ROOT Basics

Deb Mohapatra

This lecture is based on Fermilab ROOT tutorial and ROOT lecture in CERN summer school

Introduction

WHAT IS ROOT ?

- ROOT is an object oriented framework
- It has a C/C++ interpreter (CINT) and C/C++ compiler (ACLIC)
- ROOT is used extensively in High Energy Physics for “data analysis”
  - Reading and writing data files
  - Calculations to produce plots, numbers and fits.

WHY ROOT ?

- It can handle large files (in GB) containing N-tuples and Histograms
- Multiplatform software
- Its based on widely known programming language C++
- Its free

Outline of this lecture

- Overview of ROOT framework
- GUI and command line basics
- CINT: Interpreter for C and C++ code
- Graphs, Histograms and Root Trees
- Functions and fitting

Learning ROOT

- http://root.cern.ch/root/Tutorials.html
- http://root.cern.ch/root/HowTo.html
- http://www-root.fnal.gov/root/
- http://www.slac.stanford.edu/BFROOT/www/doc/workbook/root{1,2,3}/root{1,2,3}.html

ROOT Mailing Lists

- roottalk@root.cern.ch
- http://root.cern.ch/root/roottalk/AboutRootTalk.html

Root Interactive Session

- Set ROOTSYS to the directory where ROOT is installed
- Add ROOT libraries to the LD_LIBRARY_PATH
- Include the ROOT executable binary files to the binary path

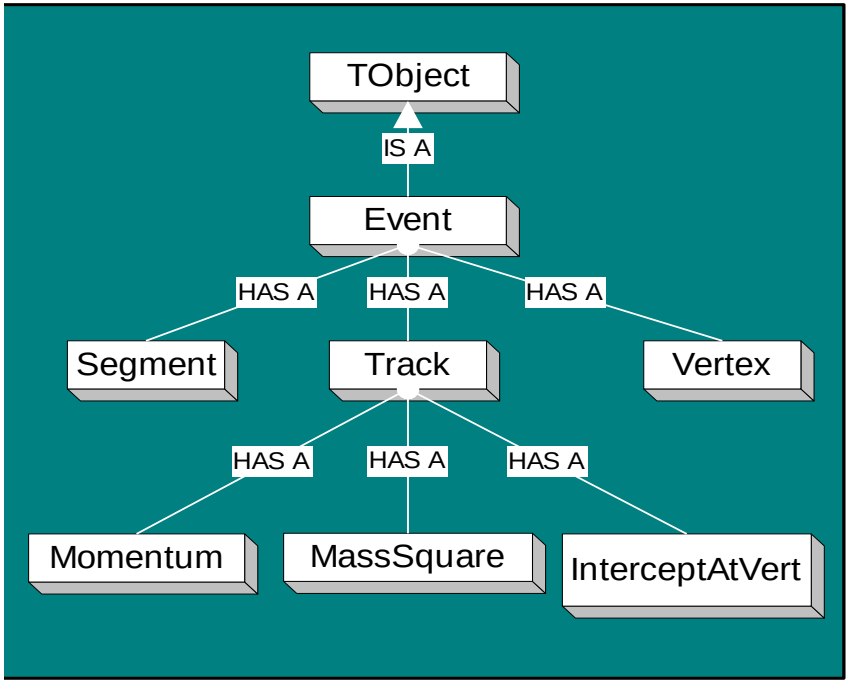
BASH	export ROOTSYS=/cern/root export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH export PATH=$ROOTSYS/bin:$PATH
TCSH	setenv ROOTSYS /cern/root setenv LD_LIBRARY_PATH $ROOTSYS/lib:$LD_LIBRARY_PATH setenv PATH $ROOTSYS/bin:$PATH

- You may add the above lines to your ~/.cshrc or ~/.bashrc
- You may define your root settings in ~/.rootlogon.C
- History of all commands are stored in ~/.root_hist

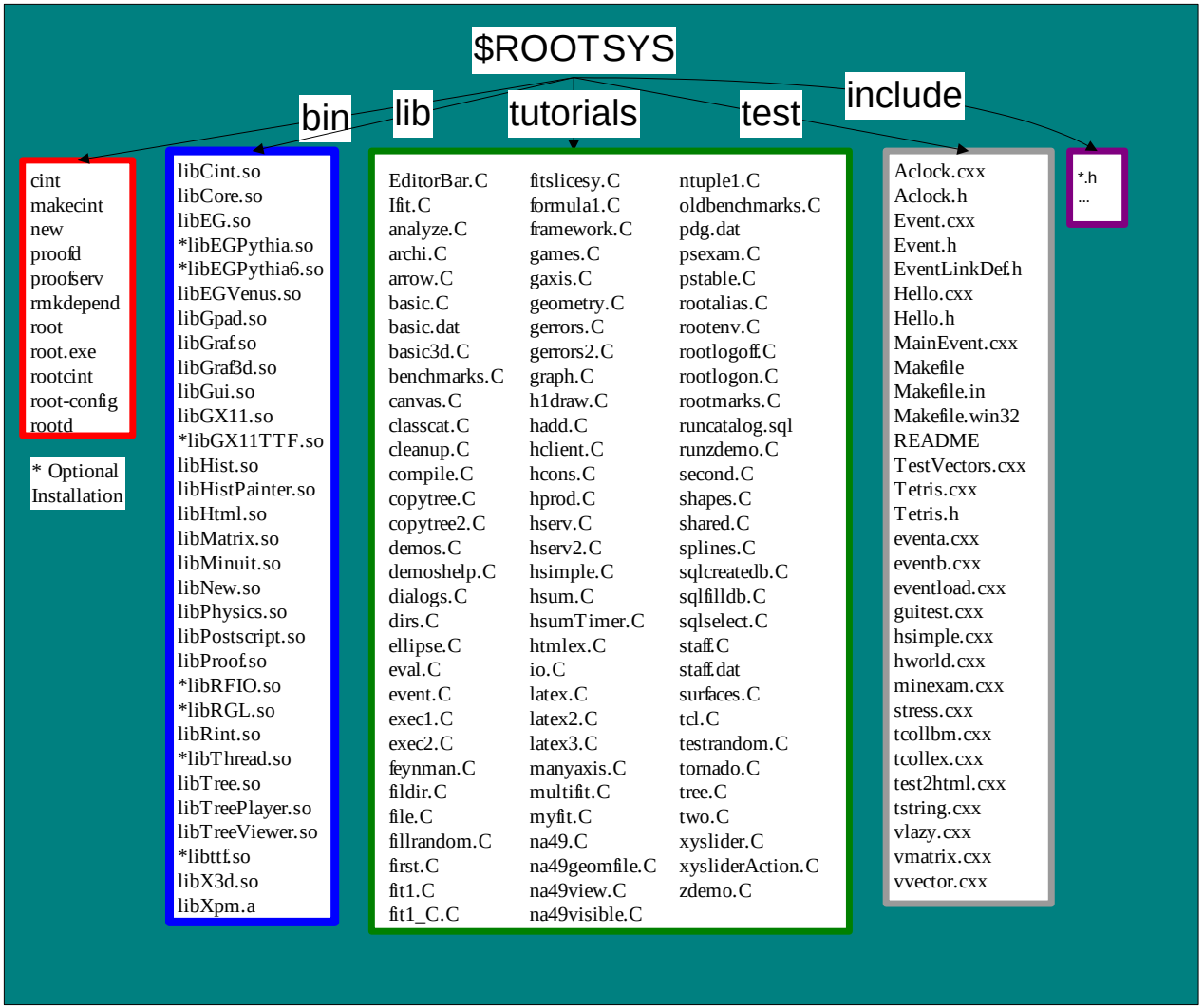
Object Oriented Concepts

- Class: the description of a “thing” in the system
- Object: instance of a class
- Methods: functions for a class

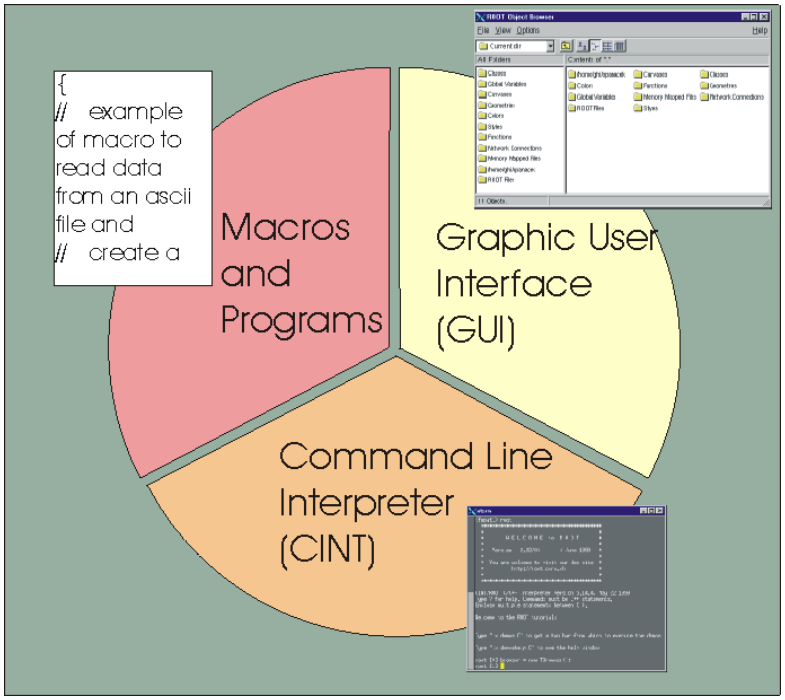
- Members: a “has a” relationship to the class.
- Inheritance: an “is a” relationship to the class.



The Framework Organization



User Interfaces




```
corvair.phys.vt.edu — ssh — 80×24
corvair:~> setenv ROOTSYS /cern/root
corvair:~> setenv LD_LIBRARY_PATH $ROOTSYS/lib:$LD_LIBRARY_PATH
corvair:~> setenv PATH $ROOTSYS/bin:$PATH
corvair:~> root
  ******************************************
  *                                        *
  *       W E L C O M E  to  R O O T       *
  *                                        *
  *   Version   5.16/00      27 June 2007   *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

Compiled on 29 June 2007 for linux with thread support.

CINT/ROOT C/C++ Interpreter version 5.16.21, June 22, 2007
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
```

.q	**Quit**
.L macro.C	Load a macro file
.x macro.C	Load and execute a macro file
.x macro.C++	Compile and execute

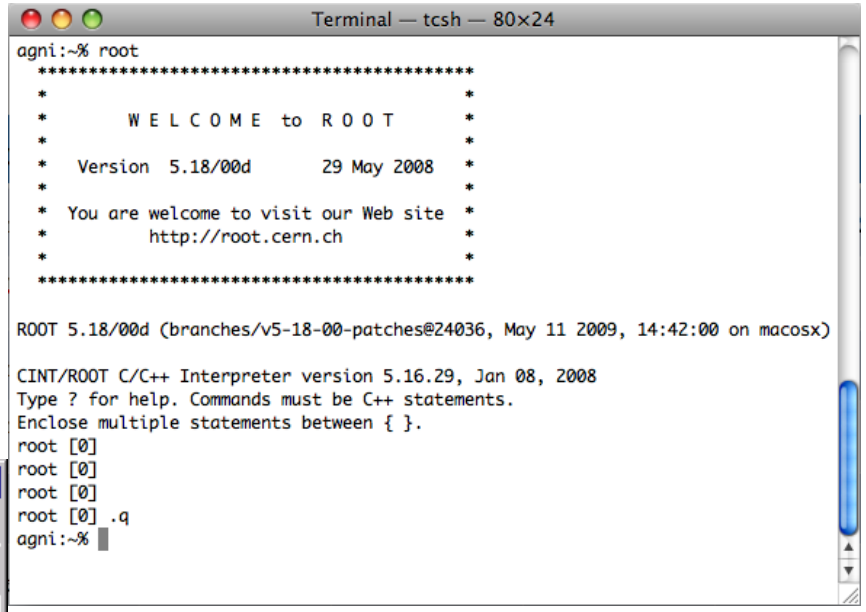
GUI Basics

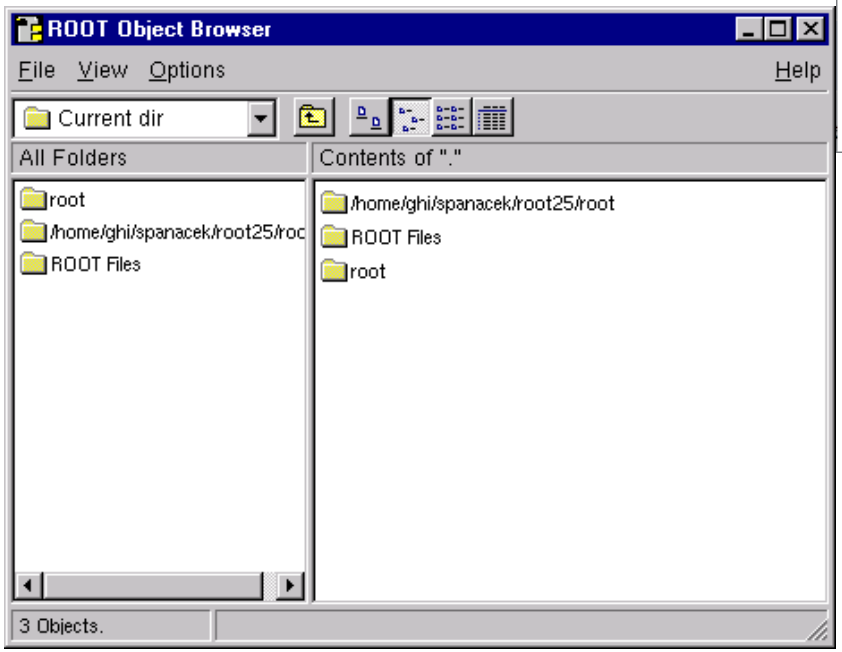
Start root

> root

Quit root (just in case)

root[0]>.q



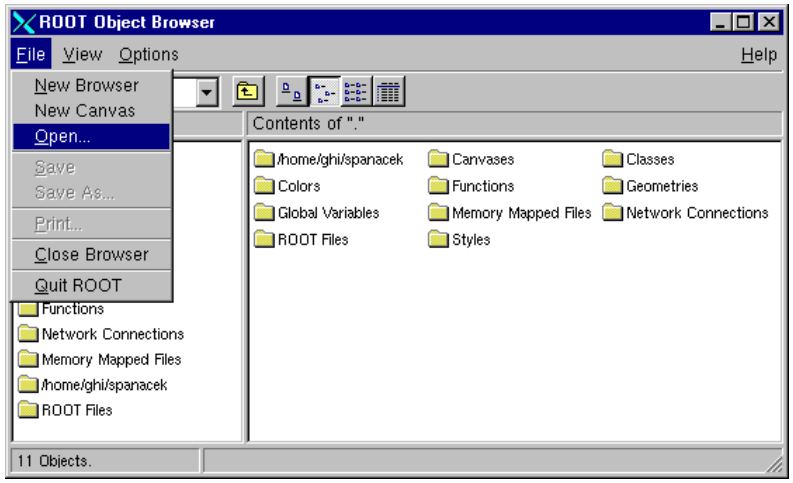


Display the browser

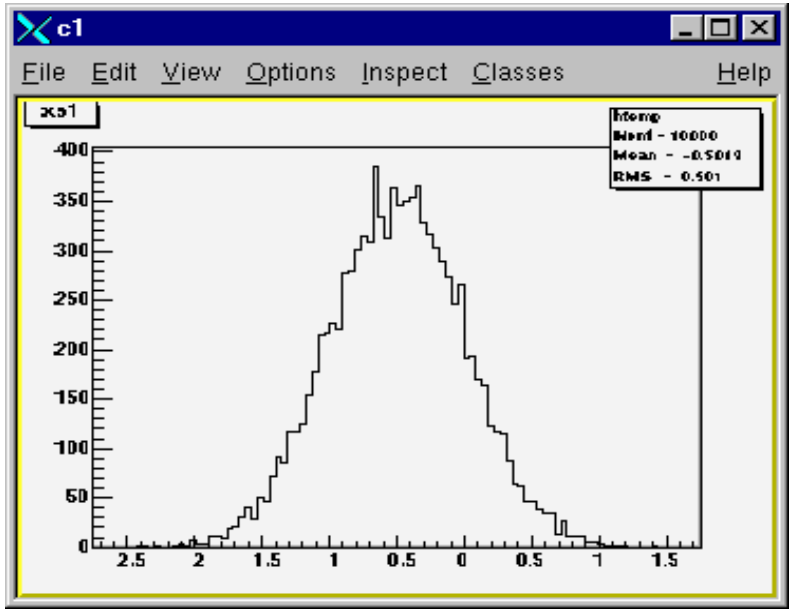
TBrowser b;

Displaying a Histogram

Open the root file
Browse the file

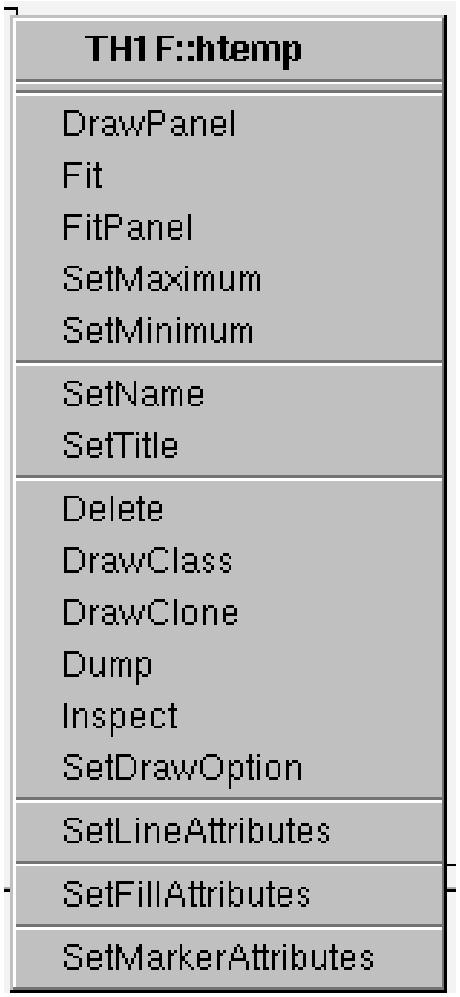


Display a histogram
The Canvas

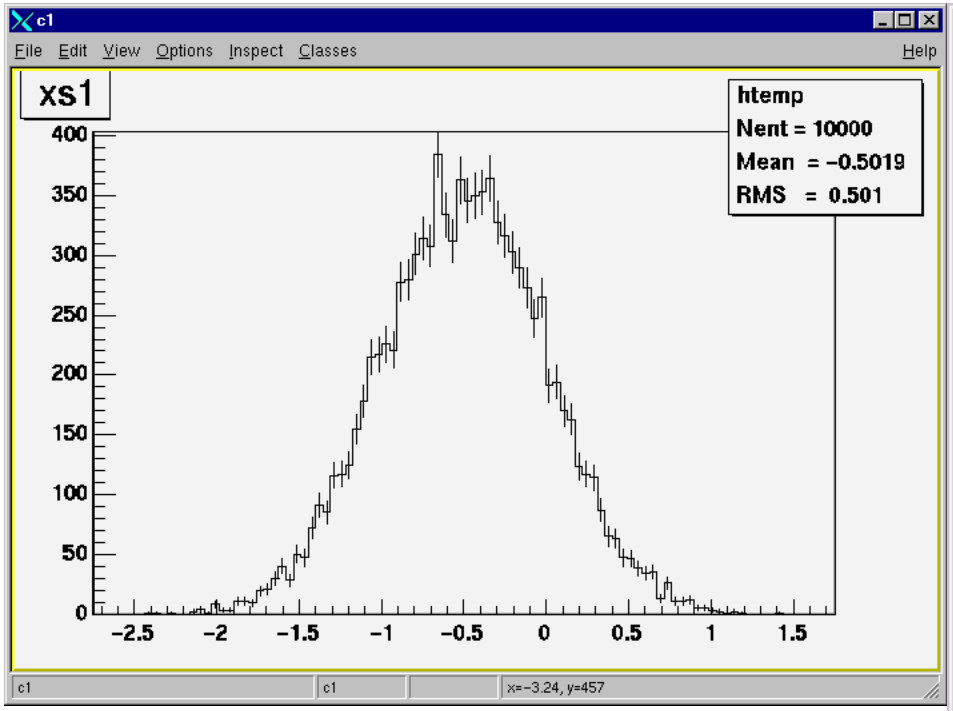


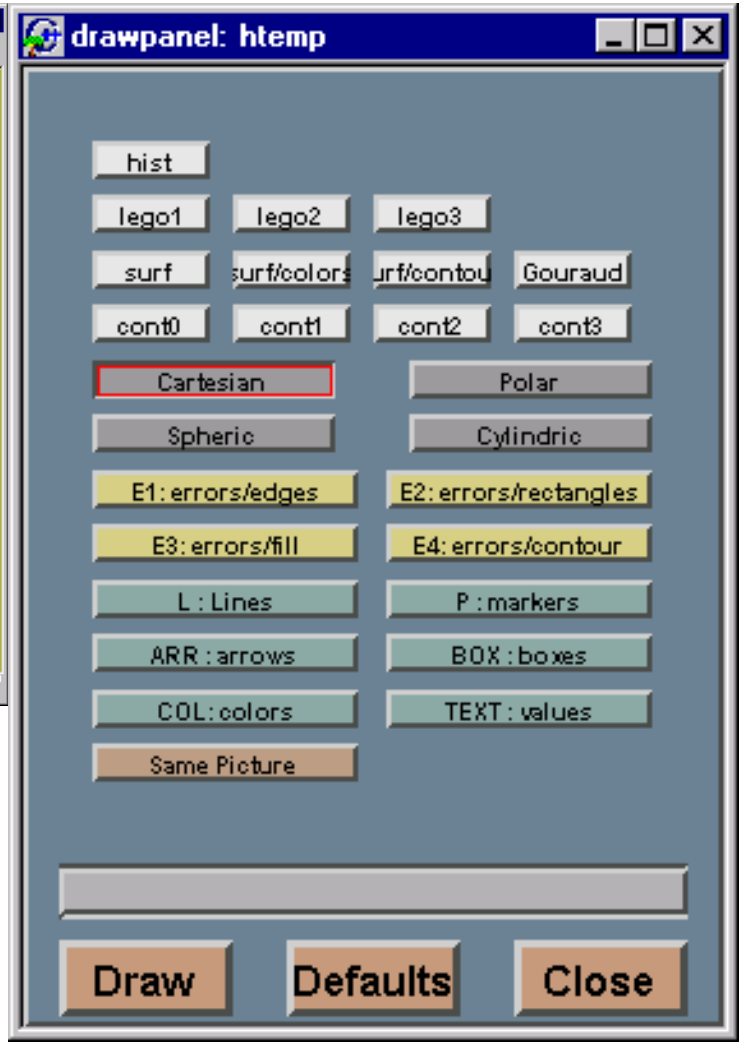
Basic Navigation by Clicking

- Left Click
  - select the object
  - drag the object
  - resize the object
- Right Click
  - context menu
  - class::name
  - methods
- Middle Click
  - activate canvas
  - freezes event status bar

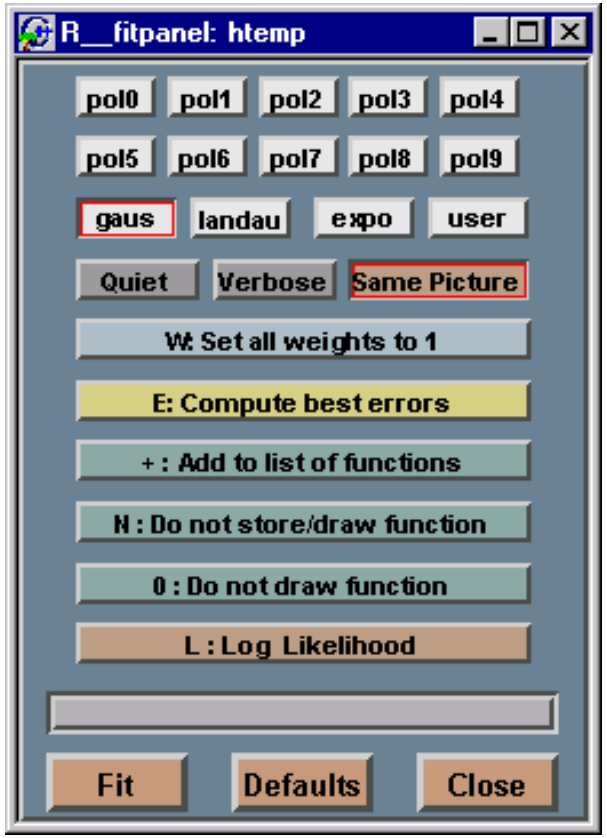


The Draw Panel

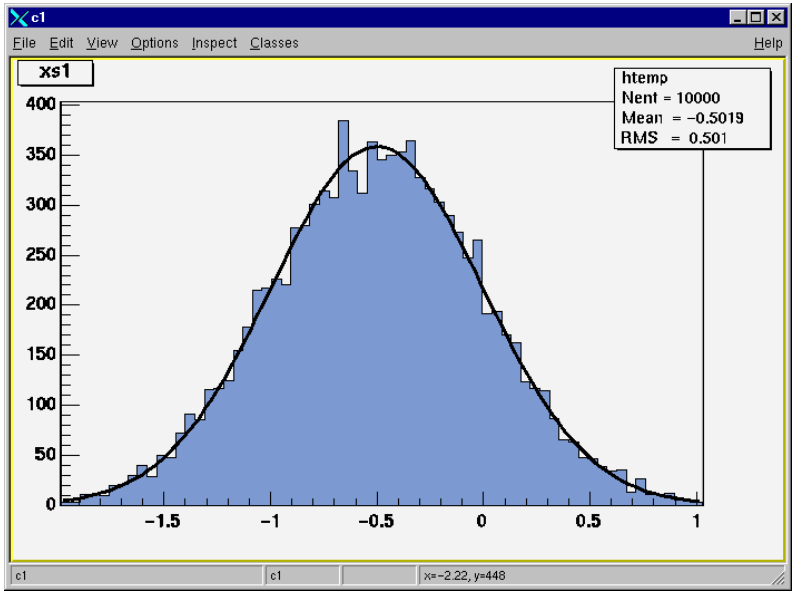




Fitting, Coloring, and Zooming

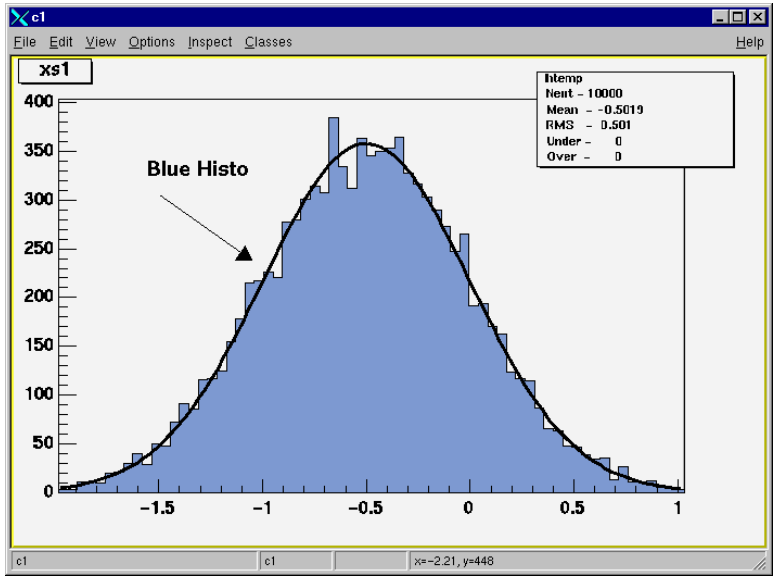


- Adding a gaussian fit
- Coloring the histogram
- Zooming/unzooming

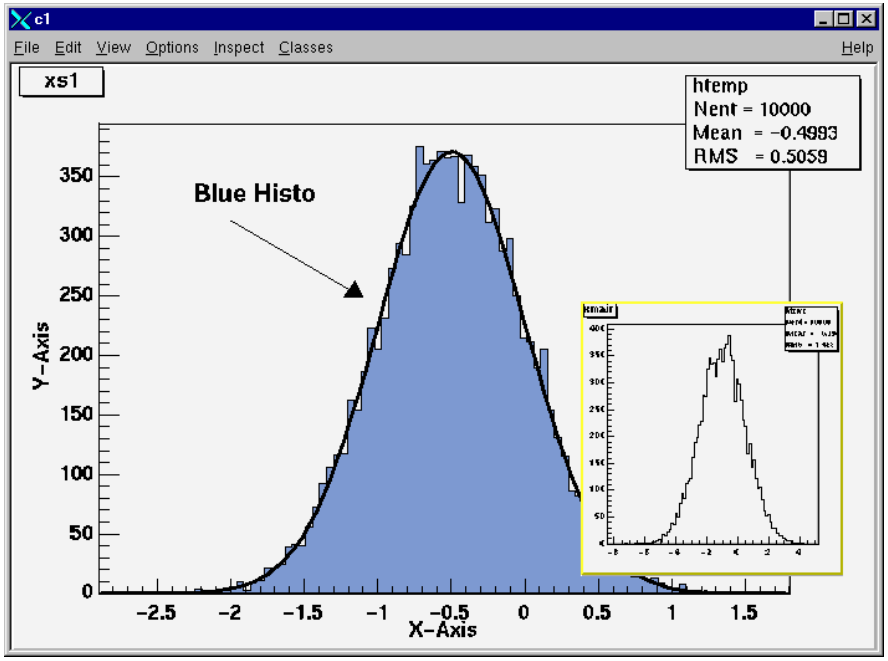


Adding Objects to the Canvas

- The Editor
- Adding an Arrow
- Adding Text

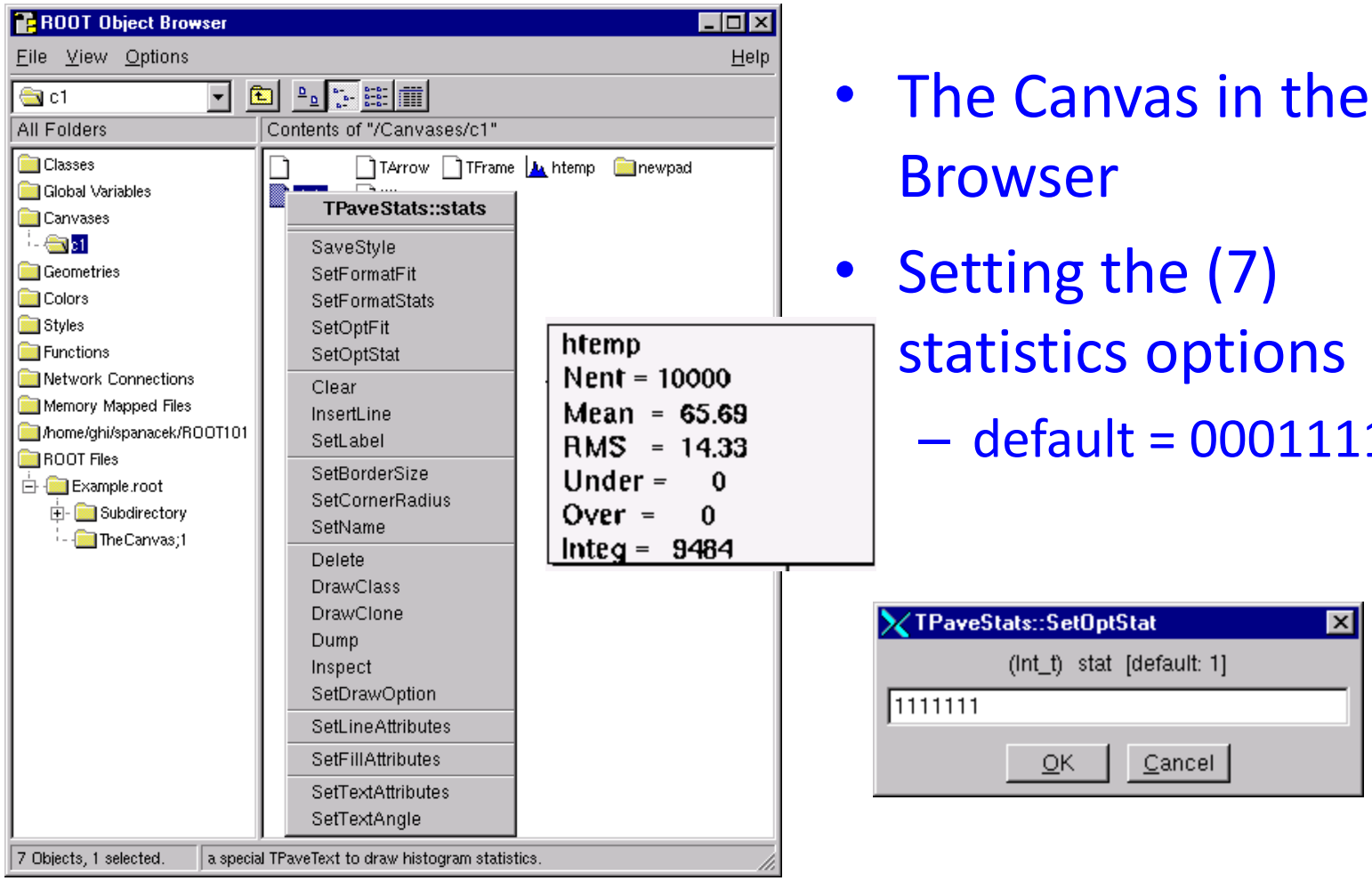


Adding another Pad



- Add a Pad
- Select the new Pad
- Draw a histogram
- Add a title for the axis

Modifying the Statistics



- The Canvas in the Browser
- Setting the (7) statistics options
  - default = 0001111

Reading & Storing Data in Root

- Data can be read from files/database/network
- Data is generally stored as a TTree/TNtuple (similar to a table with rows and columns)
- Each row represents an event
- Each column represents a quantity
- Trees can be created from ASCII files.

Read data from ASCII file to Root File

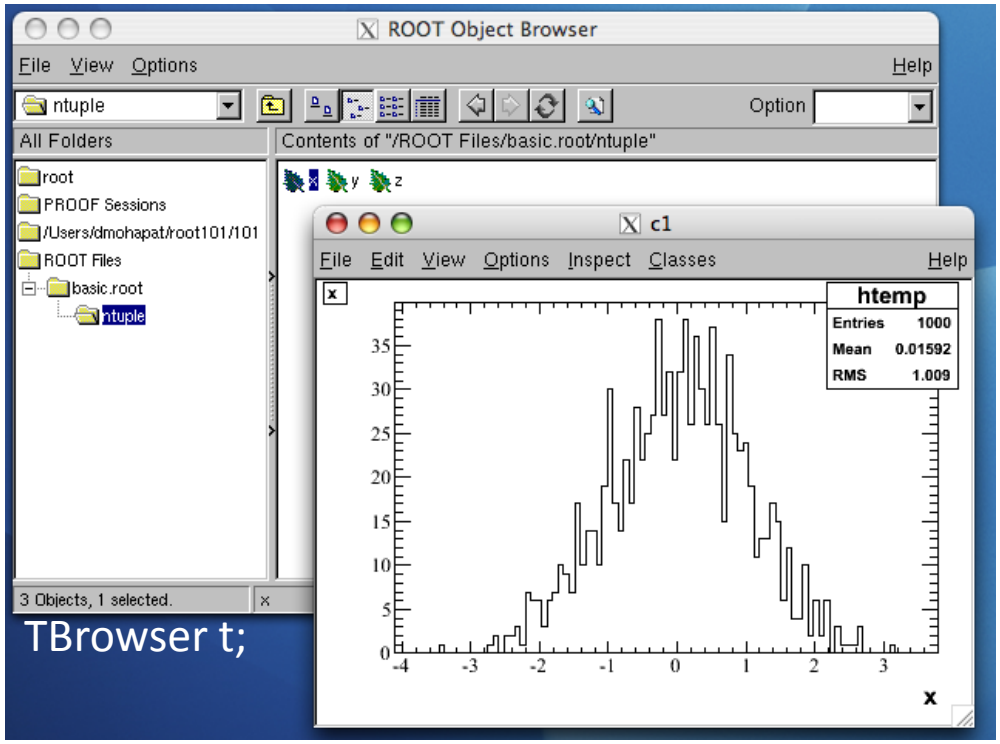
basic.dat

```
-1.102279   -1.799389   4.452822
1.867178    -0.596622   3.842313
-0.524181   1.868521    3.766139
-0.380611   0.969128    1.084074
0.552454    -0.212309   0.350281
-0.184954   1.187305    1.443902
0.205643    -0.770148   0.635417
1.079222    -0.327389   1.271904
-0.274919   -1.721429   3.038899
2.047779    -0.062677   4.197329
-0.458677   -1.443219   2.293266
0.304731    -0.884636   0.875442
-0.712336   -0.222392   0.556881
-0.271866   1.181767    1.470484
0.886202    -0.654106   1.213209
-2.035552   0.527648    4.421883
-1.459047   -0.463998   2.344113
1.230661    -0.005650   1.514559
0.088787    1.885329    3.562347
-0.314154   -0.329161   0.207040
-0.198253   0.646070    0.456712
-1.636217   1.049551    3.778762
1.221109    0.814383    2.154327
1.413135    1.549837    4.398942
-0.174494   -1.330937   1.801841
-1.464173   -0.912864   2.977124
...         ...         ...
```

tree.C

```
#include "Riostream.h"
void tree() {
  ifstream in;
  in.open(Form("basic.dat"));

  Float_t x,y,z;
  Int_t nlines = 0;
  TFile *f = new TFile("basic.root","RECREATE");
  TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
  TNtuple *ntuple = new TNtuple("ntuple","data from
ascii file","x:y:z");

  while (1) {
    in >> x >> y >> z;
    if (!in.good()) break;
    if (nlines < 5) printf("x=%8f, y=%8f, z=%8f\n",x,y,z);
      h1->Fill(x);
      ntuple->Fill(x,y,z);
    nlines++;
  }
  printf(" found %d points\n",nlines);

  in.close();


   f->Write();
}
```

Read the Root file and the Tree



ntuple->Scain();

```
root [2] ntuple->Scan()
************************************************
*    Row   *         x *         y *         z *
************************************************
*        0 * -1.102278 * -1.799389 * 4.4528222 *
*        1 * 1.8671779 * -0.596621 * 3.8423130 *
*        2 * -0.524181 * 1.8685209 * 3.7661390 *
*        3 * -0.380611 * 0.9691280 * 1.0840740 *
*        4 * 0.5524539 * -0.212309 *  0.350281 *
*        5 * -0.184954 * 1.1873049 * 1.4439020 *
*        6 * 0.2056429 * -0.770147 * 0.6354169 *
*        7 * 1.0792219 * -0.327389 * 1.2719039 *
*        8 * -0.274919 * -1.721428 * 3.0388989 *
*        9 * 2.0477790 * -0.062677 * 4.1973290 *
*       10 * -0.458676 * -1.443218 * 2.2932660 *
*       11 * 0.3047310 * -0.884635 * 0.8754420 *
*       12 * -0.712336 * -0.222391 * 0.5568810 *
*       13 * -0.271865 * 1.1817669 * 1.4704840 *
*       14 * 0.8862019 * -0.654106 * 1.2132090 *
*       15 * -2.035552 * 0.5276479 * 4.4218831 *
*       16 * -1.459046 * -0.463997 * 2.3441131 *
*       17 * 1.2306610 * -0.005650 * 1.5145590 *
*       18 * 0.0887869 * 1.8853290 * 3.5623469 *
*       19 * -0.314153 * -0.329160 * 0.2070399 *
*       20 * -0.198253 * 0.6460700 * 0.4567120 *
*       21 * -1.636217 * 1.0495510 * 3.7787621 *
*       22 * 1.2211090 * 0.8143829 * 2.1543269 *
*       23 * 1.4131350 * 1.5498369 * 4.3989419 *
*       24 * -0.174493 * -1.330937 * 1.8018410 *
Type <CR> to continue or q to quit ==>
```

nTuple->Print();

```
root [3] .ls
TFile**         basic.root
 TFile*         basic.root
  KEY: TH1F     h1;1    x distribution
  KEY: TNtuple  ntuple;1        data from ascii file
root [4] ntuple->Print()
******************************************************************************
*Tree    :ntuple    : data from ascii file                                     *
*Entries :     1000 : Total =           14144 bytes  File  Size =      11792 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :x         :                                                        *
*Entries :     1000 : Total  Size=       4604 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :y         :                                                        *
*Entries :     1000 : Total  Size=       4604 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :z         :                                                        *
*Entries :     1000 : Total  Size=       4604 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
```

What do we learn from this Macro

How to	Commands

Create a Root file ?	TFile *f = new TFile("basic.root","RECREATE");
	//option: NEW, CREATE, RECREATE, UPDATE, or READ
	//Book and fill histograms and trees
	//------
	f->Write(); //write the file
	f->Close(); //close the file

Book and fill a histogram ?	TH1F *h1 = new TH1F("h1","x distribution",100,-4,4); /*do some calculation and get the parameter that you want to fill*/ h1->Fill(x);
Book and fill a tree ?	TNtuple *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z"); /*do some calculation and get the parameter that you want to fill*/ ntuple->Fill(x,y,z);
CINT Data types	Int_t and Float_t (see http://root.cern.ch/root/html/ListOfTypes.html)

Root Data Types

C++ type	Size (bytes)	ROOT types	Size (bytes)	FORTRAN analog
(unsigned)char	1	(U)Char_t	1	CHARACTER*1
(unsigned)short (int)	2	(U)Short_t	2	INTEGER*2
(unsigned)int	2 or 4	(U)Int_t	4	INTEGER*4
(unsigned)long (int)	4 or 8	(U)Long_t	8	INTEGER*8
float	4	Float_t	4	REAL*4
double	8 (=4)	Double_t	8	REAL*8
long double	16 (= double)			REAL*16

How to Compile a Macro ?

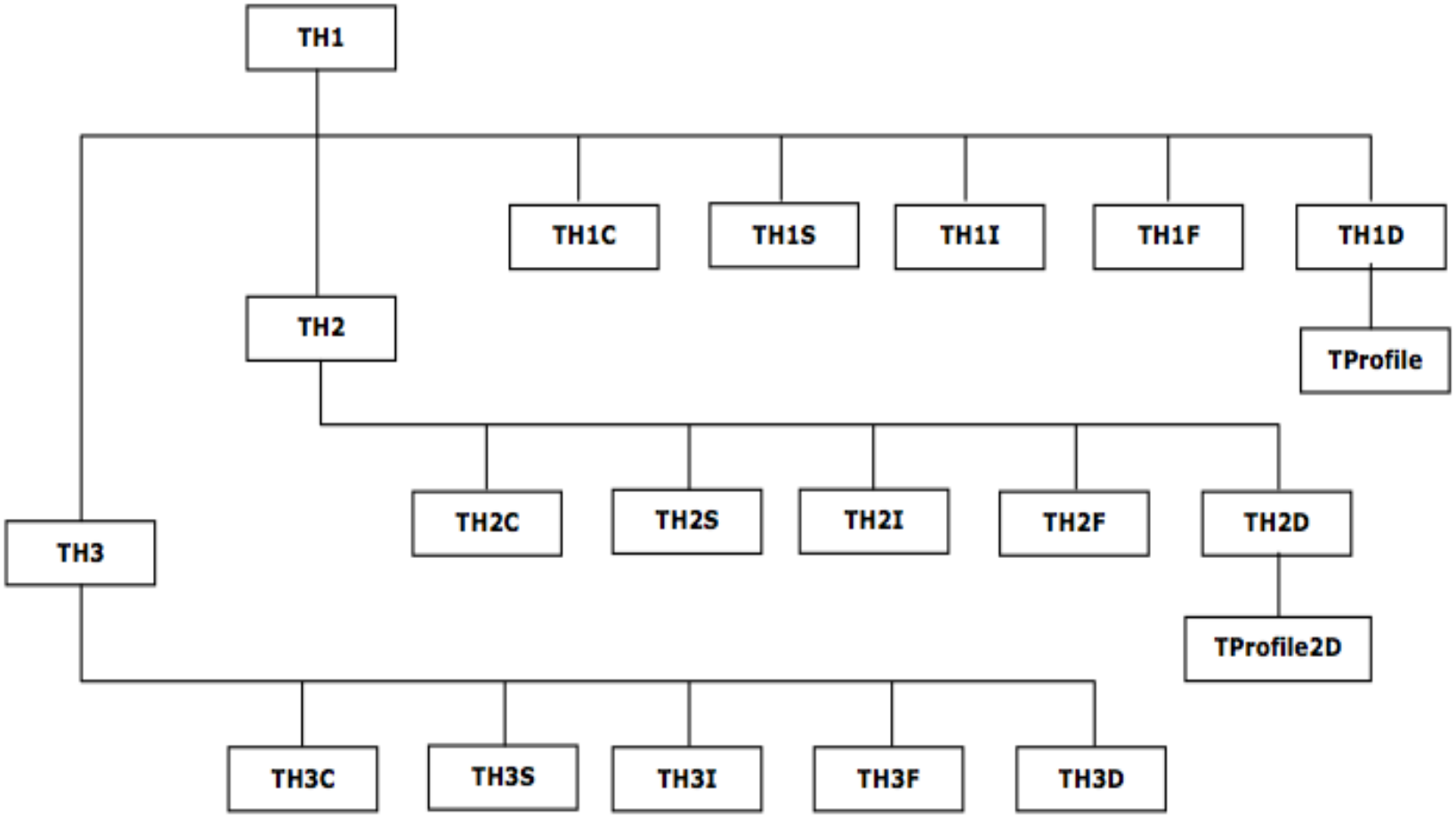
Running the macro
root [0] .x tree.C or,
root [0] .L tree.C
root [1] tree()

```
#include "Riostream.h"
void tree(){
  ifstream in;
  in.open(Form("basic.dat"));

  Float_t x,y,z;
  Int_t nlines = 0;
  TFile *f = new TFile("basic.root","RECREATE");
  TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
  TNtuple *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z");

  while (1) {
    in >> x >> y >> z;
    if (!in.good()) break;
    if (nlines < 5) printf("x=%8f, y=%8f, z=%8f\n",x,y,z);
    h1->Fill(x);
    ntuple->Fill(x,y,z);
    nlines++;
  }
  printf(" found %d points\n",nlines);

  in.close();
  f->Write();
  f->Close();

}
```

Compiling the macro
root [0] .L tree1.C++
root [1] main()

```
#include "Riostream.h"
#include "TFile.h"
#include "TH1.h"
#include "TNtuple.h"

int main() {
  ifstream in;
  in.open(Form("basic.dat"));

  Float_t x,y,z;
  Int_t nlines = 0;
  TFile *f = new TFile("basic.root","RECREATE");
  TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
  TNtuple *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z");

  while (1) {
    in >> x >> y >> z;
    if (!in.good()) break;
    if (nlines < 5) printf("x=%8f, y=%8f, z=%8f\n",x,y,z);
    h1->Fill(x);
    ntuple->Fill(x,y,z);
    nlines++;
  }
  printf(" found %d points\n",nlines);

  in.close();
  f->Write();
  f->Close();
  return 0;
}
```

Histograms in Root: 1D, 2D and 3D



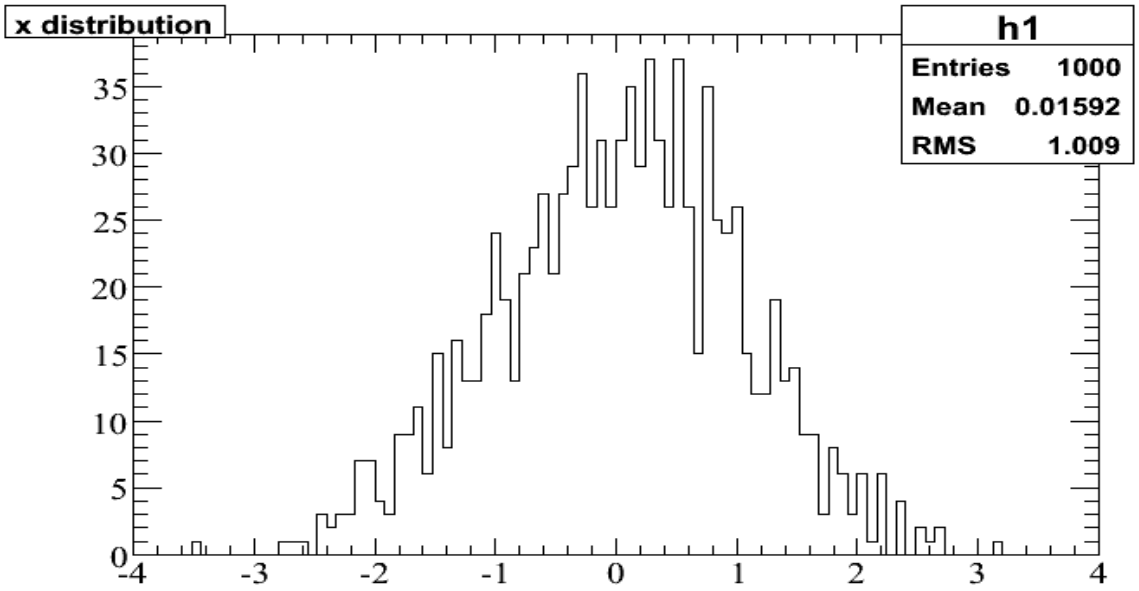
- ✓Floats: Max bin content – 7 digits
- ✓Double: Max bin content – 14 digits

1D Histograms: TH1

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

Example:

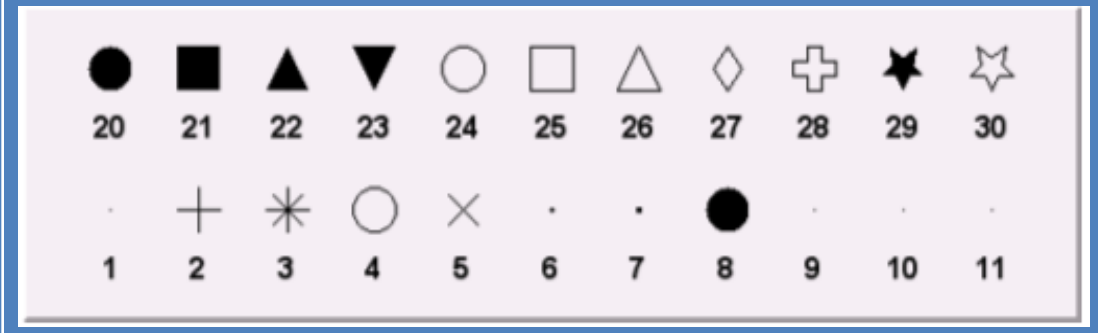
- TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
- h1->Fill(x);
- h1->Draw();



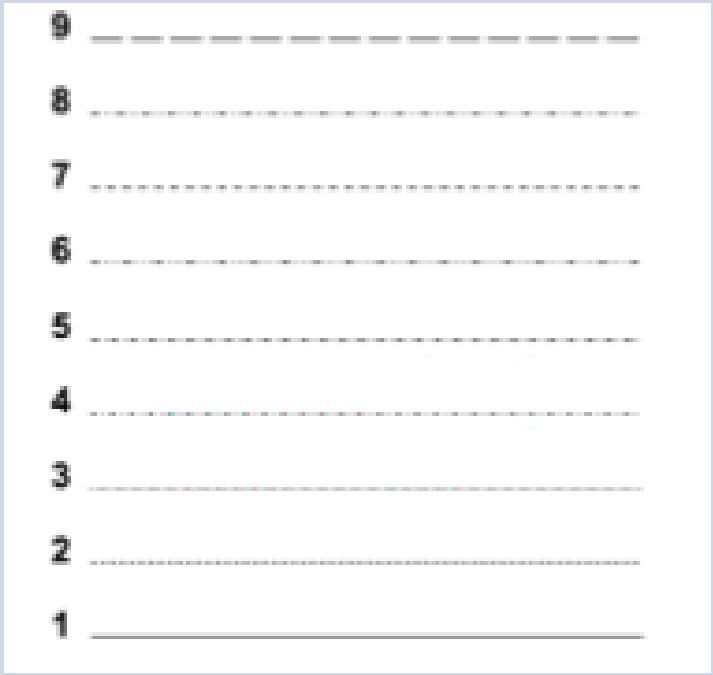
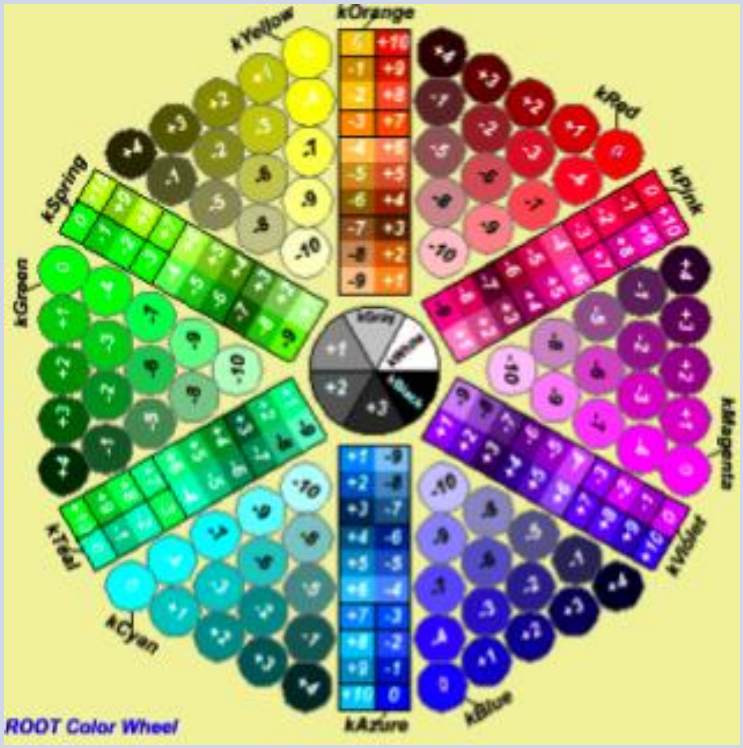
Histogram Properties

Command	Parameters
h1.GetMean()	Mean
h1.GetRMS()	Root of Variance
h1.GetMaximum();	Maximum bin content
h1.GetMaximumBin(int bin_number);	location of maximum
h1.GetBinCenter(int bin_number);	Center of bin
h1.GetBinContent(int bin_number);	Content of bin

Histogram Cosmetics

h1.SetMarkerStyle();	
h1.SetFillColor();	

Histogram cosmetics: Lines

LineStyle h1->SetLineStyle();	LineColor h1.SetLineColor();
	

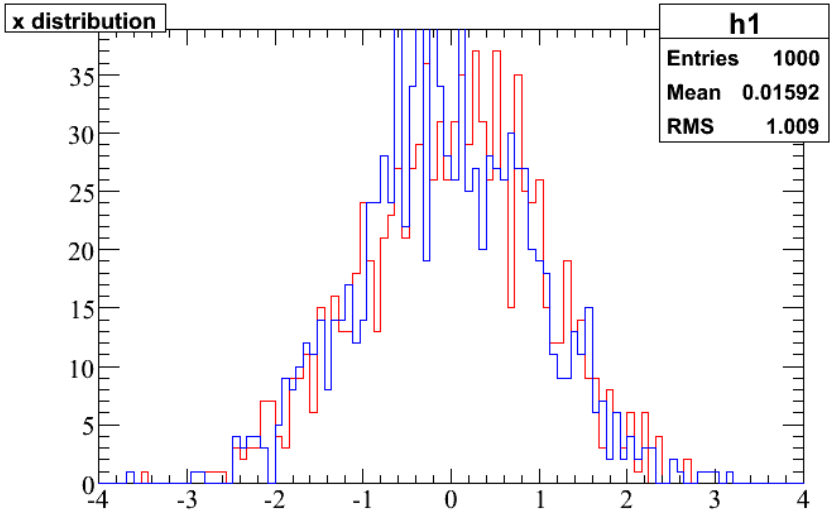
1D Histogram

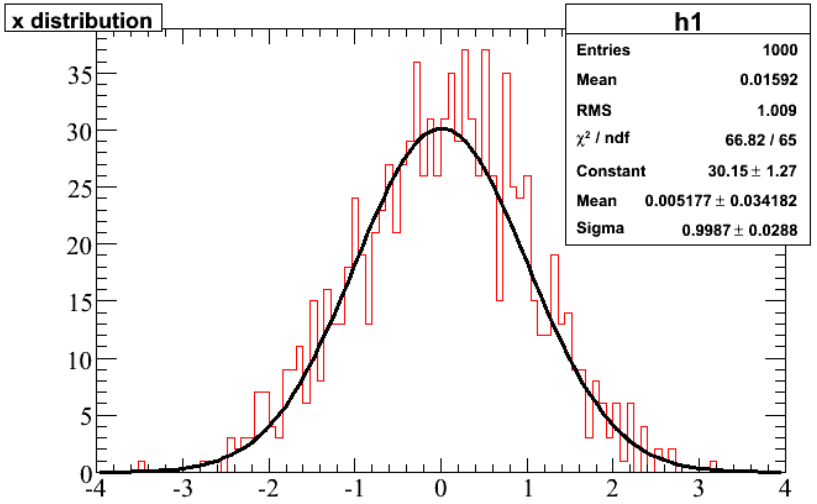
Overlapping

- h1->Draw();
- h2->Draw("same");

Fit (will be covered in detail later)

- h1->Fit("gaus");





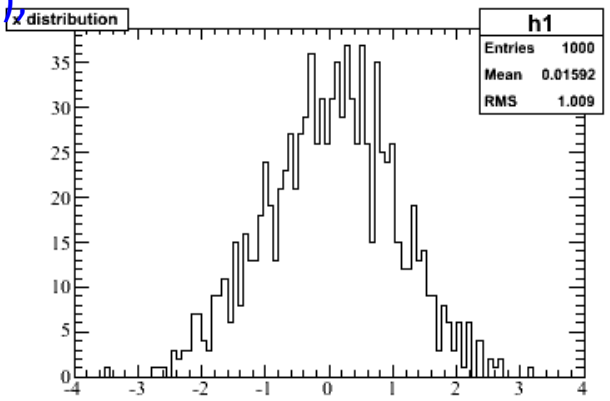
Canvas: an area mapped to a window

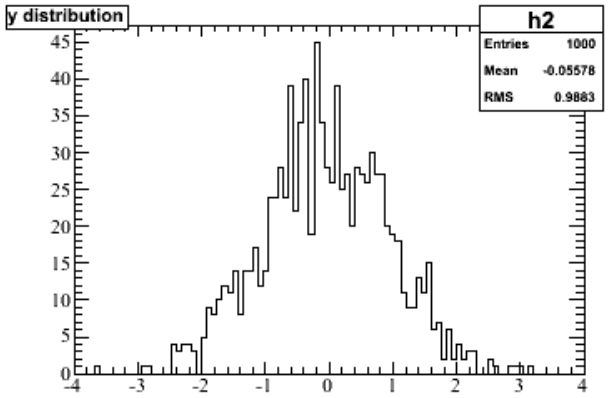
Command	Action
c1 = new TCanvas("c1","Title, w, h)	Creates a new canvas with width equal to w number of pixels and height equal to h number of pixels.
c1->Divide(2,2);	Divides the canvas to 4 pads.
c1->cd(3)	Select the 3rd Pad

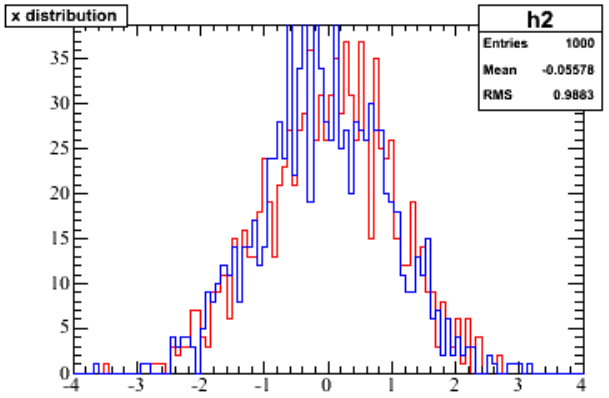
c1->SetGridx();
c1->SetGridy();
c1->SetLogy();
	You can set grid along x and y axis.
	
	You can also set log scale plots.

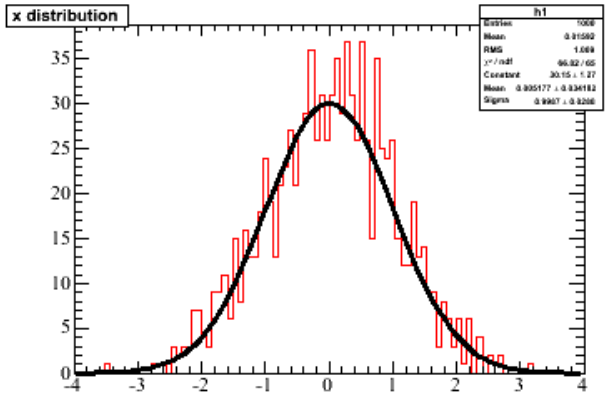
Canvas: Demo…

```
root [1] c1 = new
TCanvas("c1","Title",800,600);
root [2] c1->Divide(2,2);
root [3] c1->cd(1);
root [4] h1->Draw();
root [5] c1->cd(2);
root [6] h2->Draw();
root [7] c1->cd(3);
root [8] h1->SetLineColor(2)
root [9] h2->SetLineColor(4)
root [10] h1->Draw();
root [11] h2->Draw("same");
root [12] c1->cd(4);
root [13] h1->Fit("gaus");
```







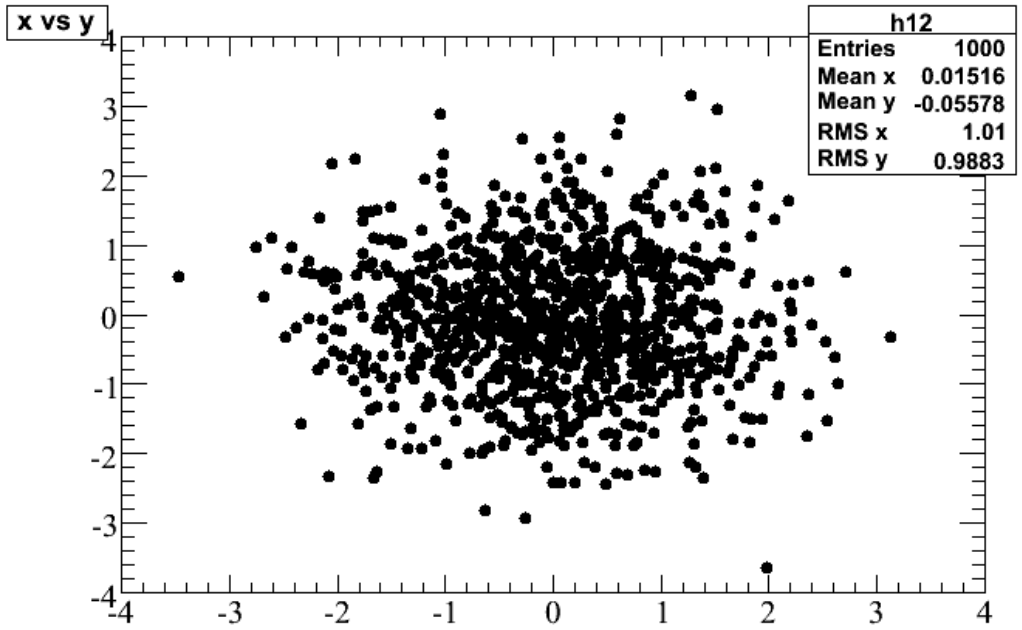


2D Histograms: TH2

- TH2F *name = new TH2F(”name",”Title”, xBins, low xbin, up xbin, yBins, low ybin, up y bin);

Example:

- TH2F *h12 = new TH2F("h12","x vs y",100,-4,4,100, -4, 4);
- h12->Fill(x,y);
- h12->Draw();

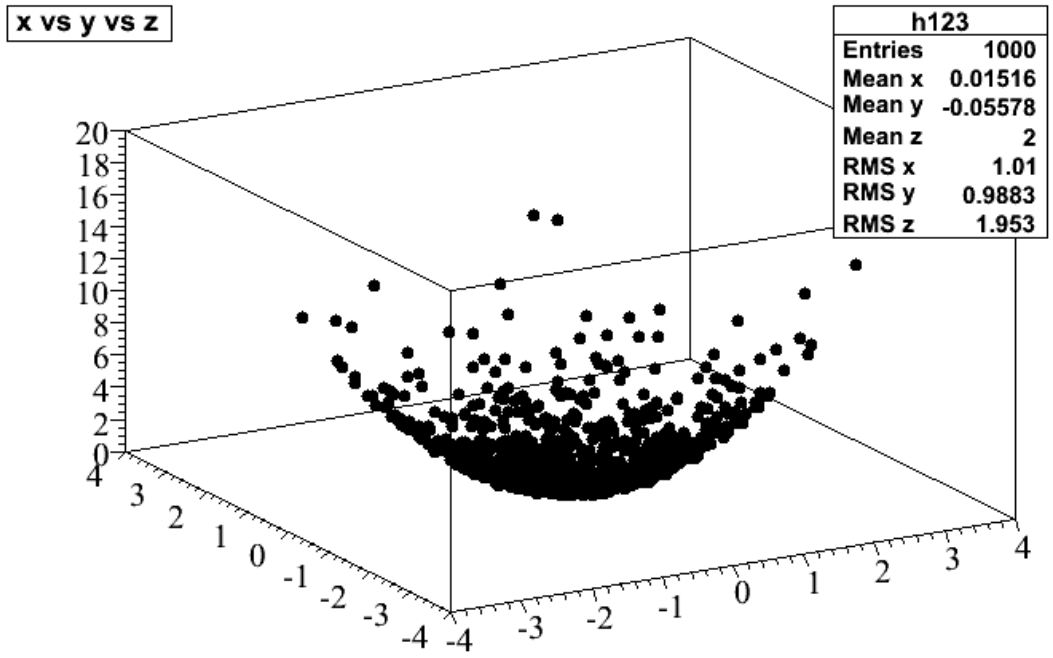


3D Histograms: TH3

- TH3F *name = new TH3F(”name",”Title”, xBins, low xbin, up xbin, yBins, low ybin, up ybin, zBins, low zbin, up zbin);

Example:

- TH3F *h123 = new TH3F("h123","x vs y vs z",100,-4,4,100, -4, 4,100,0,20);
- h123->Fill(x,y,z);
- h123->Draw();



Histogram Drawing Options

" SAME": Superimpose on previous picture in the same pad.
" CYL": Use cylindrical coordinates.
" POL": Use polar coordinates.
" SPH": Use spherical coordinates.
" PSR": Use pseudo-rapidity/phi coordinates.
" LEGO": Draw a lego plot with hidden line removal.
" LEGO1": Draw a lego plot with hidden surface removal.
" LEGO2": Draw a lego plot using colors to show the cell contents.
" SURF": Draw a surface plot with hidden line removal.
" SURF1": Draw a surface plot with hidden surface removal.
" SURF2": Draw a surface plot using colors to show the cell contents.
" SURF3": Same as SURF with a contour view on the top.
" SURF4": Draw a surface plot using Gouraud shading.
“ SURF5”: Same as SURF3 but only the colored contour is drawn.

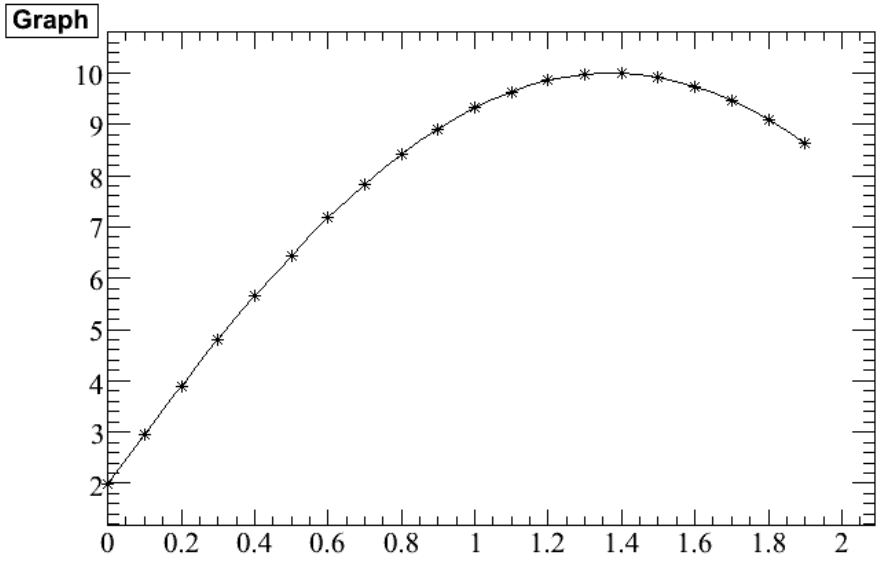
Note: Please check chapter 3 in user’s guide to learn more about options.

Graphs

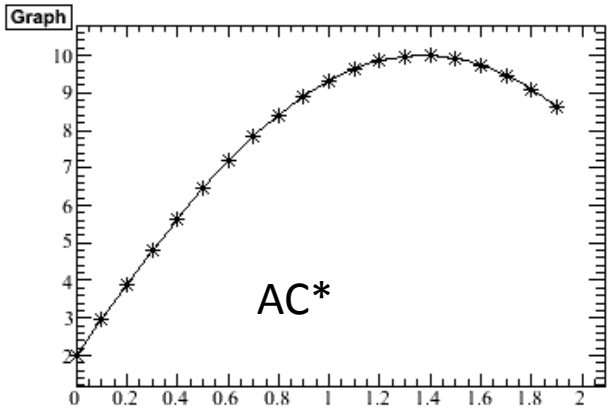
Graphics object made of two arrays X and Y, holding the x, y coordinates of n points

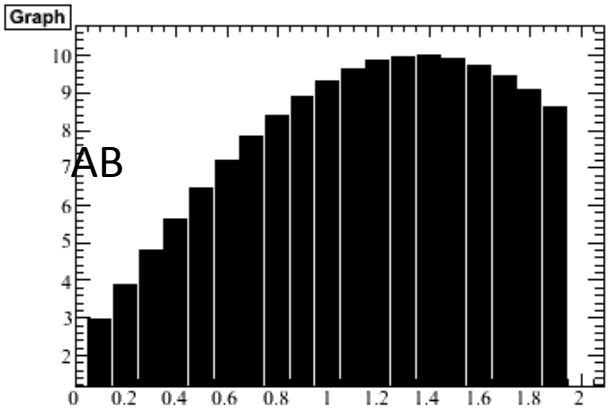
Graphs:

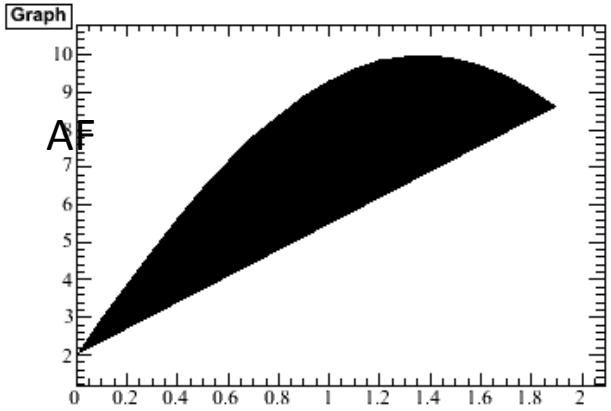
- Int_t n = 20;
- Double_t x[n], y[n];
- for (Int_t i=0; i<n; i++){
- x[i] = i*0.1;
- y[i] = 10*sin(x[i]+0.2); }
- TGraph *gr1 = new TGraph (n, x, y);
- gr1->Draw("AC*");

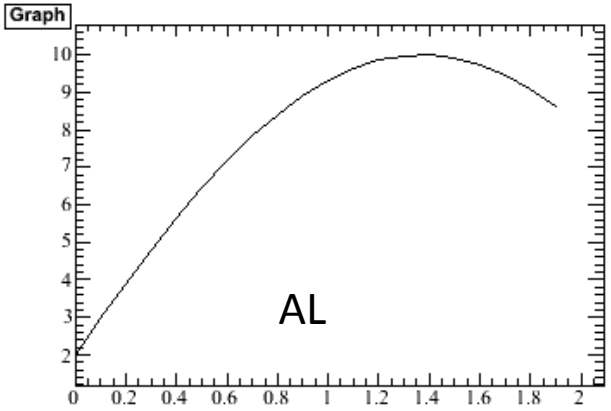


Graph Drawing Options

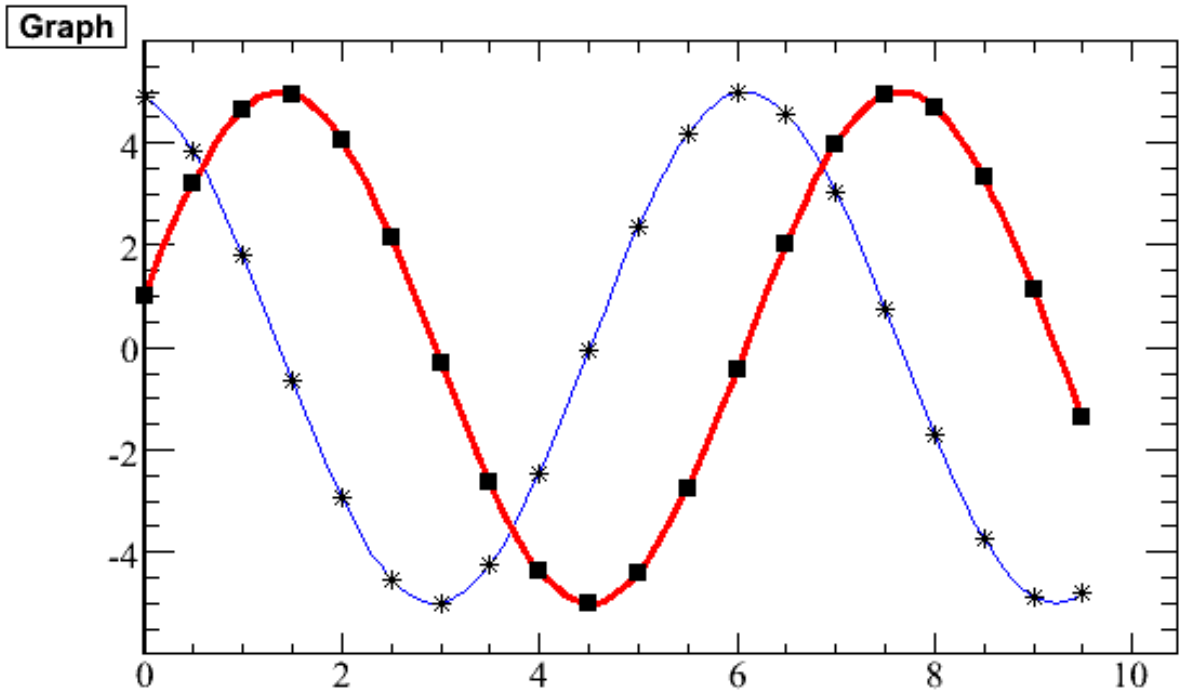








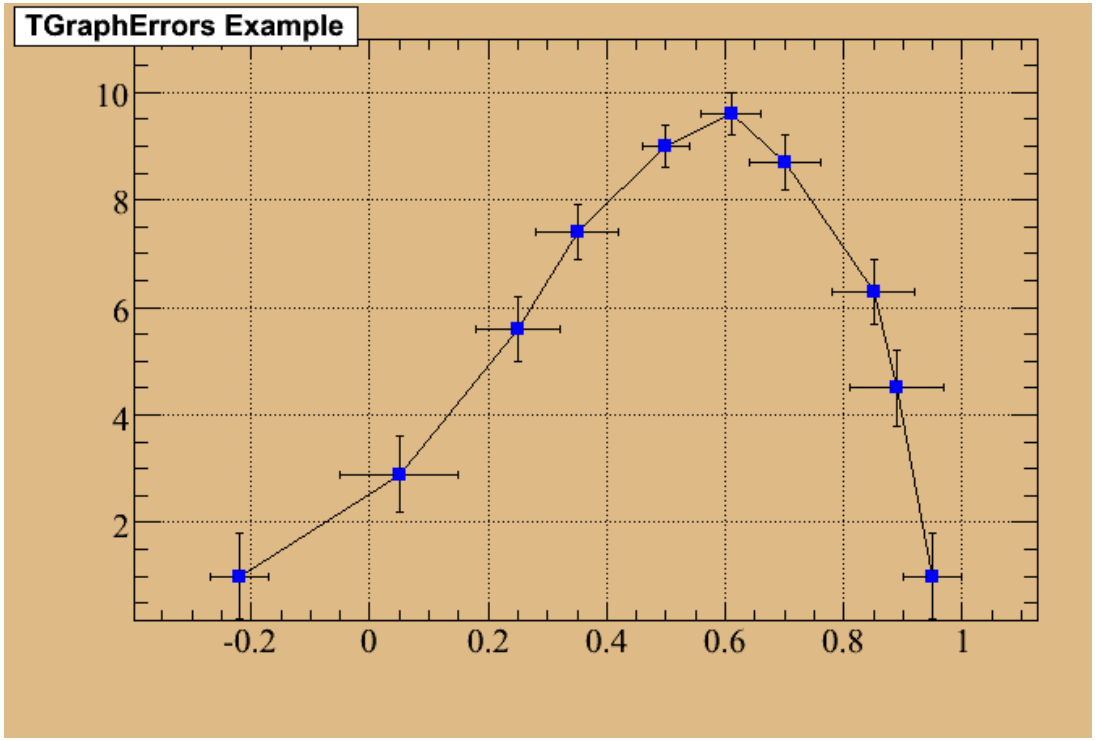
Superimpose two Graphs



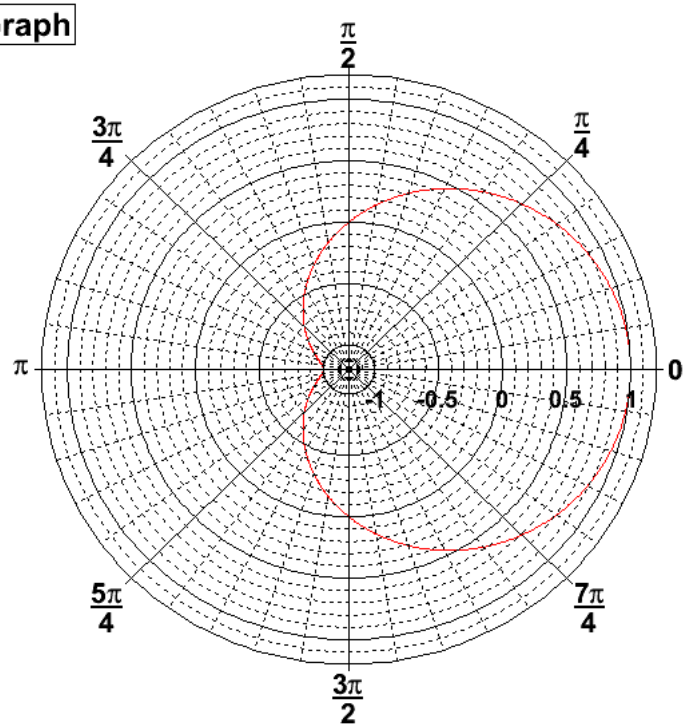
- TGraph *gr1 = new TGraph(n,x,y);
- TGraph *gr2 = new TGraph(n,x1,y1);
- gr1->SetLineColor(4);
- gr1->Draw("AC*");
- gr2->SetLineWidth(3);
- gr2->SetMarkerStyle(21);
- gr2->SetLineColor(2);
- gr2->Draw("CP");

Graph with Error bar

- Float_t x[n] = {-.22,.05,.25,.35,.5,.61,.7,.85,.89,.95};
- Float_t y[n] = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
- Float_t ex[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
- Float_t ey[n] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
- gr = new TGraphErrors(n,x,y,ex,ey);



Polar Graphs



- Generate or calculate “r” and “theta”
- TGraphPolar * grP1 = new TGraphPolar(1000,r,theta);
- grP1->Draw();

TTree

Saving data in a table with rows representing the event and columns representing quantities.

ROOT Tree

- Store large quantities of same-class objects
- TTree class is optimized to reduce disk space and enhance access speed
- TTree can hold all kind of data
- TNtuple is a TTree that is limited to only hold floating-point numbers

If we do not use TTree, we need to

- read each event in its entirety into memory
- extract the parameters from the event
- Compute quantities from the same
- fill a histogram

Create a Root TTree/TNtuple

- Tfile *F = new Tfile(“test.root”,RECREATE);
- TTree *T = new TTree("T","test");
- T->Branch("x",&x,"x/F");
- T->Branch("y",&y,"x/F");
- T->Branch("z",&z,"x/F");

// Read/or calculate x,y and z

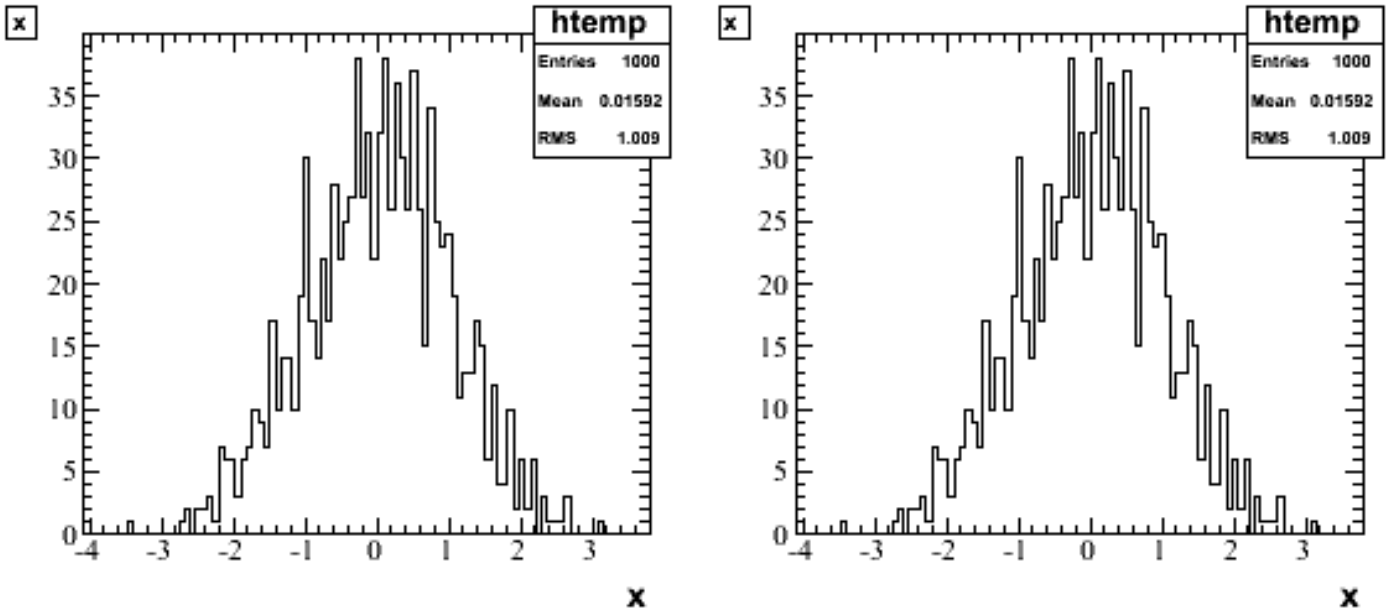
- T->Fill();
- T->Close();
- F->Close();

- Tfile *F = new Tfile(“test.root”,RECREATE);
- TNtuple *T = new TNtuple("ntuple","data from ascii file","x:y:z");

// Read/or calculate x,y and z

- T->Fill(x,y,z);
- T->Close();
- F->Close();

Draw: T->Draw(“x”);



T->Print(); //Print the Root Content

```
root [2] T->Print()
******************************************************************************
*Tree    :T         : test                                                     *
*Entries :     1000 : Total =           14076 bytes  File  Size =      11714 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :x         : x/F                                                      *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :y         : x/F                                                      *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :z         : x/F                                                      *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
```

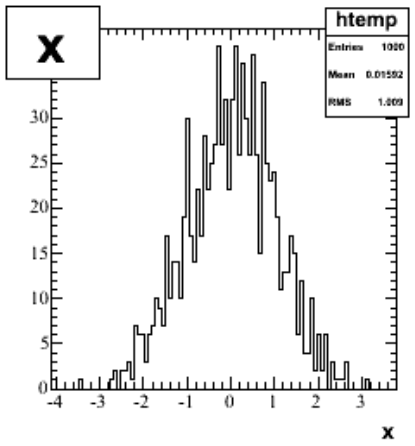
T->Scan();//scan the Root rows and columns

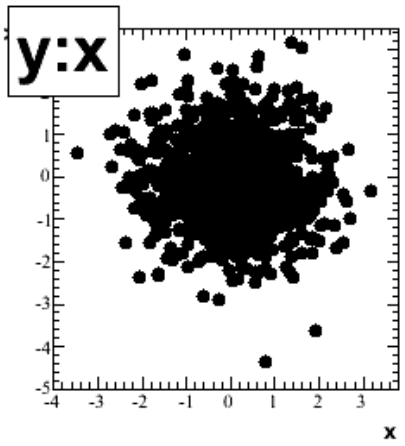
```
root [3] T->Scan()
***************************************************
*    Row   *           x *           x *           x *
***************************************************
*        0 * -1.102278 * -1.102278 * -1.102278 *
*        1 * 1.8671779 * 1.8671779 * 1.8671779 *
*        2 * -0.524181 * -0.524181 * -0.524181 *
*        3 * -0.380611 * -0.380611 * -0.380611 *
*        4 * 0.5524539 * 0.5524539 * 0.5524539 *
*        5 * -0.184954 * -0.184954 * -0.184954 *
*        6 * 0.2056429 * 0.2056429 * 0.2056429 *
*        7 * 1.0792219 * 1.0792219 * 1.0792219 *
*        8 * -0.274919 * -0.274919 * -0.274919 *
*        9 * 2.0477790 * 2.0477790 * 2.0477790 *
*       10 * -0.458676 * -0.458676 * -0.458676 *
*       11 * 0.3047310 * 0.3047310 * 0.3047310 *
*       12 * -0.712336 * -0.712336 * -0.712336 *
*       13 * -0.271865 * -0.271865 * -0.271865 *
*       14 * 0.8862019 * 0.8862019 * 0.8862019 *
*       15 * -2.035552 * -2.035552 * -2.035552 *
*       16 * -1.459046 * -1.459046 * -1.459046 *
*       17 * 1.2306610 * 1.2306610 * 1.2306610 *
*       18 * 0.0887869 * 0.0887869 * 0.0887869 *
*       19 * -0.314153 * -0.314153 * -0.314153 *
*       20 * -0.198253 * -0.198253 * -0.198253 *
*       21 * -1.636217 * -1.636217 * -1.636217 *
*       22 * 1.2211090 * 1.2211090 * 1.2211090 *
*       23 * 1.4131350 * 1.4131350 * 1.4131350 *
*       24 * -0.174493 * -0.174493 * -0.174493 *
```

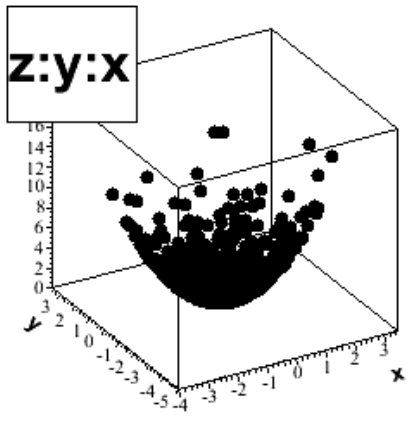
Play with Root Tree

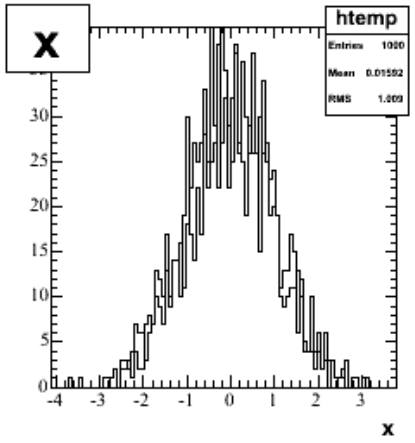
Command	Action
T->Print();	Prints the content of the tree
T->Scan();	Scans the rows and columns
T->Draw("x");	Draw a branch of tree
How to apply cuts: T->Draw("x","x>0"); T->Draw("x","x>0 && y>0");	 Draw "x" when "x>0" Draw "x" when both x >0 and y >0
T->Draw("y"," ","same");	Superimpose "y" on "x"
T->Draw("y:x");	Make "y vs x" 2d scatter plot
T->Draw("z:y:x");	Make "z:y:x" 3d plot
T->Draw("sqrt(x*x+y*y)");	Plot calculated quantity
T->Draw("x>>h1");	Dump a root branch to a histogram

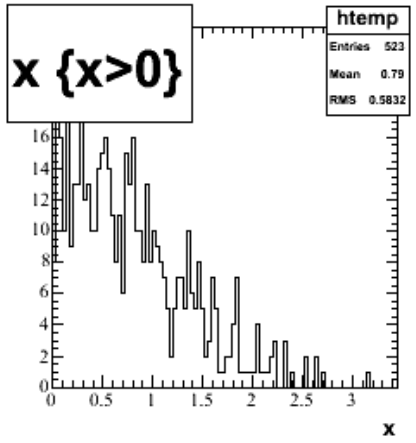
Play with Root Tree

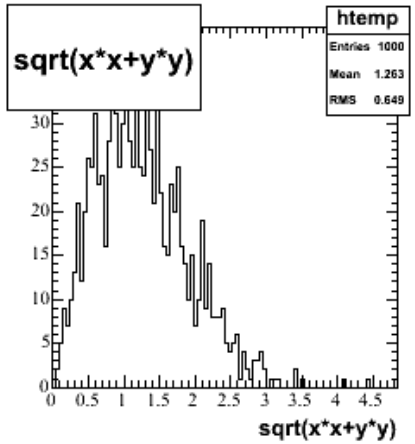






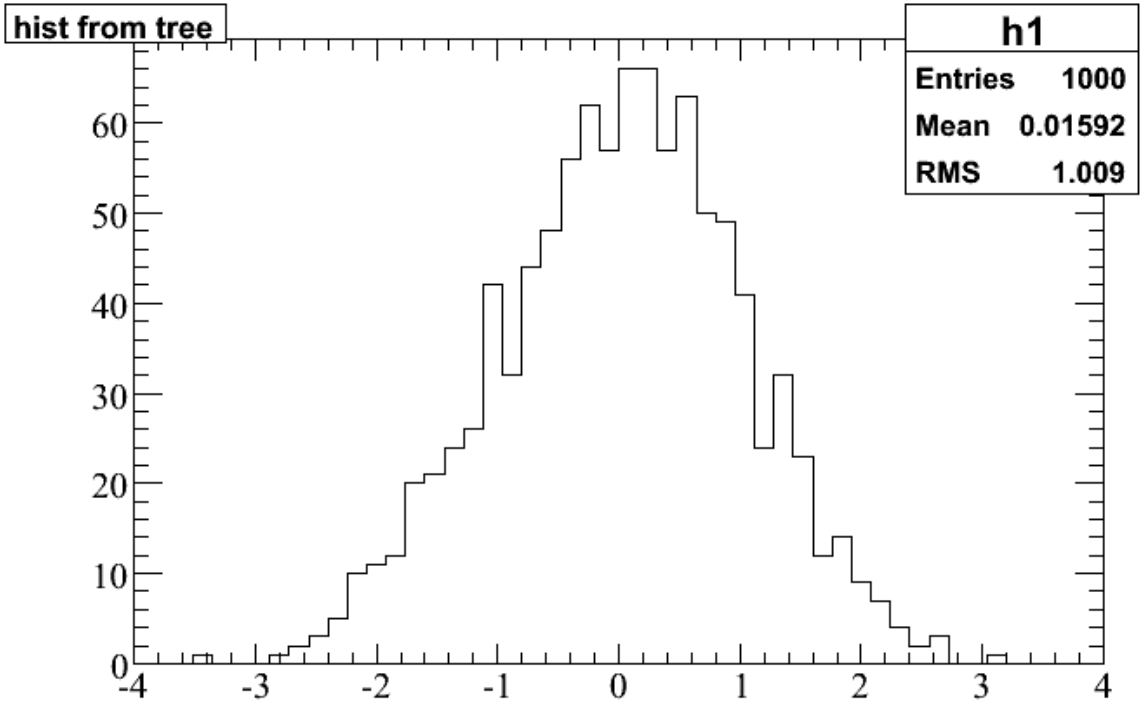






Create Histogram from Root Tree

- root [2] TH1F *h1 = new TH1F("h1","hist from tree",50, -4, 4);
- root [3] T->Draw("x>>h1");



How to deal with number of large Root files with same trees ?

- TChain chain("T"); // name of the tree is the argument
- chain.Add("file1.root");
- chain.Add("file2.root");
- chain.Add("file3.root”);

You can draw “x” from all the files in the chain at the same time

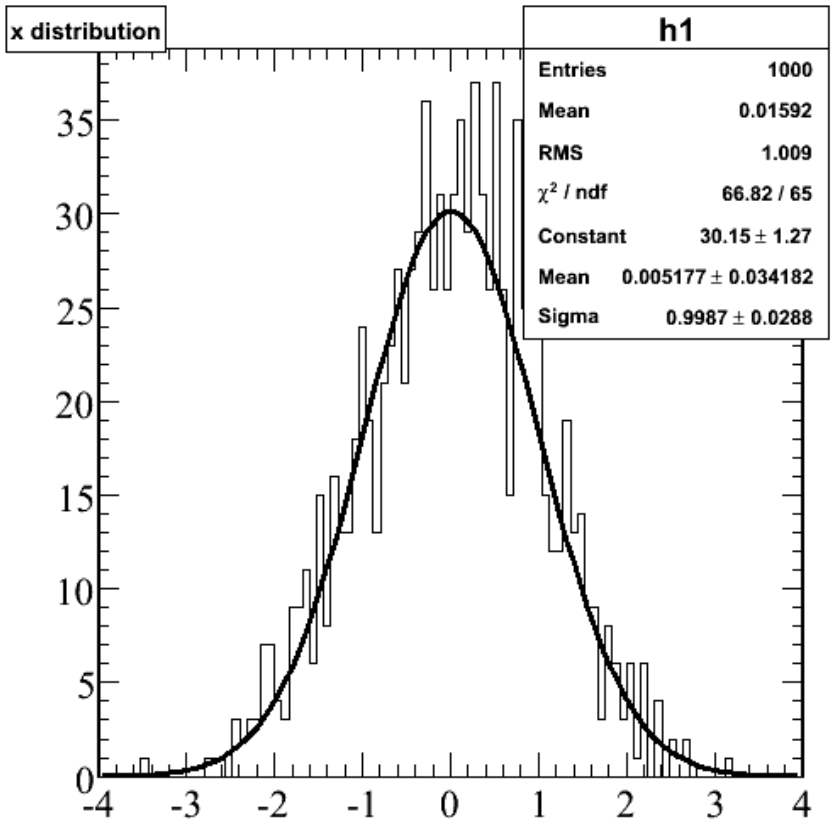
- chain.Draw("x");

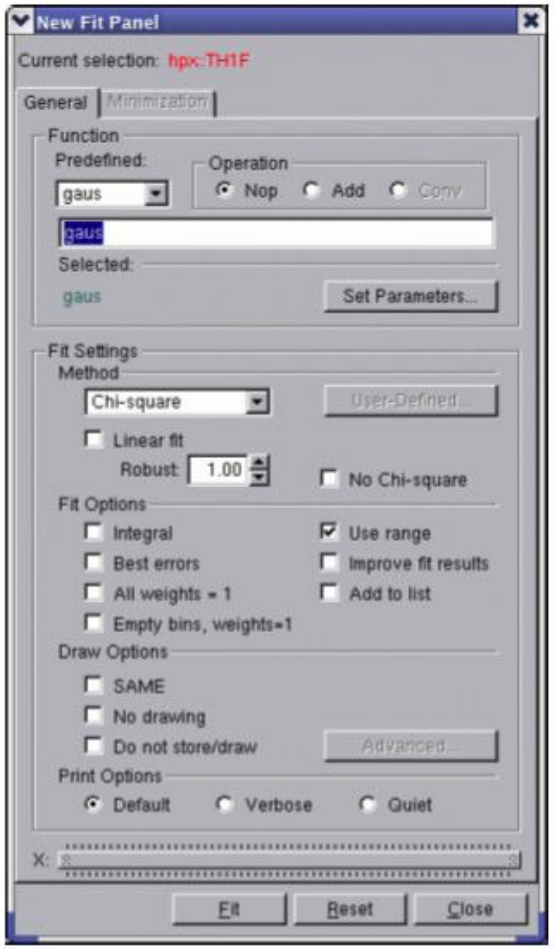
Fitting Histograms in Root

Histograms (1-D,2-D,3-D and Profiles) can be fitted with a user specified function via TH1::Fit.

It uses MINUIT as the minimization routine for fitting,

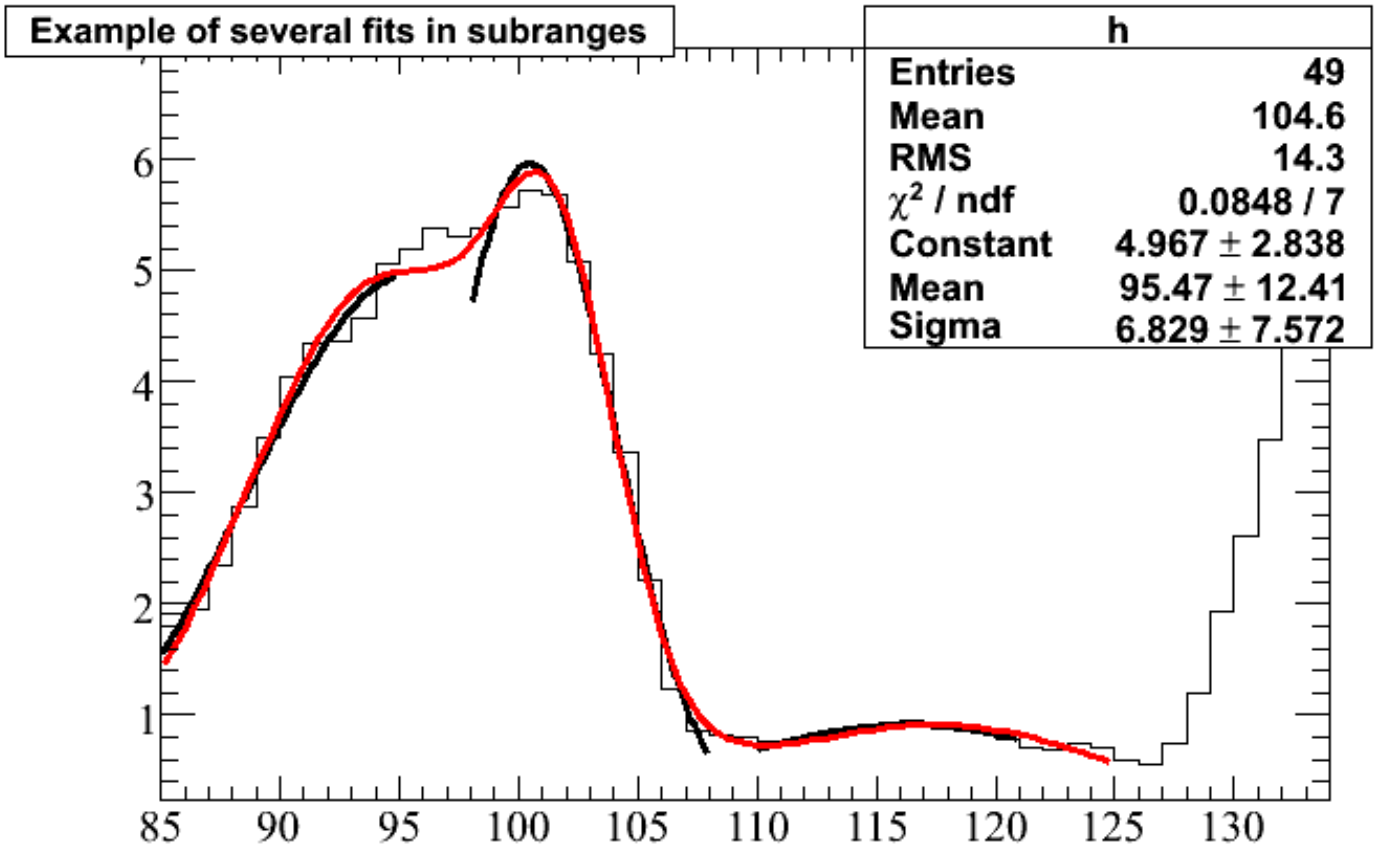
Fitting Histogram with Fit Panel





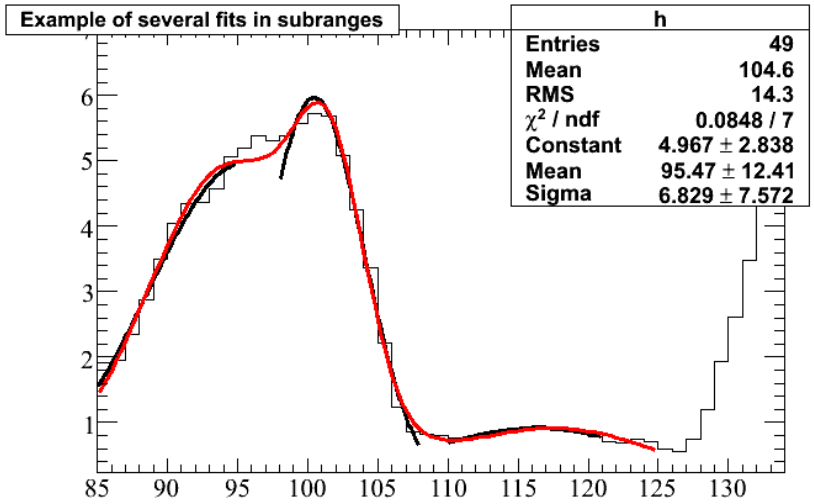
Same as: h1->Fit("gaus");

Fitting Multiple Sub Ranges

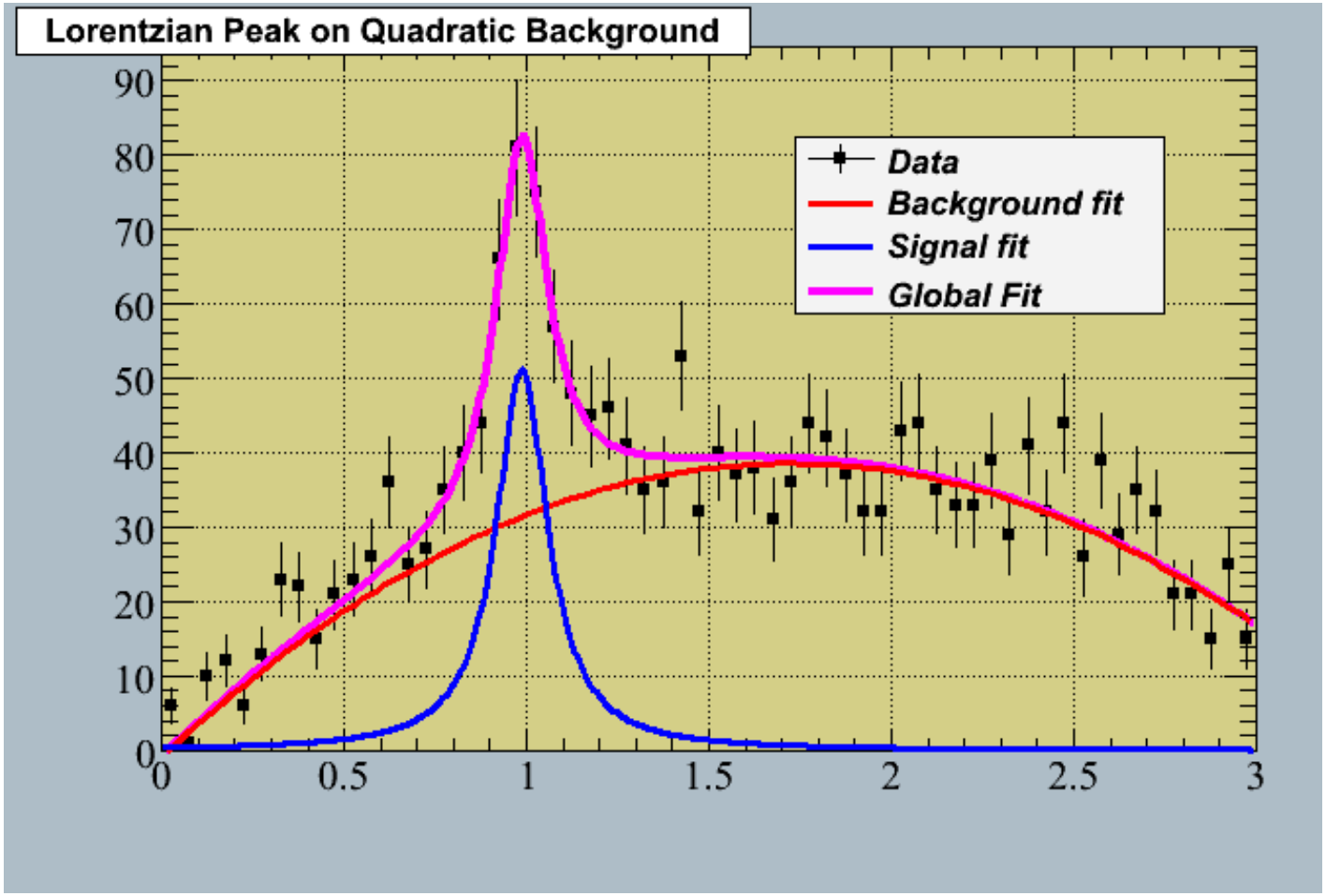


Fitting Multiple Sub Ranges contd.

- Double_t par[9];
- TF1 *g1 = new TF1("g1","gaus",85,95);
- TF1 *g2 = new TF1("g2","gaus",98,108);
- TF1 *g3 = new TF1("g3","gaus",110,121);
- TF1 *total = new TF1("total","gaus(0)+gaus(3)+gaus(6)",85,125);
- total->SetLineColor(2);
- h->Fit(g1,"R");
- h->Fit(g2,"R+");
- h->Fit(g3,"R+");
- g1->GetParameters(&par[0]);
- g2->GetParameters(&par[3]);
- g3->GetParameters(&par[6]);
- total->SetParameters(par);
- h->Fit(total,"R+");

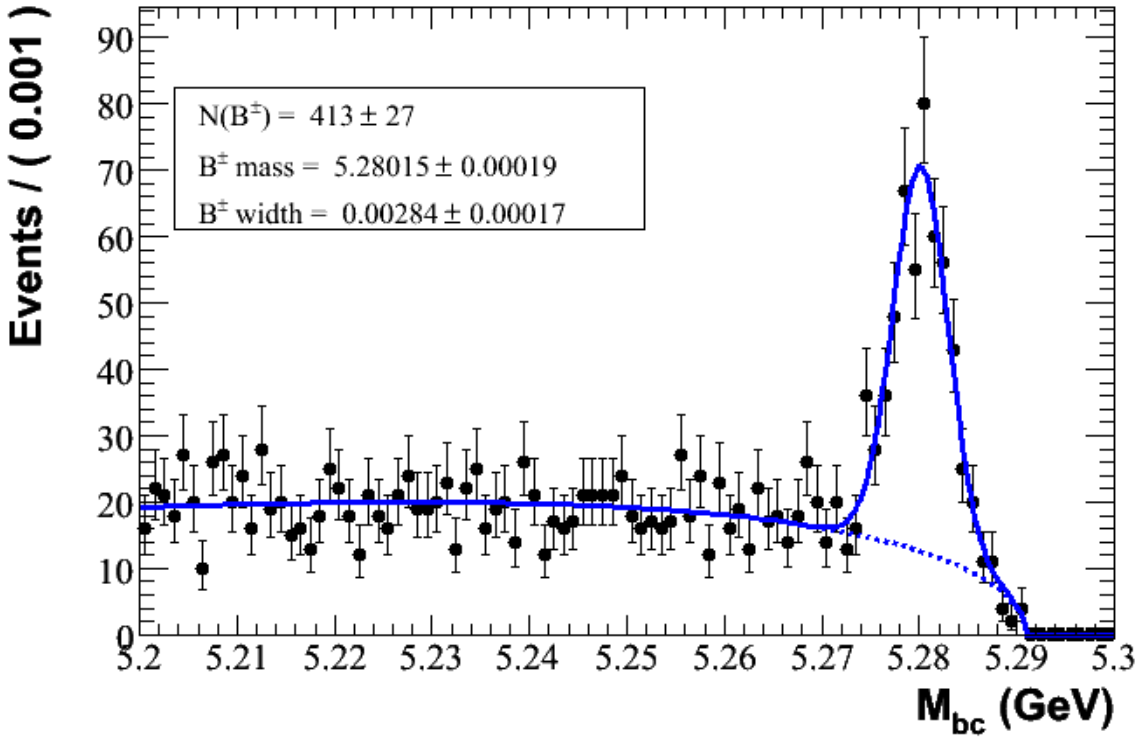


Fitting with Combining Functions



Fitting with RooFit (http://roofit.sourceforge.net/)

- RooFit packages provide a toolkit for modeling the expected distribution of events in a physics analysis
- Models can be used to perform likelihood fits, produce plots, and generate "toy Monte Carlo" samples for various studies



Next Lecture
Analysis of Muon Calibration Simulation Data with ROOT