Introduction to Statistics

Introduction to ROOT

Master Class Winter 2019/2020

Binish Batool

University of Siegen

08-10-2019

Introduction

- Root is a flexible open source software framework that provides programmable and graphical user interface for the purpose of data Analysis. It has built-in mathematical models to compare measurements and analyse them.

- How to get it download
  - Go to `https://root.cern.ch/content/release-61600` and get root_v6.16.00.win32.vc15.debug.exe

- How to run code in root
  - Open Your favourite document writer
  - void myfirstcode ()
  - {
  - for (int i = 0; i<3; i++) cout « "Hello" « i « endl ;
  - }
  - Save your newly written document file with extension '.C' e.g myfirstcode.C

Introduction (Cont...)

- How to run code in root (Cont....)
  - type *root*
  - root[0] .L myfirstcode.C (Load)
  - root[0] myfirstcode() (runs after loading)
  - root[1] .x myfirstcode.C (1 step running)
  - root[2] .q (to finish the root session)

- ROOT as a Calculator
  - root[0] 1+sqrt(9)
  - root[1] (const double)4.00000000000000000e+00
  - root[2] double val=0.17;
  - root[3] sin(val)
  - root[4] (const double)1.69182349066996029e-01

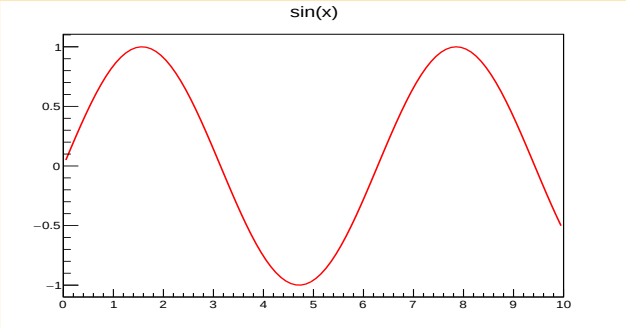
Let's fire up the ROOT

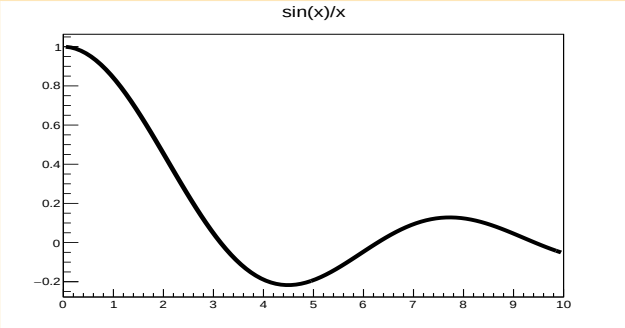
Introduction Cont...

Plotting a function:

The function class TF1 is the one of the basics ROOT classes

- Let's draw a function "sin(x)" in the range [0-10]:
  - root[0] TF1 * f1 = new TF1("f1","sin(x)",0,10);
  - root[1] f1 -> Draw();
  - root[2] f1->SaveAs("myfirstfunction.pdf");





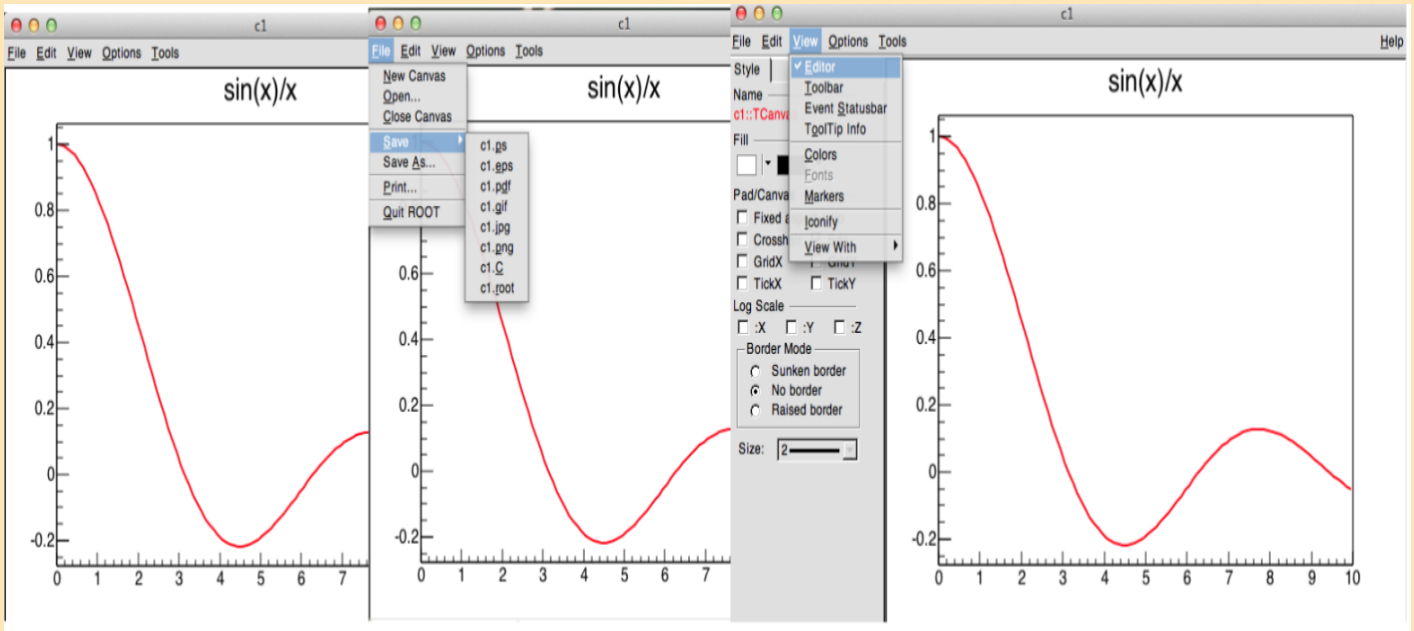
- TF1 * f1 = new TF1("f1","sin(x)/x",0,10);
- f1->SetLineColor(kBlack); f1->SetLineWidth(6.0);
- more option

  `https://root.cern.ch/doc/master/classTAttLine.html`

Introduction Cont ....

Using root GUI

- Style (line color, line width, line style)
- Edit function bin, axis
- Save different formates

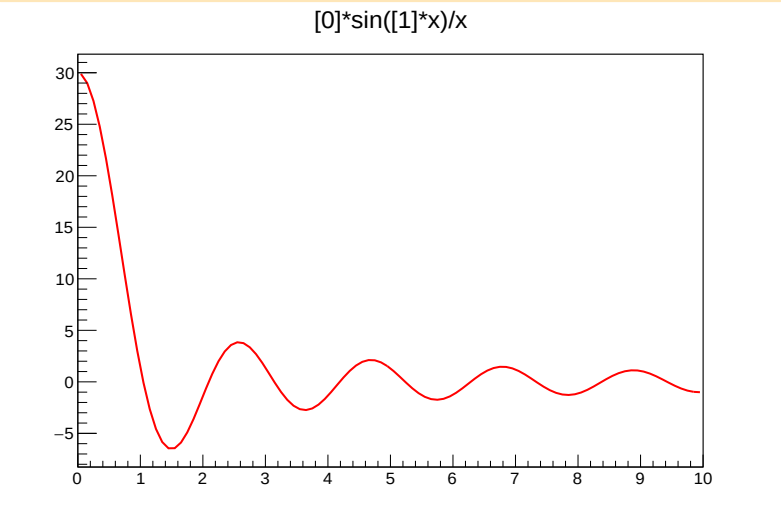


Introduction Cont ....

Function With Parameters

The function class TF1 can be used to create with any number of parameters.

- Let's create a function "p0sin(p1x)/x" with two parameters.
  - root[0] TF1 * f2 = new TF1("f2","[0]*sin([1]*x)/x",0,10);
- We need to set values of parameters:
  - root[2] f2->SetParameter(0,10);
  - root[3] f2->SetParameter(1,3); root[4] f2->Draw();



Introduction Cont ....

Function With Parameters

One can extract several informations from a function. For example:

- Extract Parameter values:.
  - root[2] f2 -> GetParameter(0);
- Extract value of a function at any point:
  - root[3] f2 -> Eval(2.5);
- Integrate in given range:
  - root[4] f2 -> Integral(0,4);
- more option
  `https://root.cern.ch/doc/master/classTF1.html`

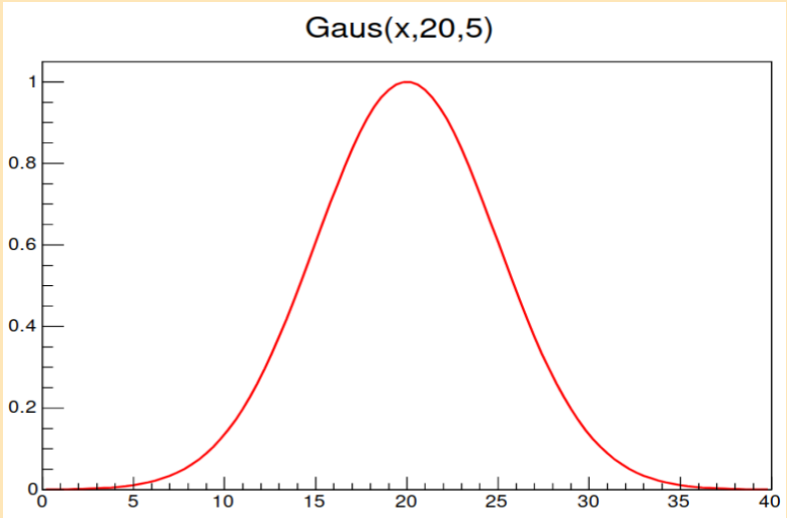
Generally
TF1 * pointer = new TF1(“name”,“formula”,min,max);
ROOT::TMath Namespace have almost all popular functions

Introduction Cont ....

Mathematical Functions in ROOT

TMath namespace provides:

- Numerical constants.
- Functions to work with arrays.
- Statistical functions (e.g. Gauss) For example, a Gauss function can be defined as:
  - root[ ] TMath::Gaus(x, mean, sigma);
- so a Gauss TF1 can be defined as
  - fb1 = new TF1("m1","Gaus(x,20,5)",0,40); fb1 ->Draw()

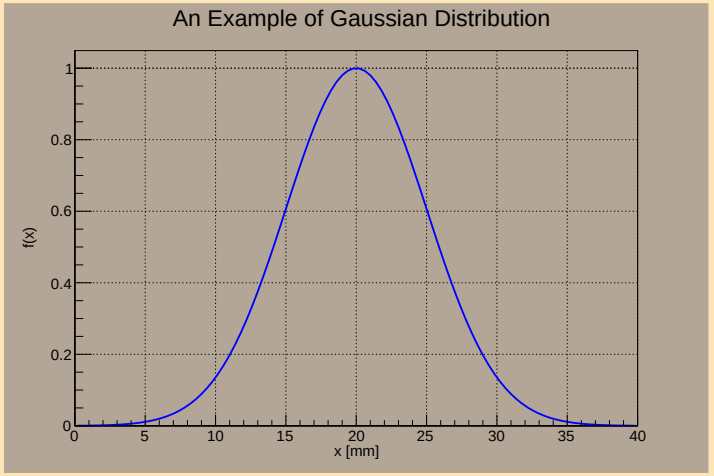


Introduction Cont ....

Mathematical Functions in ROOT Cont...

Cosmetics

- void myfunc()
- {
- TF1 *fb1 = new TF1(“m1”,“TMath::Gaus(x,20,5)”,0,40);
- fb1 -> SetTitle(”An Example of Gaussian Distribution”);
- fb1 -> GetXaxis() -> SetTitle(”x [mm]”);
- fb1 -> GetYaxis()-> SetTitle(”f(x)”);
- fb1 ->GetXaxis()->CenterTitle();
- fb1 -> GetYaxis()->CenterTitle();
- fb1 ->SetLineColor(kBlue);
- fb1 -> Draw( );
- gPad ->SetGridx();
- gPad -> SetGridy();
- gPad ->SetFillColor(24);
- fb1 ->Draw( );
- }



Introduction Cont ....

Exercise

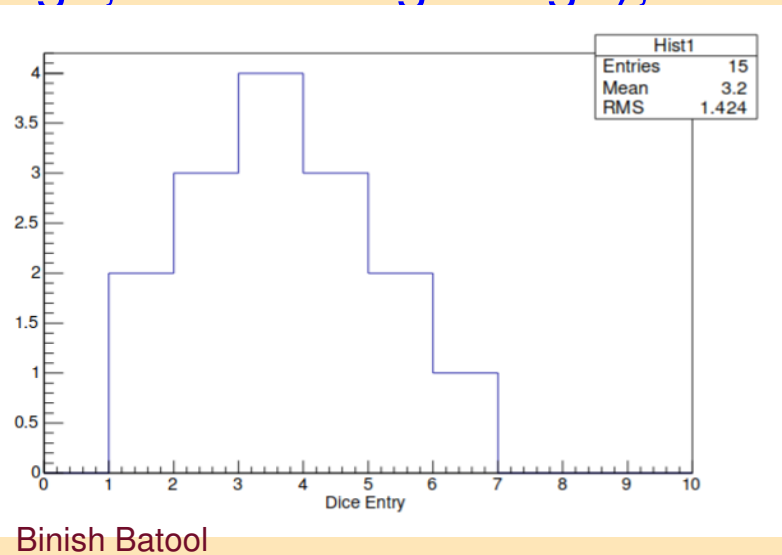
- Draw a function f (x) = p_0(ln(p_1x)/x for parameter values p0 = 5 and p1 = 3 in the range 0 to 8.
- Set the blue color to function and extract following information from the function.
- Value of function at x = 3, integral of function from 2 to 8 and function derivative at x = 2.

Introduction Cont ....

Histogram:

A Histogram represents an occurrence counting. It can be thought as a graphical representation of the data

For example, we throw a dice 15 times and collect the following data:
{1,3,2,6,3,2,1,4,3,2,4,5,4,3,5}, which makes a histogram
TH1F *Hist1 = new TH1F("Hist1","Title ",no ofBins, first bin low edge, end bin high edge);



Dice entry (x-axis)	Occurrence (y-axis)
1	2 times
2	3 times
3	4 times
4	3 times
5	2 times
6	1 time

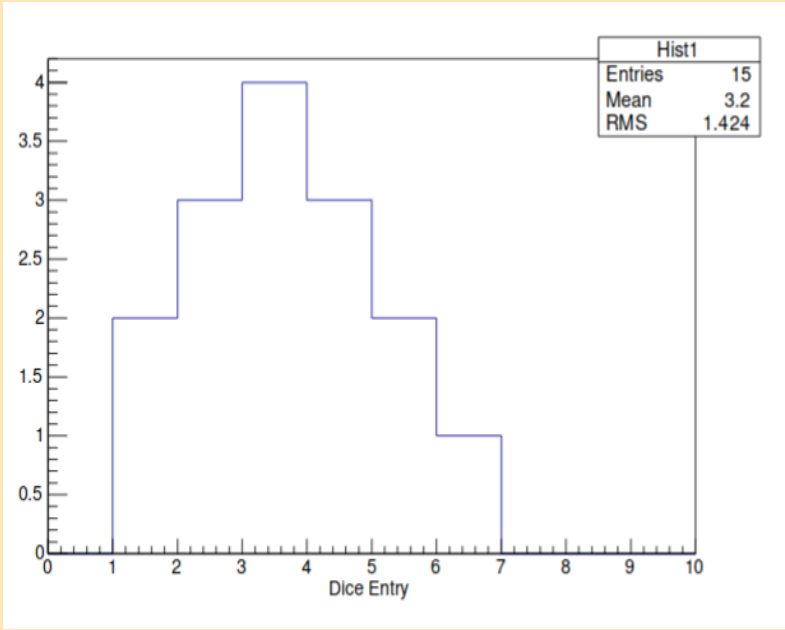
Introduction Cont ....

Filling a histogram

A Histogram is filled on the basis of entry-by-entry (event-by-event)

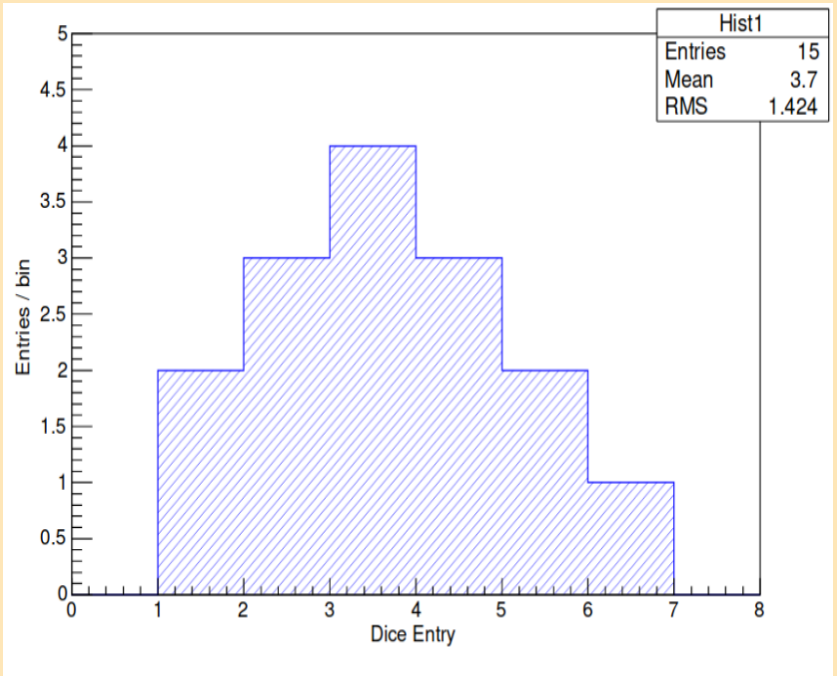
For example, in previous examples histogram was filled 15 times i.e. there are 15 events in the language of data analysis.

```
void DiceHisto()
{
const int n = 15;
int occurence[n]= {1,3,2,6,3,2,1,4,3,2,4,5,4,3,5};
TH1F* Hist1 = new TH1F("Hist1"," ",10, 0,10);

for(int i = 0; i < n; i++)
{ // Event Loop starts
Hist1 -> Fill(occurence[i]);
} // Event Loop ends

Hist1 ->GetXaxis() -> SetTitle("DiceEntry");
Hist1 -> GetXaxis() -> CenterTitle();
Hist1 -> Draw();
}
```



Cosmetics for histogram

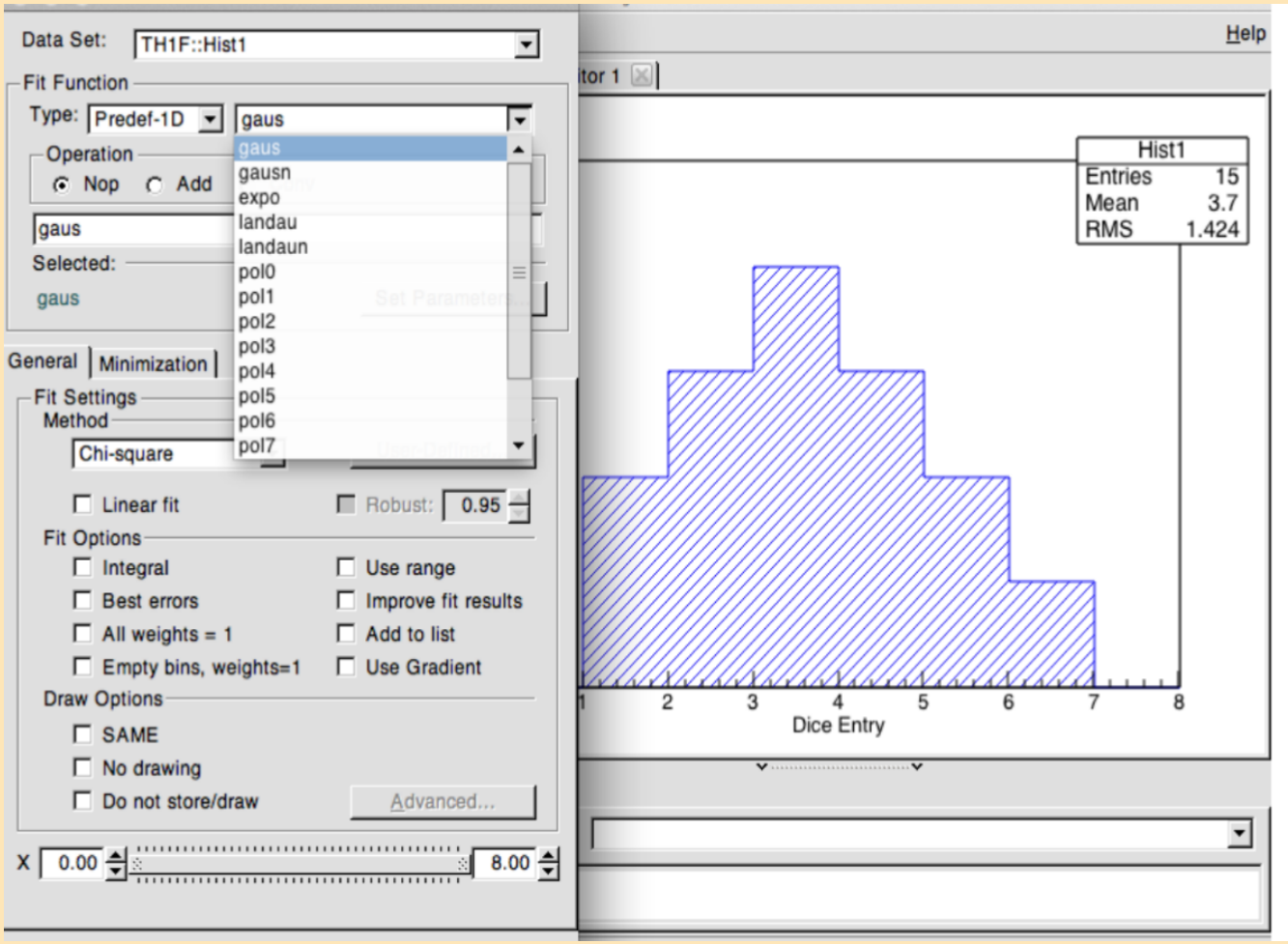

```
void DiceHisto()
{
const int n = 15;
int occurence[n]= {1,3,2,6,3,2,1,4,3,2,4,5,4,3,5};
TH1F* Hist1 = new TH1F("Hist1"," ",10, 0,10);

for(int i = 0; i < n; i++)
{ // Event Loop starts
Hist1 -> Fill(occurence[i]);
} // Event Loop ends

Hist1 ->GetXaxis() -> SetTitle("DiceEntry");
Hist1->GetYaxis()->SetTitle("Entries/bin");
Hist1->GetXaxis()->CenterTitle();
Hist1->GetYaxis()->CenterTitle();
Hist1->GetXaxis()->SetRangeUser(0,8);
Hist1->GetYaxis()->SetRangeUser(0,5);
Hist1->SetLineColor(kBlue);
Hist1->SetFillColor(kBlue);
Hist1->SetFillStyle(3002);
Hist1->Draw();
c1->SaveAs("myhisto.pdf");
}
```



Introduction Cont ....

Histogram fitting using GUI

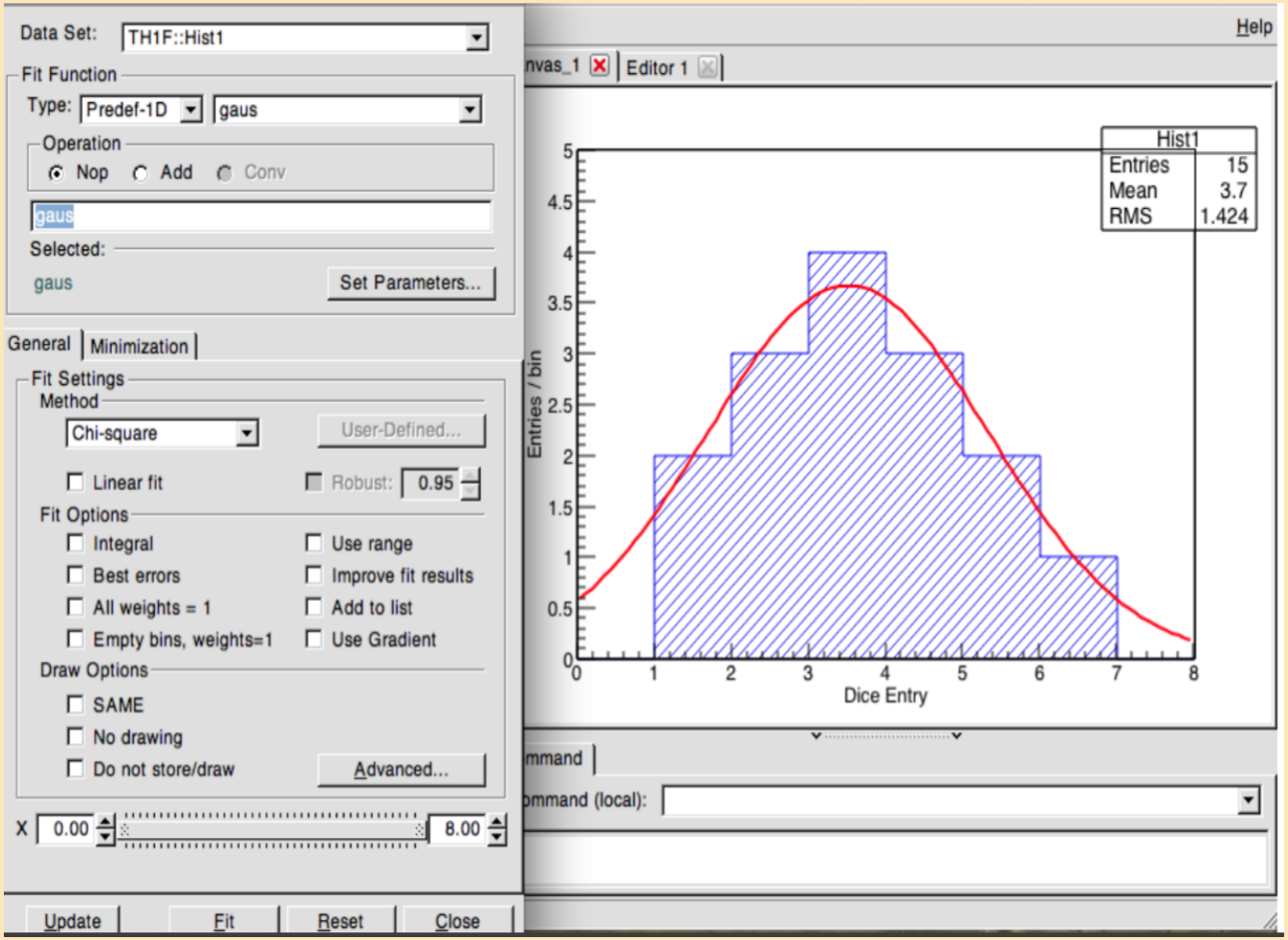
Introduction to Statistics



Introduction Cont ....

Histogram fitting using GUI

Introduction to Statistics



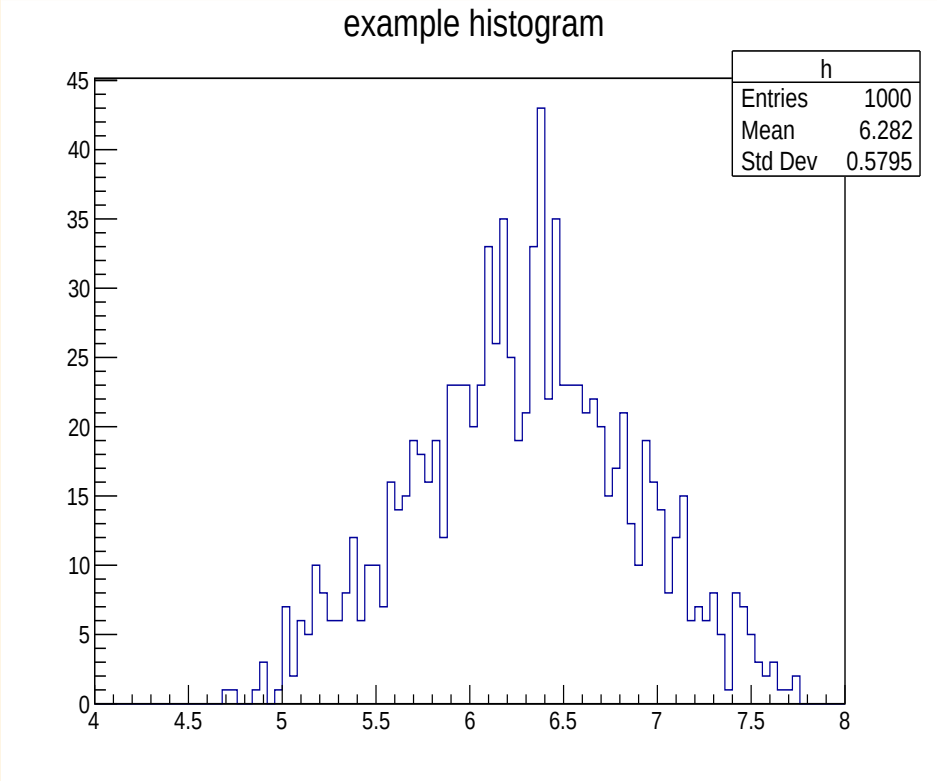
Histogram reading from text file

Introduction to Statistics

A histogram could be read from root, text or .dat file

```
void textfileReader()
{
TH1F *h = new TH1F("h", "example histogram",100,4.,8.);
ifstream inp;
double x;
inp.open("expo.dat");
while(!(inp » x)==0)
{
h->Fill(x);
}
h->Draw();
}
```

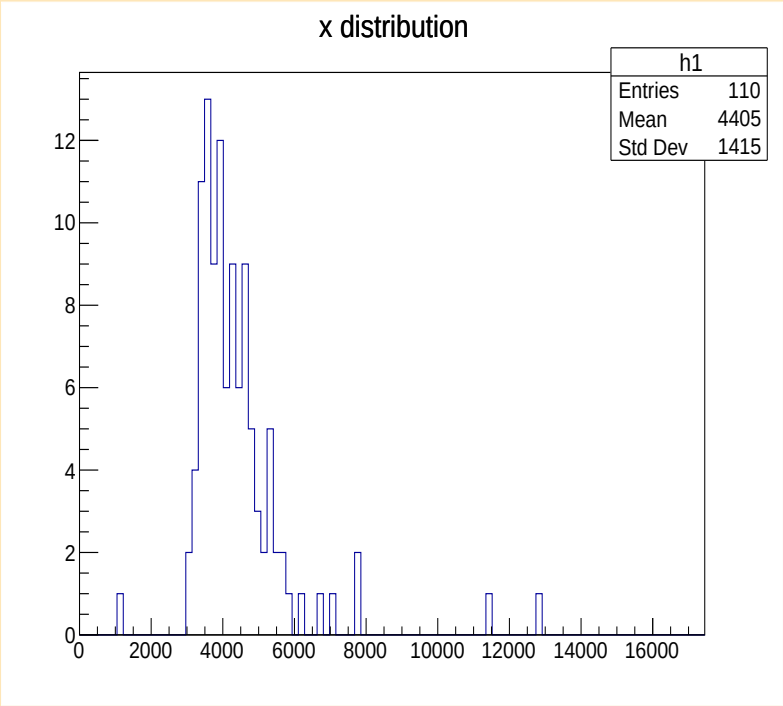
The range of such histogram depends on the minima and maxima of the datasample



Introduction Cont ....

Histogram reading if file has larger data or many columns)

```
void textfile()
        {
vector < int > datapoints;
TNtuple *ntuple = new TNtuple("ntuple", "data from file", "x");
ifstream inp; double x; inp.open("datos.txt");
while(!(inp >> x)==0)
{
ntuple->Fill(x);
}
int xbins = 100;
double xmin = ntuple->GetMinimum("x");
double xmax = ntuple->GetMaximum("x");
TH1F *h = new TH1F("h1", "x distribution", xbins, 0, xmax);
ntuple->Project("h1", "x");
h->Draw();
        }
```

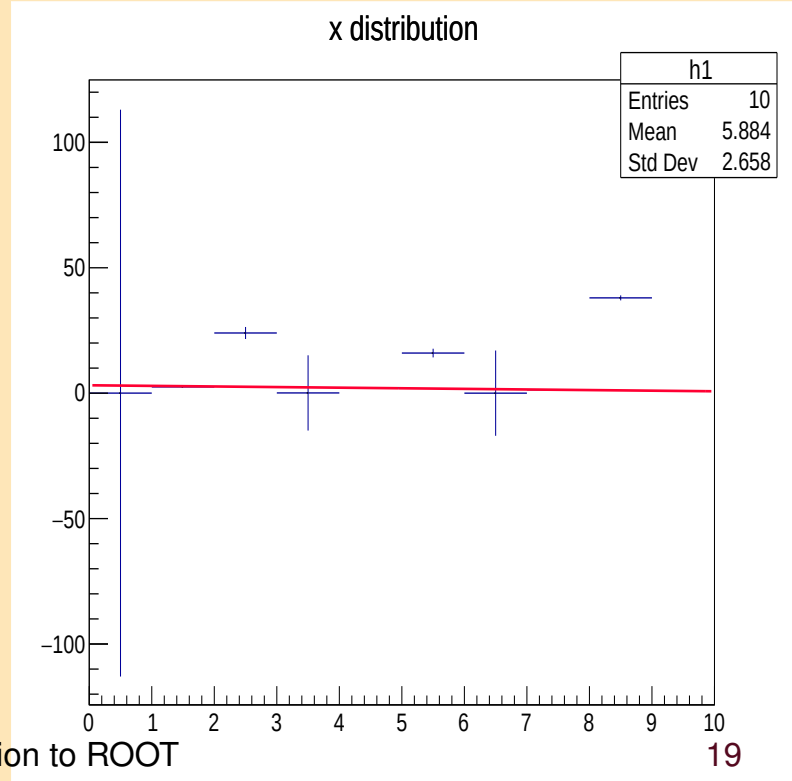


Fit (Linear, Gaussian, Exponential)


```
void mymacro()
{
double col1[12];
double col2[12];
double col3[12];
ifstream infile; double x;
infile.open("lineardata.txt");
int num =0;
while(!infile.eof())
{
infile » col1[num] » col2[num] » col3[num];
++num;
}
infile.close();
TH1F *h = new TH1F("h1", "x distribution",
10, 0, 10);
for (int i = 0; i < 10; i ++)
{
h->SetBinContent(i,col1[i]);
h->SetBinError(i,col3[i]);
}
h->Draw();
h ->Fit("gaus") ;
TF1 *gausfit = h->GetFunction("gaus");
gausfit ->SetLineColor (kRed);
}
```

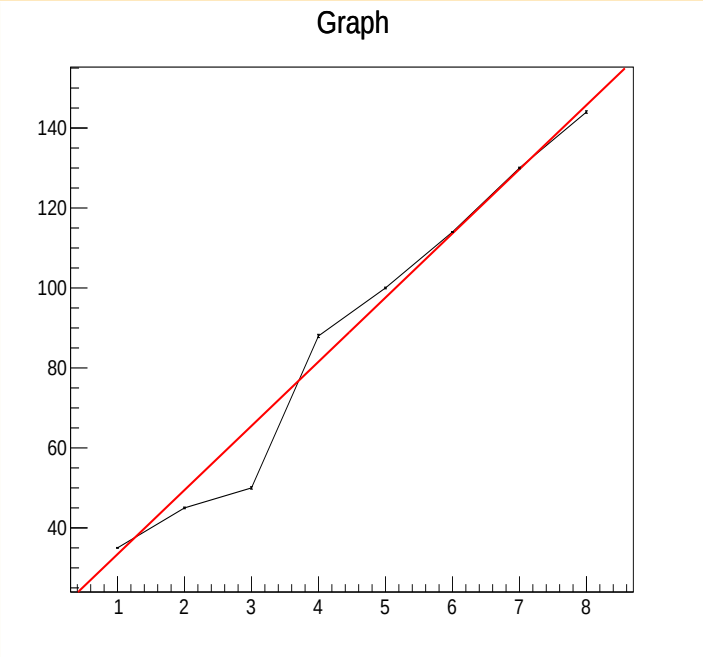
```
h->Fit("pol1");
TF1 *gausfit = h->GetFunction("pol1");
polfit ->SetLineColor(kPink);
```

```
h ->Fit("expo") ;
TF1 *gausfit = h->GetFunction("expo");
expofit ->SetLineColor (kRed);
```



Fitting using TGraph

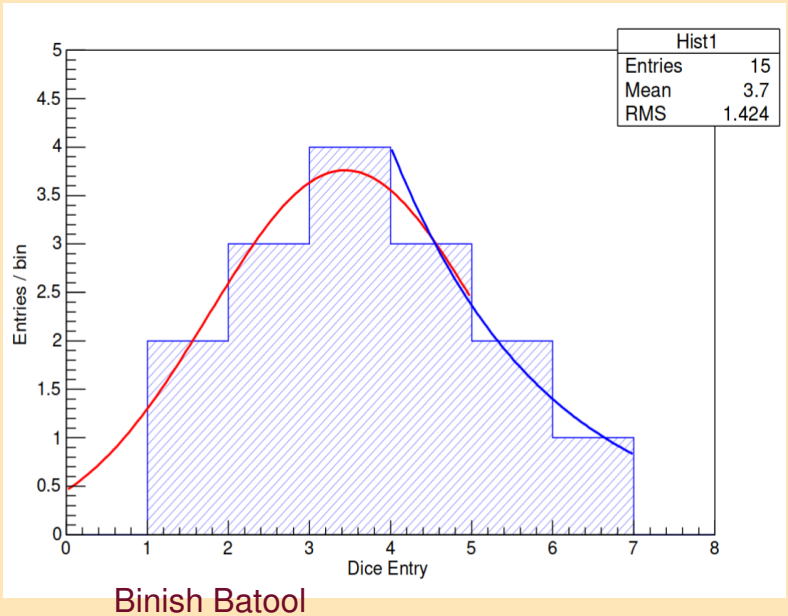

```
void mygraph() {
double col1[10];
double col2[10];
double col3[10];
ifstream infile;
double x;
infile.open("graphdata.txt");
int num =0;
while(!infile.eof())
{
infile >> col1[num] >> col2[num] >> col3[num];
++num;
}
infile.close();
int bins = 8;
TGraph *g = new TGraphErrors(bins, x-axis, y-axis, erro x-axis , error y-axis);
TGraph *g = new TGraphErrors(bins, col1, col2, 0 , col3);
g->Fit("pol1");
g->Draw() ; }
```



Introduction Cont ....

Histogram fitting using macro

- Define Hist1 same as in previous example
- define the fit fuction
  - g1 = new TF1(“m1”,“gaus”,0,5); //range (0,5)
  - g2 = new TF1(“m2”,“expo”,4,7); //range (4,7)
  - g2 -> SetLineColor(kBlue);
  - Hist1 -> Fit(g1,“R”);
  - Hist1 -> Fit (g2,"R+");




```
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
 FCN=0.0346963 FROM MIGRAD    STATUS=CONVERGED      86 CALLS          87 TOTAL
                     EDM=7.26517e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     3.76015e+00   1.55551e+00   5.40007e-04  -6.47632e-05
   2  Mean         3.43840e+00   9.64578e-01   3.77605e-04   1.15037e-04
   3  Sigma        1.67249e+00   1.37496e+00   1.08339e-04  -1.82000e-04
 FCN=0.0240263 FROM MIGRAD    STATUS=CONVERGED      44 CALLS          45 TOTAL
                     EDM=6.18558e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     3.49073e+00   2.74819e+00   2.02368e-04   8.41835e-05
   2  Slope       -5.25762e-01   5.25854e-01   3.87272e-05   6.36175e-04
```

Thank you

Backup

Introduction Cont ....

Two Histograms on a same canvas

Now we throw two dices 15 times. One is the real dice and other one is a simulated dice whose data is generated by a computer program.

Real data: { 1,3,2,6,3,2,1,4,3,2,4,5,4,3,5 }

and simulated data : { 1,3,2,6,5,2,2,4,3,2,4,6,4,3,5 }

- void TwoDicesHisto() {
- const int n = 15;
- int occurence1[n]= /1,3,2,6,3,2,1,4,3,2,4,5,4,3,5/;
- int occurence2[n]= /1,3,2,6,5,2,2,4,3,2,4,6,4,3,5/;
- TH1F* Hist1 = new TH1F(“Hist1”,“ ”,8, 0,8);
- TH1F* Hist2 = new TH1F(“Hist2”,“ ”,8, 0,8);
- for(int i = 0; i < n; i++){ / Event Loop starts
- Hist1 ->Fill(occurence1[i]);
- Hist2 ->Fill(occurence2[i]);} / Event Loop ends
- Hist1 ->Draw();
- Hist2 ->Draw(”same”); }

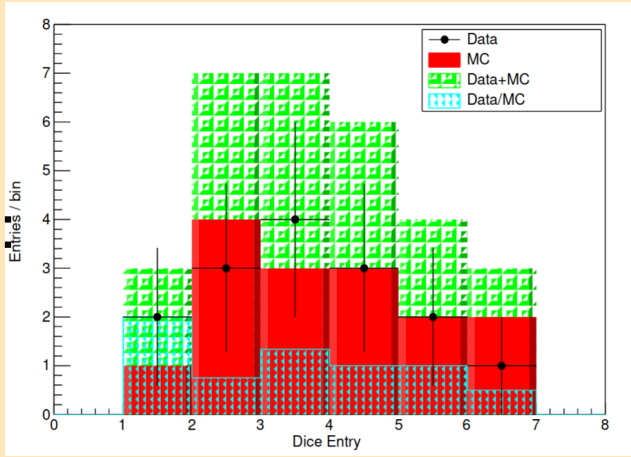
Introduction Cont ....

Histogram manipulation: Adding/Dividing two histograms

- You can add/divide two histograms of same binning:
  - TH1F* Hist1 = new TH1F("Hist1"," ",8, 0,8);
  - TH1F* Hist2 = new TH1F("Hist2"," ",8, 0,8);
  - TH1F* Hist1plus2 = new TH1F("Hist1plus2","",8, 0,8);
  - TH1F* Hist1over2 = new TH1F("Hist1over2","",8, 0,8);

- After filling "Hist1" and "Hist2", operation can be done:
  - Hist1plus2 -> Add(Hist1,Hist2);
  - Hist1over2 -> Divide(Hist1,Hist2);

The histogram after adding/dividing two histograms is shown (made by Asif).



Introduction Cont ....

Weighting and Scaling histograms

Weighting a histogram means multiplying a constant factor (weight) to each bin content.

- For 1D adding weight is like following:
  - h1D ->Fill(x, weight)

- For example a histogram (h1) has a specific distribution with 100 entries and the other histogram (h2) has the same distribution but with 1000 entries. Now scaling/normalizing h1 with respect to h2 is following:
  - h1 > Scale(10);

or scaling h2 with respect to h1 is

- - h2 ->Scale(0.1);