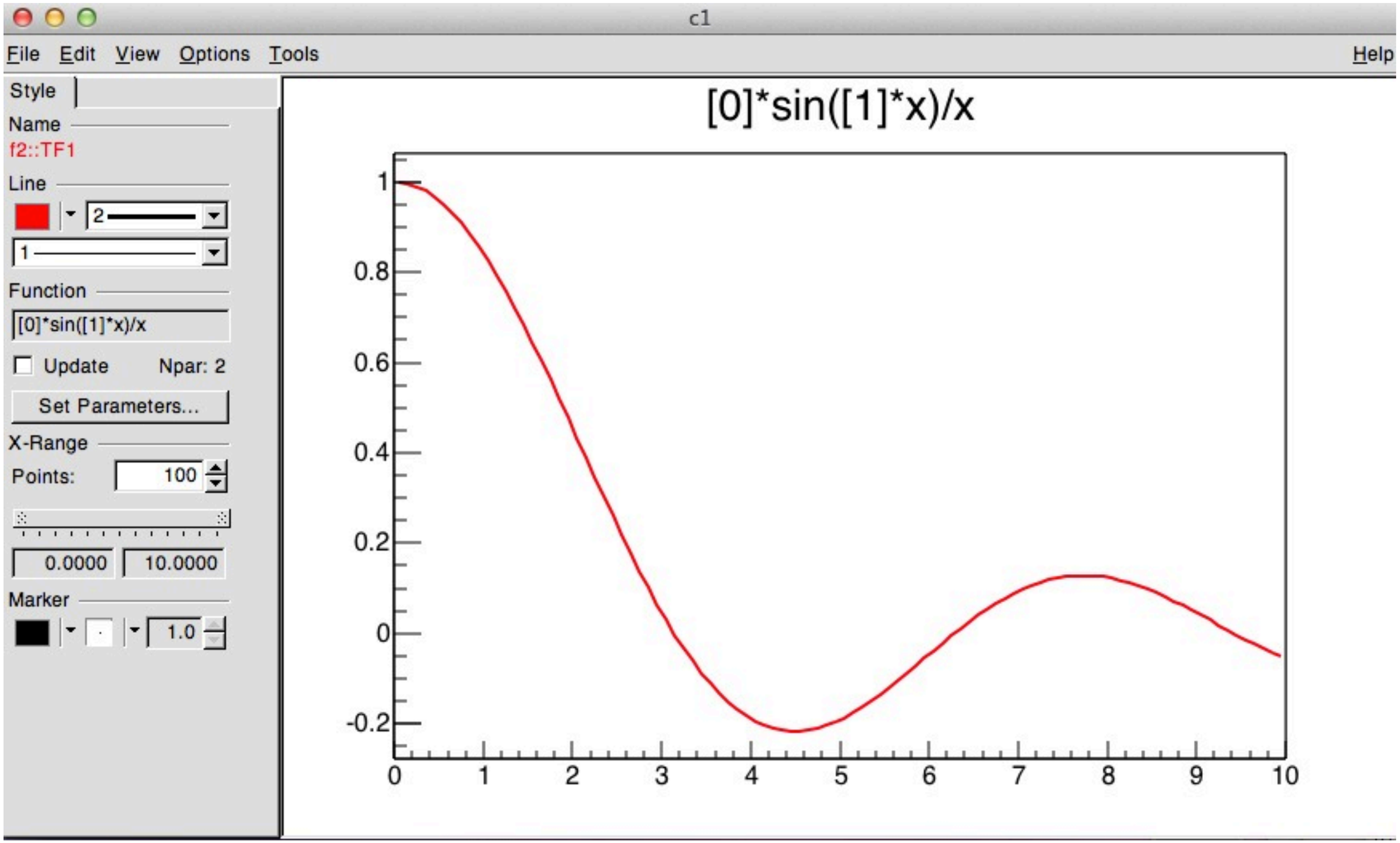
GUI Editor for ROOT Objects

- View the GUI Editor from the Canvas View Menu



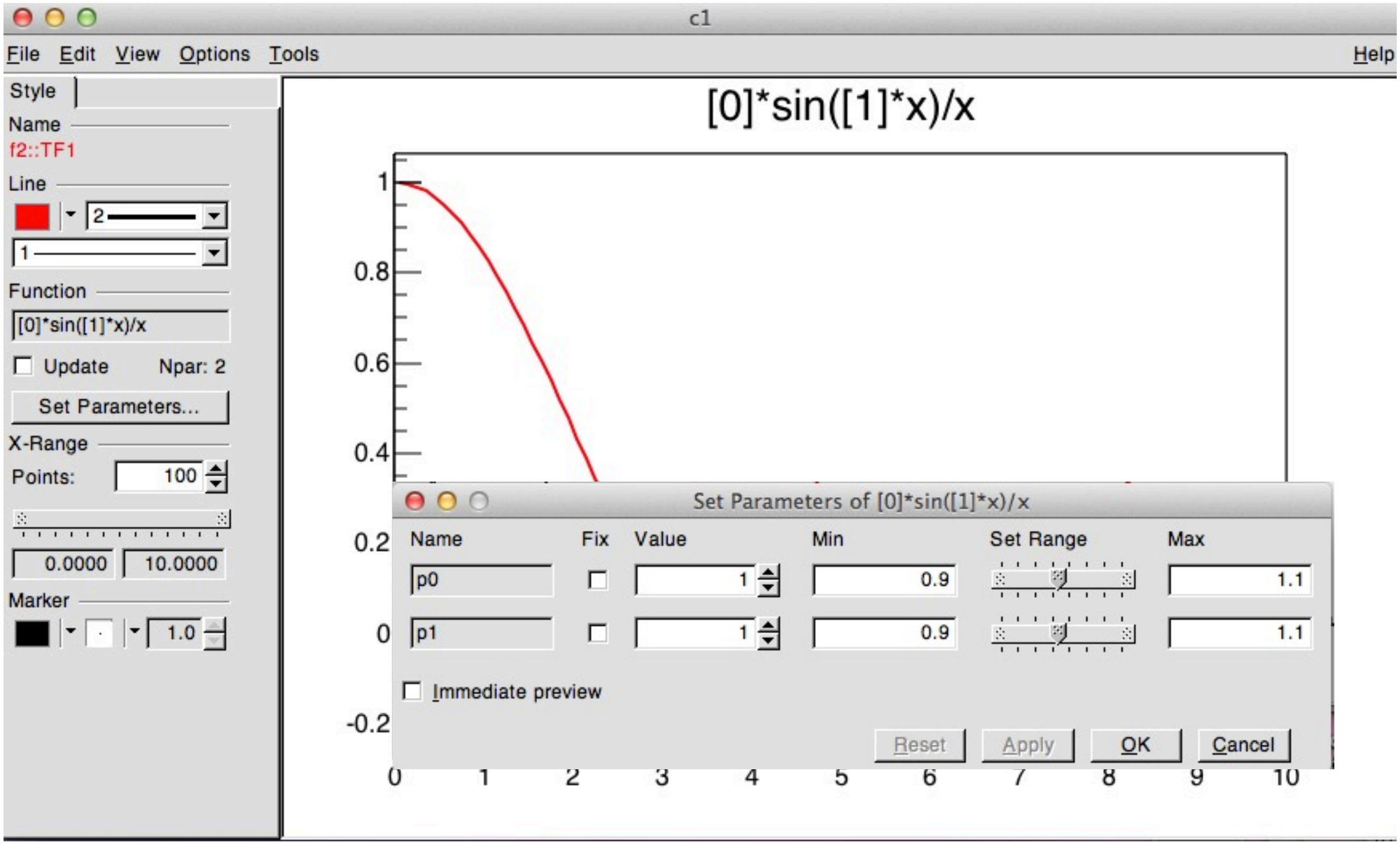
Function GUI Editor

- Click on the function with the mouse



Function GUI Editor: Set Parameters

- Click on “Set Parameters”



Time for Exercises!

Put in practice the concepts to which you were just exposed: read the instructions here

https://twiki.cern.ch/twiki/bin/view/Main/RootIRMMTutorial2013StartingROOTExercises

and solve exercise 2

Mathematical Functions in ROOT

- **`TMath`**: a namespace providing the following functionality:
  - Numerical constants.
  - Trigonometric and elementary mathematical functions.
  - Functions to work with arrays and collections (e.g sort, min max of arrays,...)
  - Statistic Functions (e.g. Gauss)
  - Special Mathematical Functions (e.g. Bessel functions)
  - For more details, see the reference documentation of TMath.

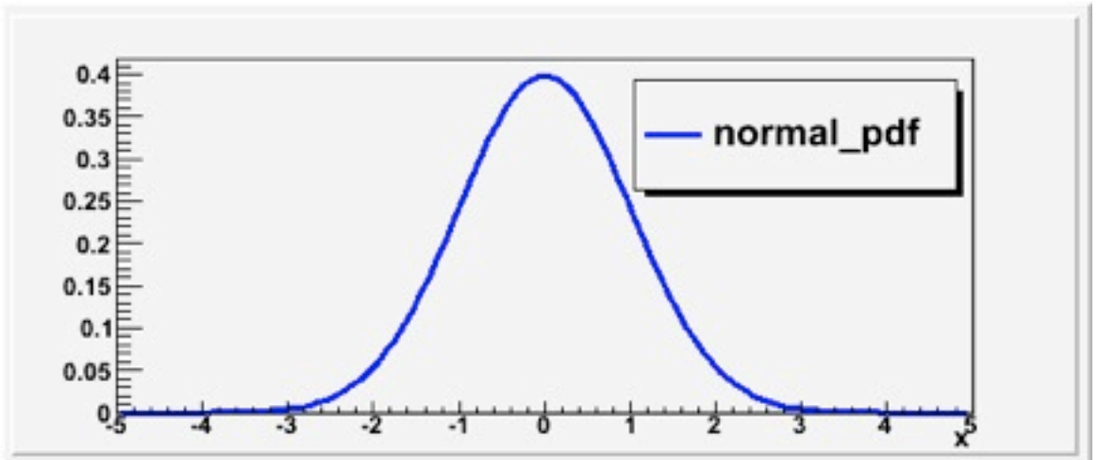
```
TMath::Gaus( x, mean, sigma);
```

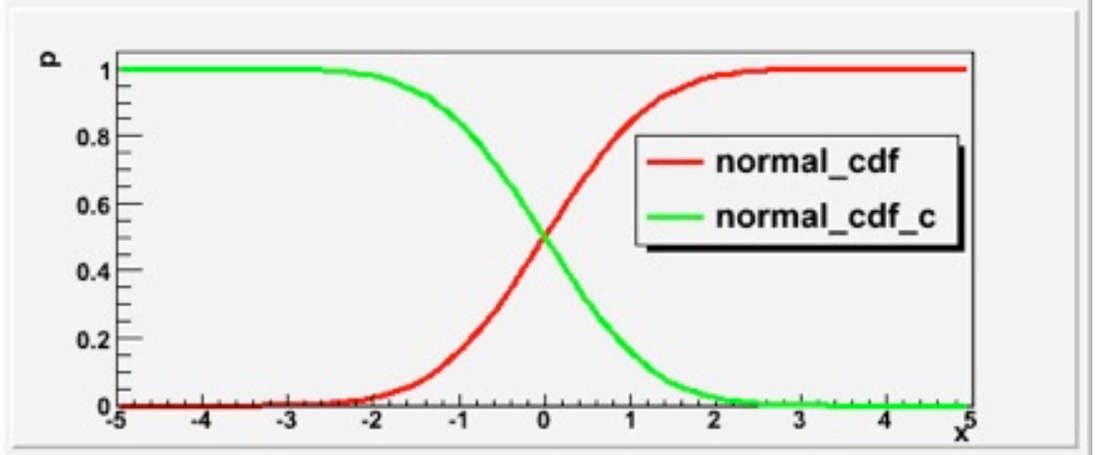
- Functions provided in **`ROOT::Math`** namespace
  - special functions (many implemented using the Gnu Scientific Library)
  - statistical functions
  - For more details, see the reference documentation of ROOT::Math functions

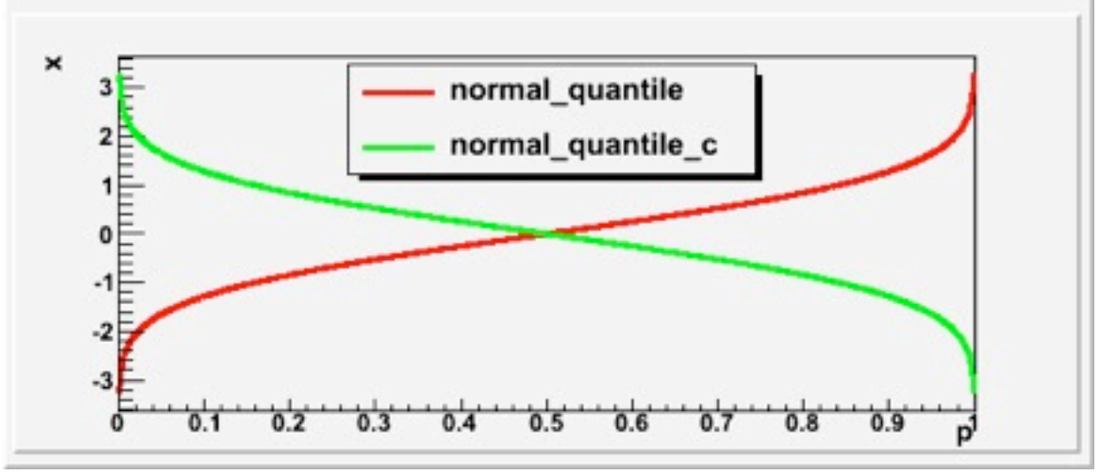
```
ROOT::Math::cyl_bessel_i(nu,x);
```

ROOT::Math Statistical Functions

- Statistical Functions are provided in a coherent naming scheme
  - probability density functions (pdf)
    - i.e. normal distribution:
      - `normal_pdf( x, sigma, mu)`
  - cumulative distributions (cdf)
    - lower tail: `normal_cdf (x, sigma, mu)`
    - upper tail: `normal_cdf_c (x, sigma, mu)`
  - inverse of cumulative distributions (quantiles)
    - inverse of lower cumulative
      - `normal_quantile (z, sigma)`
    - inverse of lower cumulative
      - `normal_quantile_c (z, sigma)`
- All major statistical distributions available
  - *normal, lognormal, Landau, Cauchy, χ^2, gamma, beta, F, t, poisson, binomial, etc..*
- Defined as free functions in `ROOT::Math` namespace







Examples for using Distributions

- p values
  - probability after a fit

```
double prop = ROOT::Math::chisquared_cdf_c(val, ndf);
```

- significance : (number of sigma's)
  - probability α to observe s or larger signal in the case of pure background fluctuation

$$\alpha = \int_{n\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \mathrm{d}x = 1 - \frac{1}{2}\mathrm{erf}\left(\frac{n}{\sqrt{2}}\right)$$

```
double sig = ROOT::Math::normal_quantile_c(alpha,1.);
```

  - implementation is careful to avoid numerical error
    - uses inverse error function or its complement depending on whether α is close to 0 or 1.

Plotting Measurements

- The Graph class (**`TGraph`**):
  - for plotting and analyzing 2-dimensional data (X,Y),
  - contains a set of N distinct points (X_i , Y_i) i = 1,..N.
  - can be constructed from a set of x,y arrays:

```
root [1] double x[] = { 1,2,3,4,5};
root [2] double y[] = { 0.5,2.,3.,3.2,4.7};
root [3] TGraph * g = new TGraph(5,x,y);
```

  - or directly from a text file containing rows of (X,Y) data

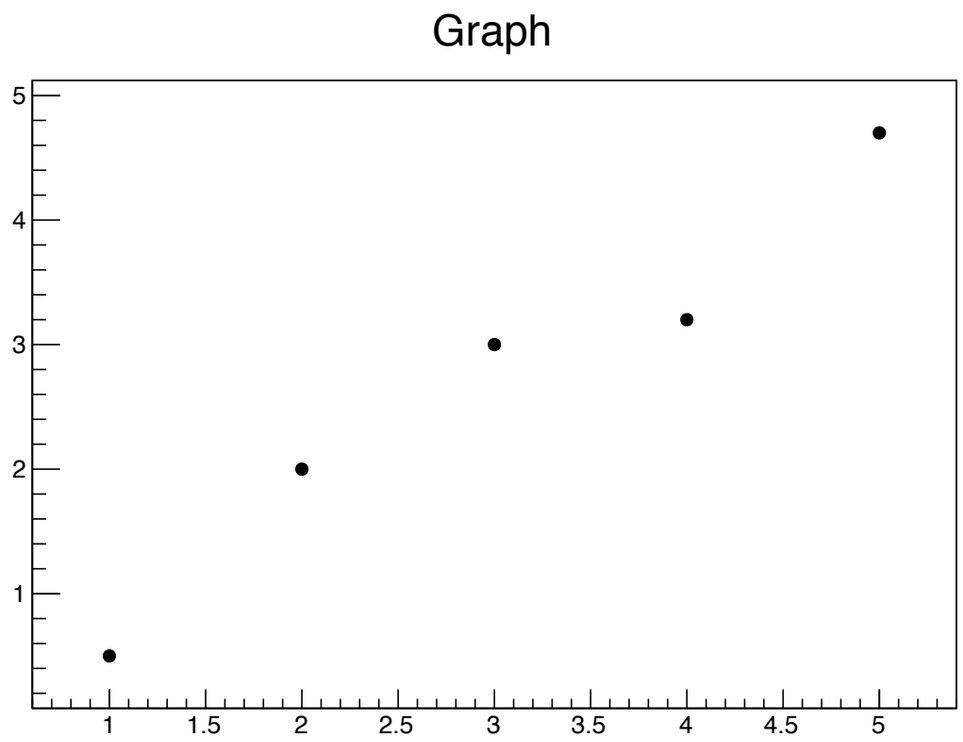
```
root [1] TGraph * g = new TGraph("XYData.txt");
```

Displaying a Graph

- To display the graph:

```
root [4] g->Draw("AP");
```

  - option “A” means displaying the axis,
  - option “P” means displaying the points.

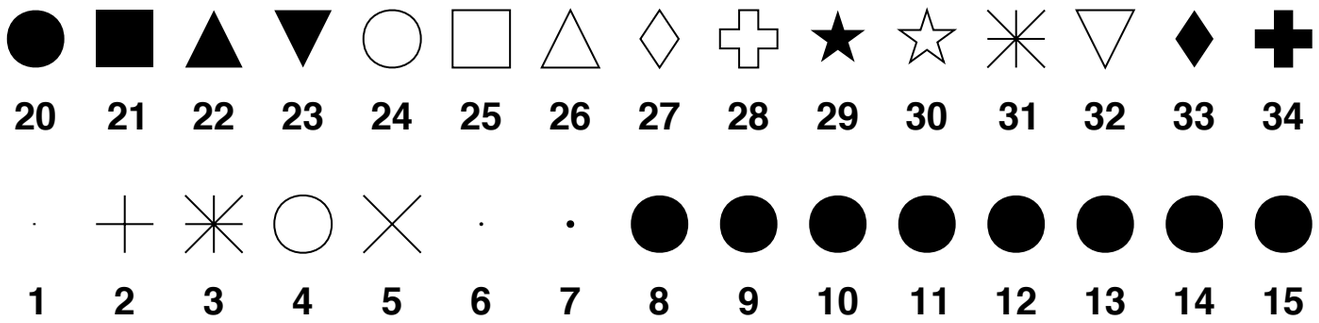


To change the point markers do:

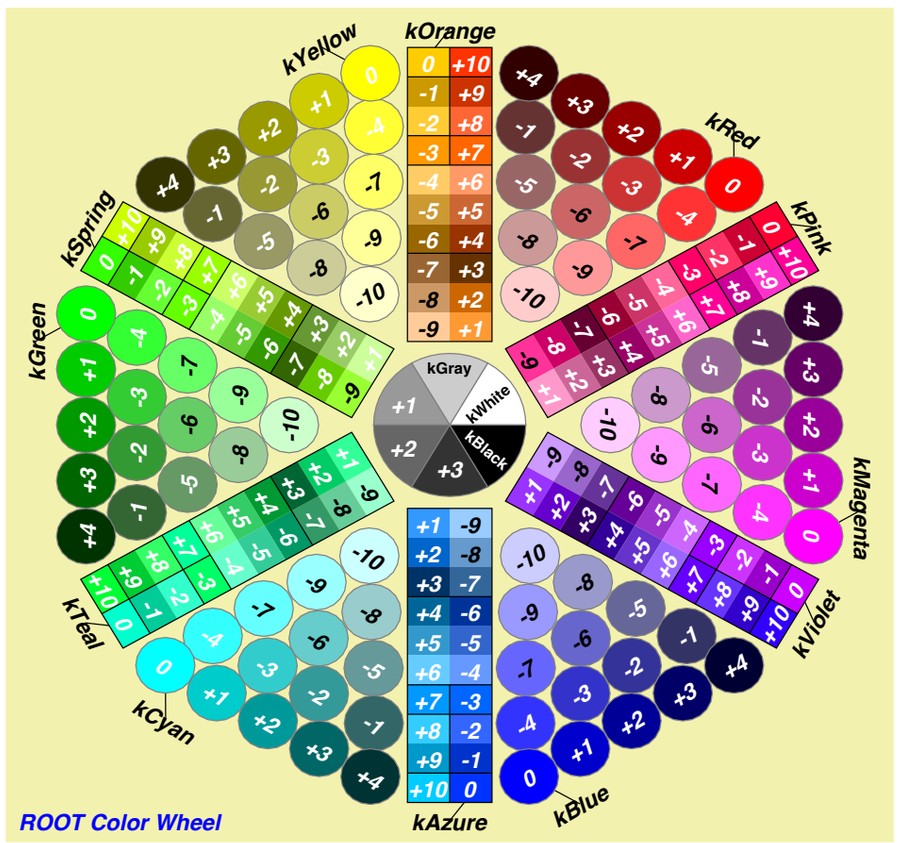
```
root [4] g->SetMarkerStyle(20);
```

Markers and Colors

- Available markers in ROOT



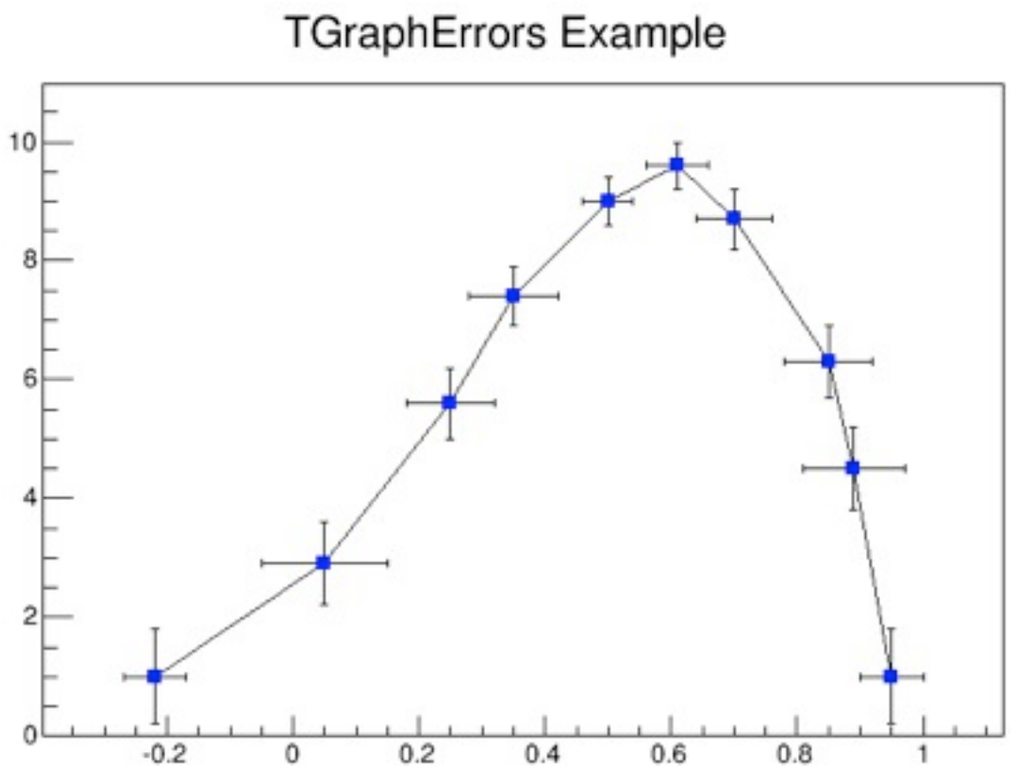
- Available Colors



you can access them from the Canvas View Menu

Types of Graphs

- ROOT provides various types of Graphs:
  - **`TGraph`**： (x,y) data points.
  - **`TGraphErrors`**:
    - (x,y) data points with error bars (σ_x,σ_y).
  - **`TGraphAsymmErrors`**:
    - (x,y) data points with asymmetric error bars [(σ^-_x,σ^+_x),(σ^-_y,σ^+_y)].




```
TGraphErrors *gr = new TGraphErrors(n,x,y,ex,ey);
gr->SetTitle("TGraphErrors Example");
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->Draw("ALP");
```

Use the drawing option “L” for displaying a line connecting the points

Graphs Drawing Options

- The drawing of Graphs is done via the **`TGraphPainter`**
  - see http://root.cern.ch/root/html/TGraphPainter.html
  - the documentation lists all drawing options for the different types of graphs available in ROOT

Graphs' plotting options

Graphs can be drawn with the following options:

- **"A"** Axis are drawn around the graph
- **"L"** A simple polyline is drawn
- **"F"** A fill area is drawn ('CF' draw a smoothed fill area)
- **"C"** A smooth Curve is drawn
- **"*"** A Star is plotted at each point
- **"P"** The current marker is plotted at each point
- **"B"** A Bar chart is drawn
- **"1"** When a graph is drawn as a bar chart, this option makes the bars start from the bottom of the pad. By default they start at 0.
- **"X+"** The X-axis is drawn on the top side of the plot.
- **"Y+"** The Y-axis is drawn on the right side of the plot.

Drawing options can be combined. In the following example the graph is drawn as a smooth curve (option "C") with markers (option "P") and with axes (option "A").

Time for Exercises!

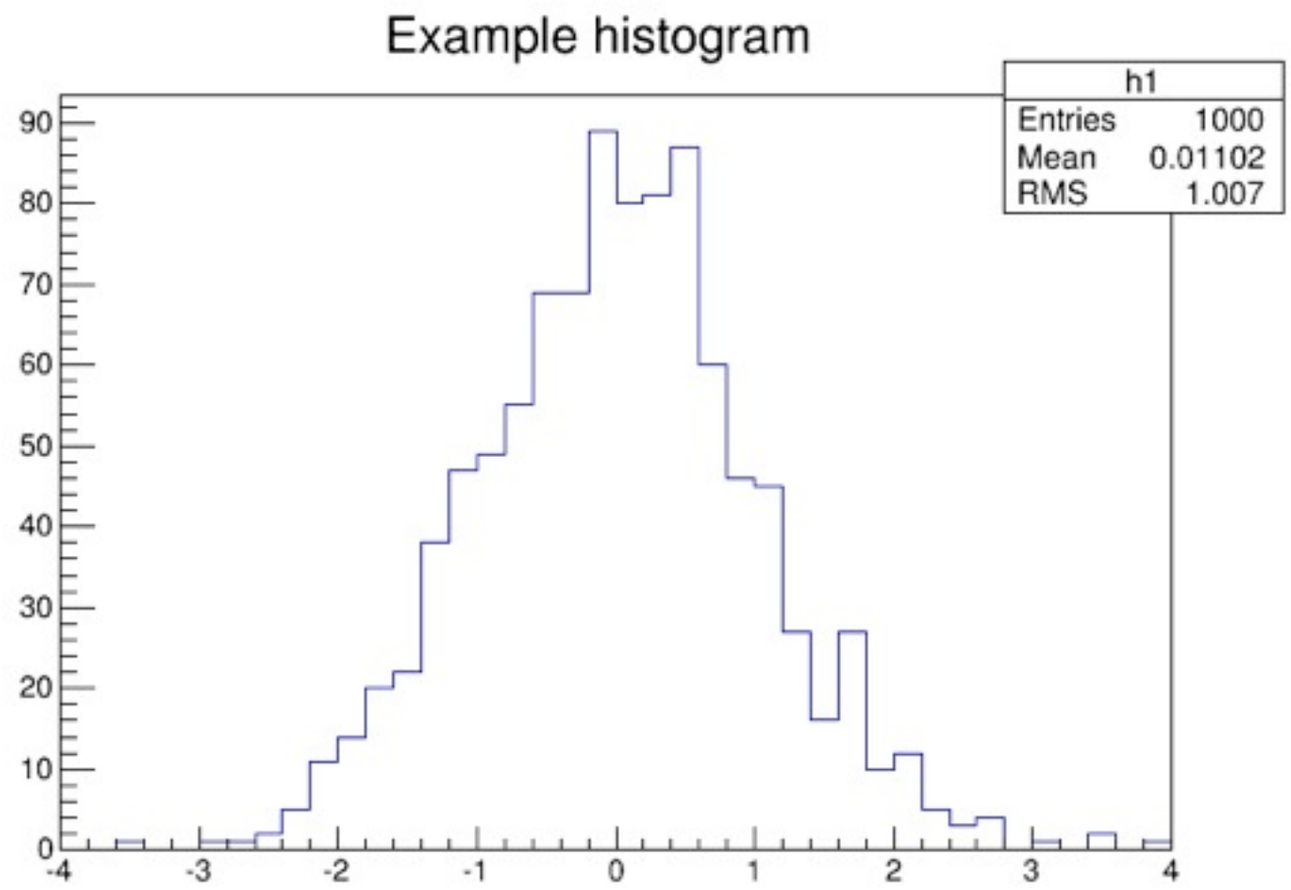
Put in practice the concepts to which you were just exposed: read the instructions here

https://twiki.cern.ch/twiki/bin/view/Main/RootIRMMTutorial2013StartingROOTExercises

and solve exercise 3

Histograms

- What is a histogram?
  - *from Wikipedia:*
  - a **histogram** is a graphical representation showing a visual impression of the distribution of data. It is an estimate of the probability distribution of a continuous variable and was first introduced by Karl Pearson.[1]
  - A histogram consists of tabular frequencies, shown as adjacent rectangles, erected over discrete intervals (bins), with an area equal to the frequency of the observations in the interval.
  - The height of a rectangle is also equal to the frequency density of the interval, i.e., the frequency divided by the width of the interval. The total area of the histogram is equal to the number of data entries.



Histograms in ROOT

- Used to display and estimate the distribution of a variable (e.g. observed energy spectrum)
  - visualize number of events in a certain range, called *bin*
  - *bins* typically have equal widths, but not always
    - ROOT supports histograms with equal and variable bins
- Histograms can be used for further analysis
  - e.g to understand the underlying parent distribution
- ROOT provides various types of histograms depending on:
  - contained data type (double, float, integer, char)
  - choice of uniform or variable bins
  - dimension (1,2 or 3)

How to use ROOT Histograms

- Example of creating a one-dim. histogram:

```
TH1D * h1 = new TH1D("h1","Example histogram",40,-4.,4.);
```

histogram type
TH1D: one-dimension
using double types

name

title

axis settings

number of bins

min,max values

- Filling histogram:

```
h1->Fill(x);
```

Fill the histogram with one observation “

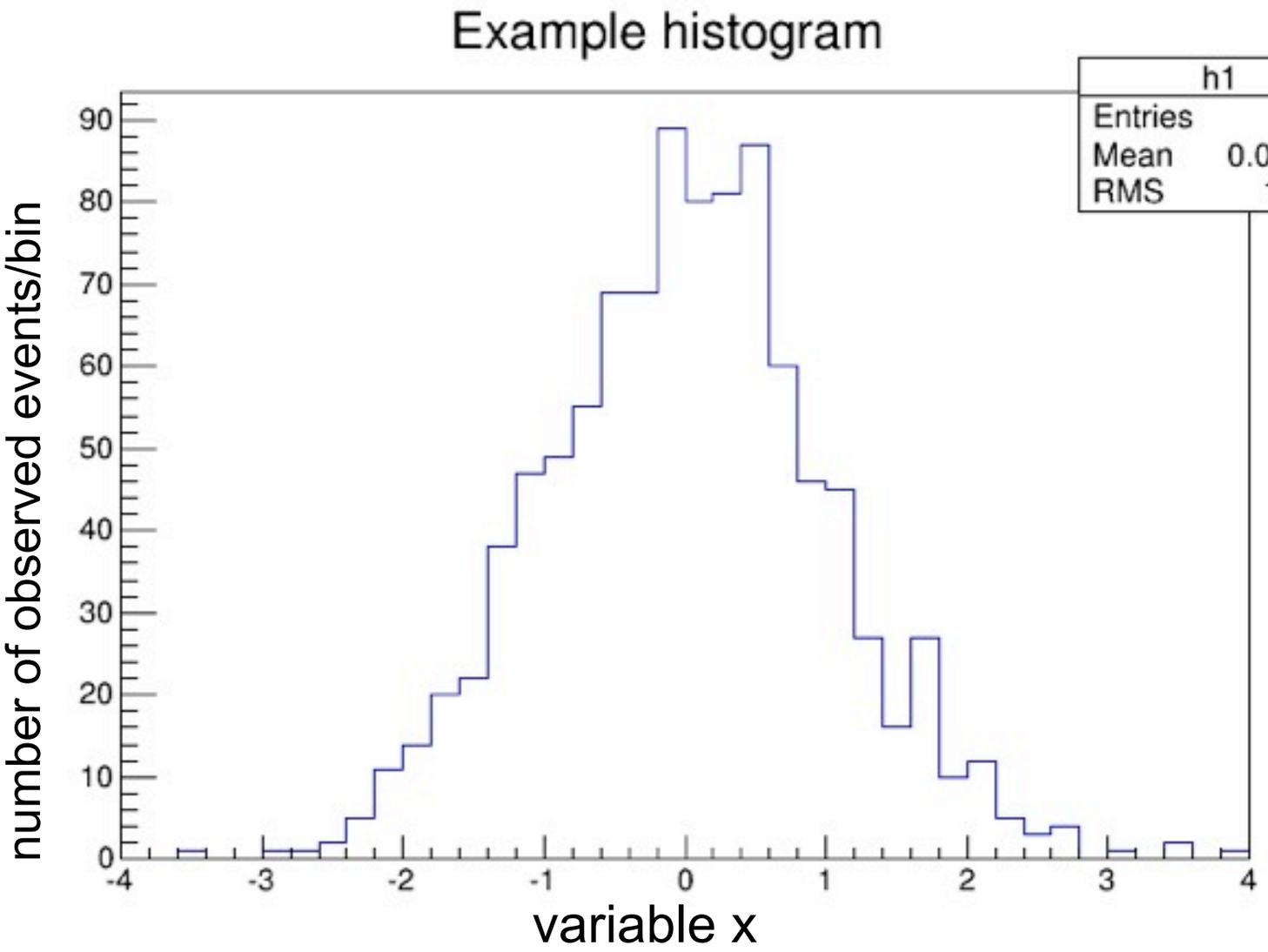
```
for (int i = 0; i<1000; ++i) {
  double x = gRandom->Gaus(0,1);
  h1->Fill(x);
}
```

Fill the histogram with 1000 gaussian distributed random numbers

Displaying ROOT histograms

- Drawing histograms in a ROOT canvas:

```
h1->Draw();
```



histogram name

The histogram statistics:

- Total number of entries
- Sample Mean
- Sample Standard Deviation (RMS)

Histogram Statistics

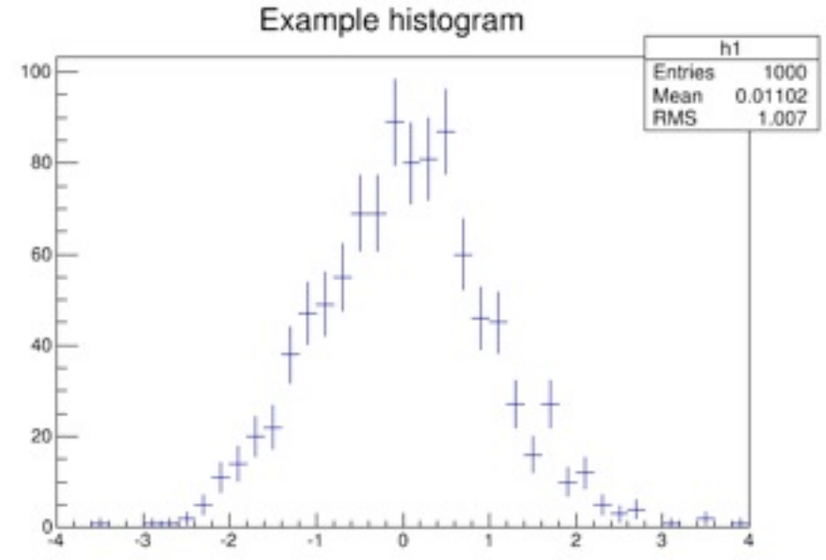
- To extract statistics information from an histogram:

```
root [] h1->GetEntries()
(const Double_t)1.0000000000000000e+03
root [] h1->Integral()
(const Double_t)1.0000000000000000e+03
root [] h1->GetMean()
(const Double_t)1.1017279203592710e-02
root [] h1->GetMeanError()
(const Double_t)3.1831174486931387e-02
root [] h1->GetRMS()
(const Double_t)1.0065901197694480e+00
root [] h1->GetRMSError()
(const Double_t)2.2508039332841407e-02
root [] h1->GetSkewness()
(const Double_t)1.1782073846449019e-01
root [] h1->GetKurtosis()
(const Double_t)2.5896196835884000e-01
```

Histogram Drawing Options

- Various drawing options are available:
  - draw error bars on every bin

```
h1->Draw("E");
```



  - "SAME"

```
h1->Draw("SAME");
```

    - draw the histogram on the canvas without replacing what is already there
    - use to plot one histogram on top of another
- The default drawing option is "HIST" for histograms without errors (unweighted histograms) and "E" for weighted histograms
- For displaying the histogram in log scale in one axis, e.g. the y axis:

```
gPad->SetLogy();
```

Histogram Drawing Options

- Histogram drawing is handled internally by the **`THistPainter`** class.
- The documentation for all the drawing options can be found in the class reference page
  - http://root.cern.ch/root/html/THistPainter.html

class THistPainter: public TVirtualHistPainter

The histogram painter class

- Introduction
- Histograms' plotting options
  - Options supported for 1D and 2D histograms
  - Options supported for 1D histograms
  - Options supported for 2D histograms
  - Options supported for 3D histograms
  - Options supported for histograms' stacks (THStack)
- Setting the Style
- Setting line, fill, marker, and text attributes
- Setting Tick marks on the histogram axis
- Giving titles to the X, Y and Z axis
- The option "SAME"
  - Limitations
- Superimposing two histograms with different scales in the same pad
- Statistics Display
- Fit Statistics
- The error bars options
- The bar chart option
- The "BAR" and "HBAR" options

Time for Exercises!

Put in practice the concepts to which you were just exposed: read the instructions here

https://twiki.cern.ch/twiki/bin/view/Main/RootIRMMTutorial2013StartingROOTExercises

and solve exercise 4

Global Pointers

- `gSystem`: Interface to the operating system.
- `gStyle` : Interface to the current graphics style.
- `gPad`: Interface to the current graphics Pad.
- `gROOT`: Entry point to the ROOT system.
- `gRandom`: Interface to the current random number generator.

Summary

- How to work from the ROOT prompt
  - how to run interpreted C/C++ code
  - what is a ROOT macro
  - how to run compiled macro with ACLic
- Started looking at some basics ROOT objects
  - functions
  - graphs for plotting measurements
  - histograms
- How to plot these objects

- We did not pay attention at the C++ syntax
  - next lecture will look in more detail at C++