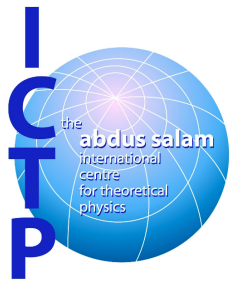
Introduction to ROOT

Asif Saddique

National Centre for Physics (NCP), Islamabad, Pakistan

Histogram Operations
Tutorial # 2

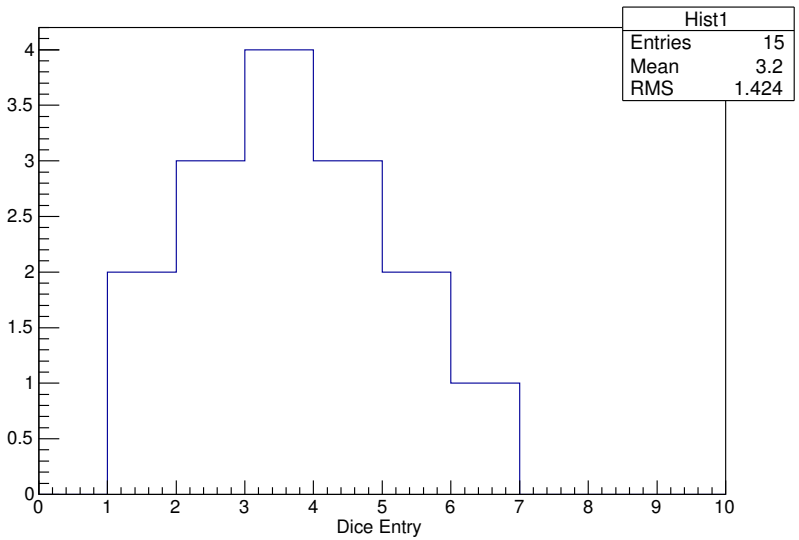
November 26, 2014



ICTP-NCP School on LHC Physics

What is a Histogram?

A Histogram represents an occurrence counting. It can be thought as a graphical representation that shows the visual distribution 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.



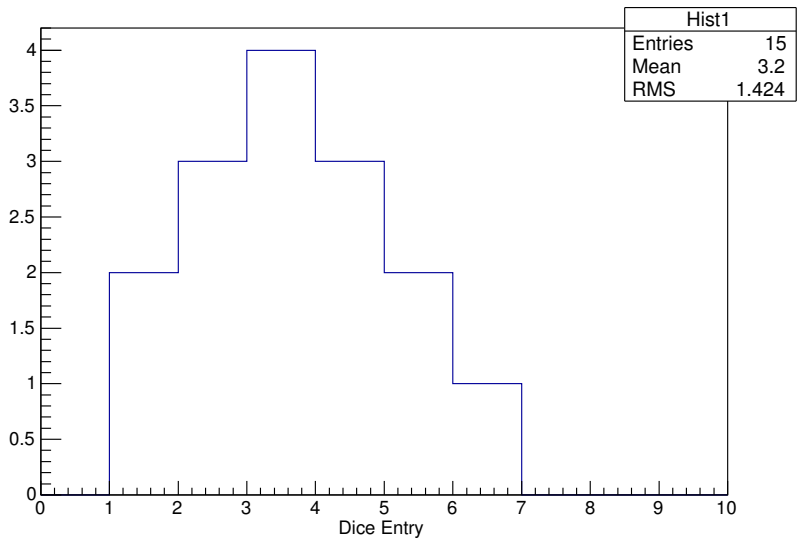
Dice entry (x-axis)	Occurrence (y-axis)
1	2 times
2	3 times
3	4 times
4	3 times
5	2 times
6	1 time

TH1F *Hist1 = new TH1F("Hist1","Title ",#ofBins, StartBin, EndBin);
The statistics box also gives other informations about the distributions.

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.

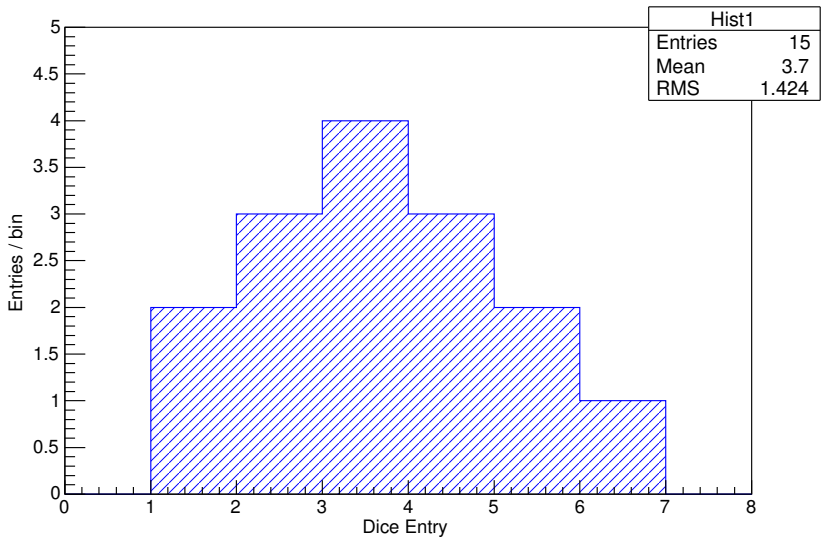



```
void DiceHisto() {
const int n = 15;
int occurence[n]= {1,3,2,6,3,2,1,4,3,2,4,5,4,3,5};
TH1F* Hist1 = new TH1F("Hist1"," ",10, 0,10);
for(int i = 0; i < n; i++){ // Event Loop starts
Hist1->Fill(occurence[i]);
} // Event Loop ends
Hist1->GetXaxis()->SetTitle("Dice Entry");
Hist1->GetXaxis()->CenterTitle();
Hist1->Draw();
}
```

Now let's try to add cosmetics (change axis range, put titles, fill color and put a fill style) to this plot.

Cosmetics for a Histogram

A user can change the range/color/filling etc. even after filling a 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("Dice Entry");
Hist1->GetYaxis()->SetTitle("Entries/bin");
Hist1->GetXaxis()->CenterTitle();
Hist1->GetYaxis()->CenterTitle();
Hist1->GetXaxis()->SetRangeUser(0,8);
Hist1->GetYaxis()->SetRangeUser(0,5);
Hist1->SetLineColor(4);
Hist1->SetFillColor(4);
Hist1->SetFillStyle(3002);
Hist1->Draw();
c1->SaveAs("/Users/asif/RootTut/Pics/myhisto.eps"); }
```

Please Note: Histogram appears on a default canvas "c1" and you can also save your histo in many formats (eps, png, pdf, tif etc.)

Let's forget about cosmetics for now, and try to add another histogram to the same canvas.

Writing a ROOT file

- You can write your plots in a ROOT file.

```
TFile *file = new TFile("dicehisto.root","recreate");
Hist1->Write();
```

Instead of "recreate" you can use "update", "new" etc., you can also find other details:
`http://root.cern.ch/root/html/TFile.html`

Here, histogram will sit in top level directory.

- You can also make a subdirectory in the ROOT file.

```
file->mkdir("general");
```

- Now go into the created directory:

```
file->cd("general");
```

- Finally write your plot:

```
Hist1->Write();
```

Writing a ROOT file

```
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("Dice Entry");
Hist1->GetYaxis()->SetTitle("Entries/bin");
Hist1->GetXaxis()->CenterTitle();
Hist1->GetYaxis()->CenterTitle();
Hist1->GetXaxis()->SetRangeUser(0,8);
Hist1->GetYaxis()->SetRangeUser(0,5);
Hist1->SetLineColor(4);
Hist1->SetFillColor(4);
Hist1->SetFillStyle(3002);
Hist1->Draw();
TFile *file = new TFile("dicehisto.root","recreate");
file->mkdir("general");
file->cd("general");
Hist1->Write();
c1->SaveAs("/Users/asif/RootTut/Pics/myhisto.eps"); }
```

- Blue lines are the addition to previous code.
- After running code, we will have a root file "dicehisto.root" containing a directory "general", which has a histogram "Hist1".

Browsing a ROOT file

- To browse your ROOT file, type following in the terminal:

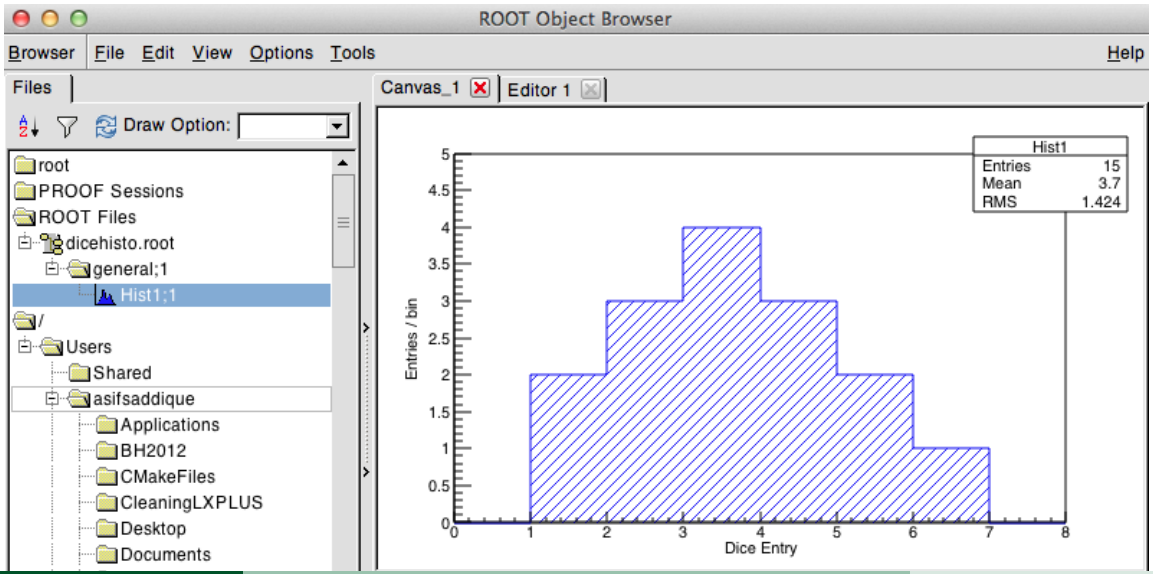
```
$ root /PathtoRootFile/dicehisto.root
```

It will connect the next prompt to your ROOT file.

- Then in the prompt:

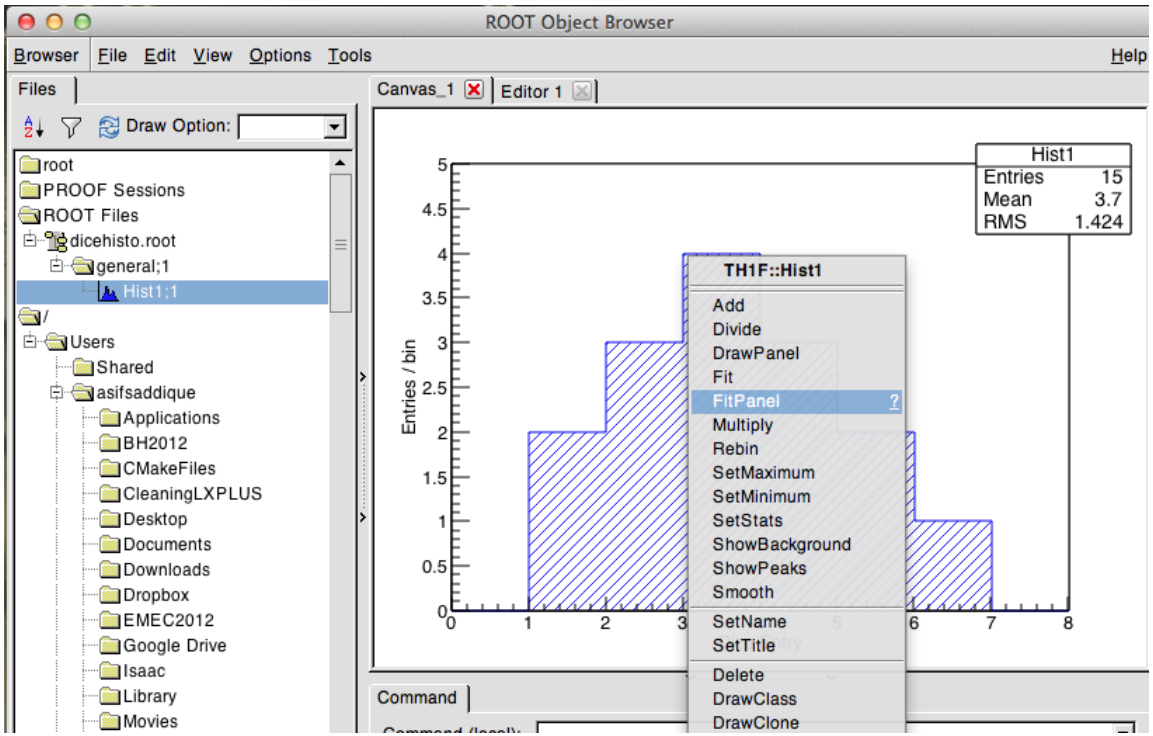
```
root [ ] TBrowser br //"br" can be any name you like, or alternatively
root [ ] new TBrowser
```

It will open a GUI to browse your file.



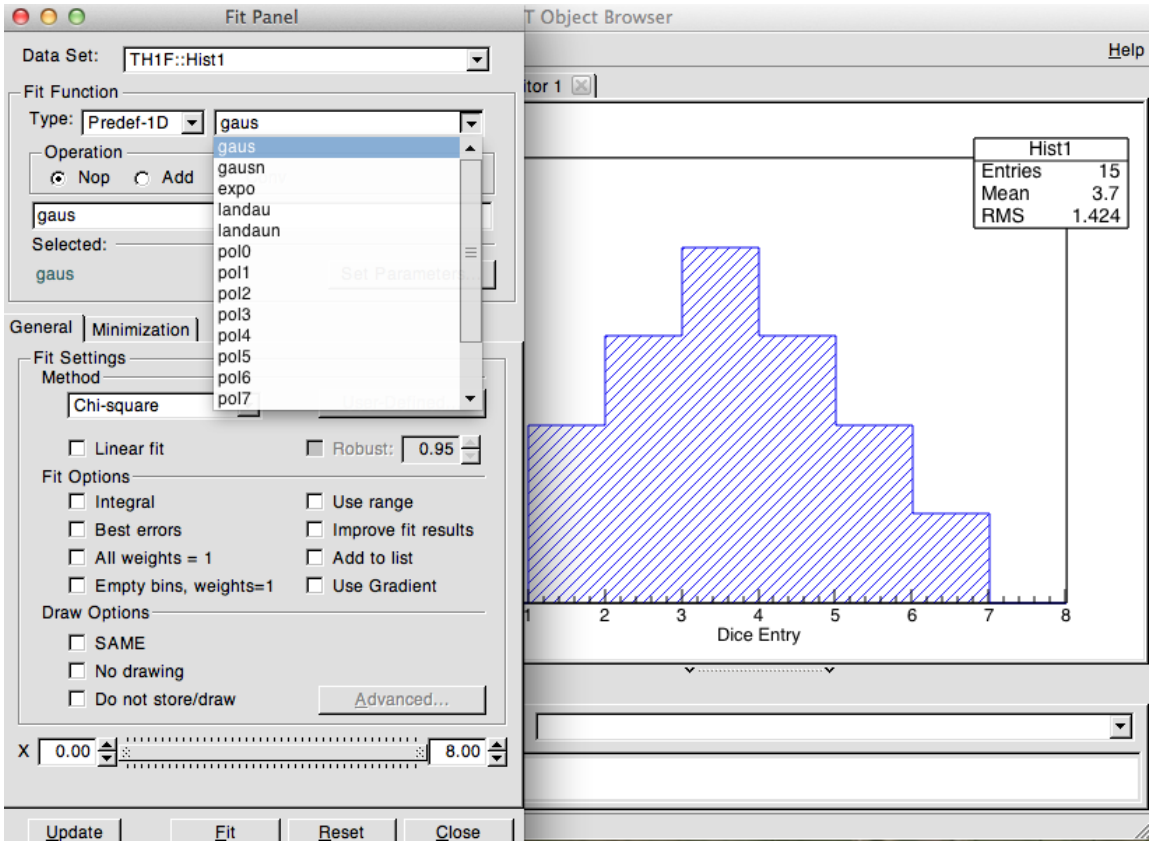
Browsing a ROOT file → Fitting a Histogram

- Right click on the histogram's bin-line to see the drop-down menu.
- Click on FitPannel:



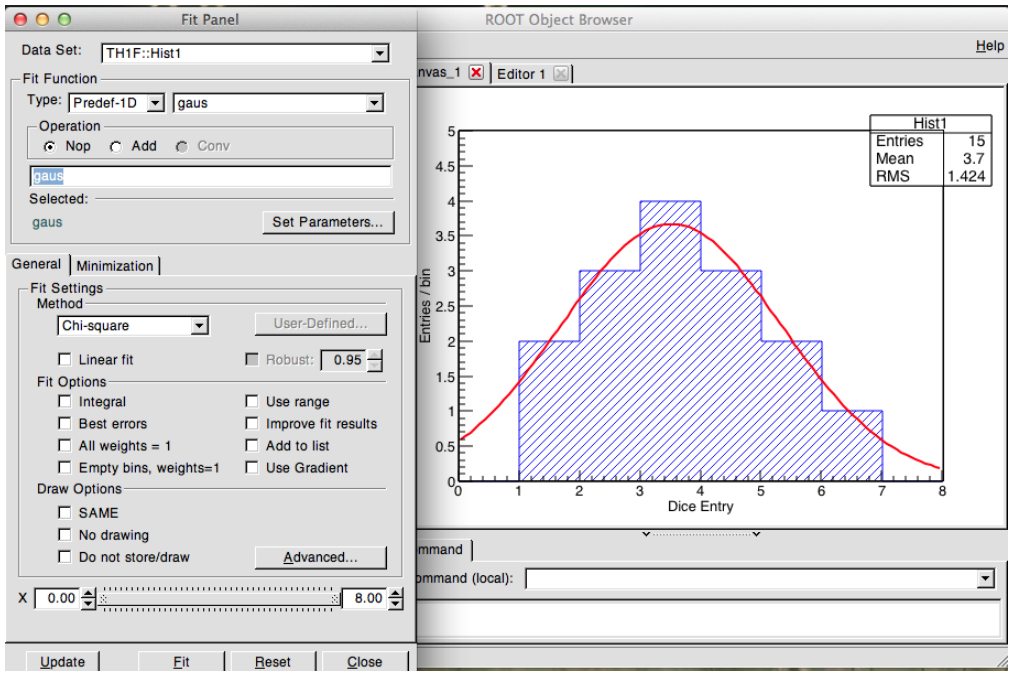
Browsing a ROOT file → Fitting a Histogram

- FitPanel has menu of built in function, select any function.
- Click on "Fit" button in the bottom



Browsing a ROOT file → Fitting a Histogram

- You can see a function is fitted to our histogram.



- After fitting fit statistics will appear in the prompt.

```
root [1] new TBrowser
(class TBrowser*)0x7fc2d24eedb0
root [2]  FCN=0.045818 FROM MIGRAD    STATUS=CONVERGED      68 CALLS          69 TOTAL
                     EDM=1.54757e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     3.67278e+00   1.30110e+00   4.74563e-04  -2.42435e-04
   2  Mean         3.51798e+00   6.43109e-01   2.96418e-04   6.36496e-04
   3  Sigma        1.81792e+00   7.32324e-01   5.31369e-05   1.48599e-03
```

Reading Histo from a ROOT file → Fitting

- You can read your plots from a ROOT file.
- It is useful for the further operation on a histogram.

```
TFile *myfile = new TFile("dicehisto.root","read");
TH1F *h = (TH1F*)gDirectory → Get("general/Hist1");
h → Draw();
```

- Now define function you want to fit:

```
g1 = new TF1("m1","gaus",0,5);
g2 = new TF1("m2","expo",4,7);
g2→SetLineColor(4);
```

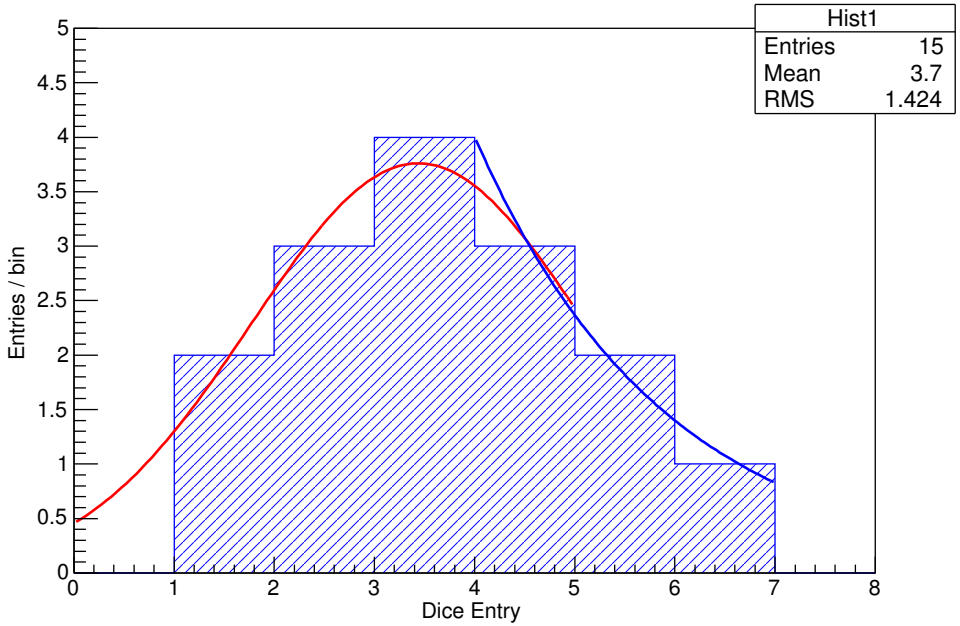
By default fit has red color, so "m1" is red and "m2" is blue.

- Perform fit in the given range:

```
h → Fit(g1,"R");
h → Fit(g2,"R+");
```

Reading Histo from a ROOT file → Fitting

- The fitted plot looks like following:



- Fit statistics from terminal window are shown as:

```
Processing ./ReadingRootFile_FitHisto.C...
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
```

Stat-box Switches

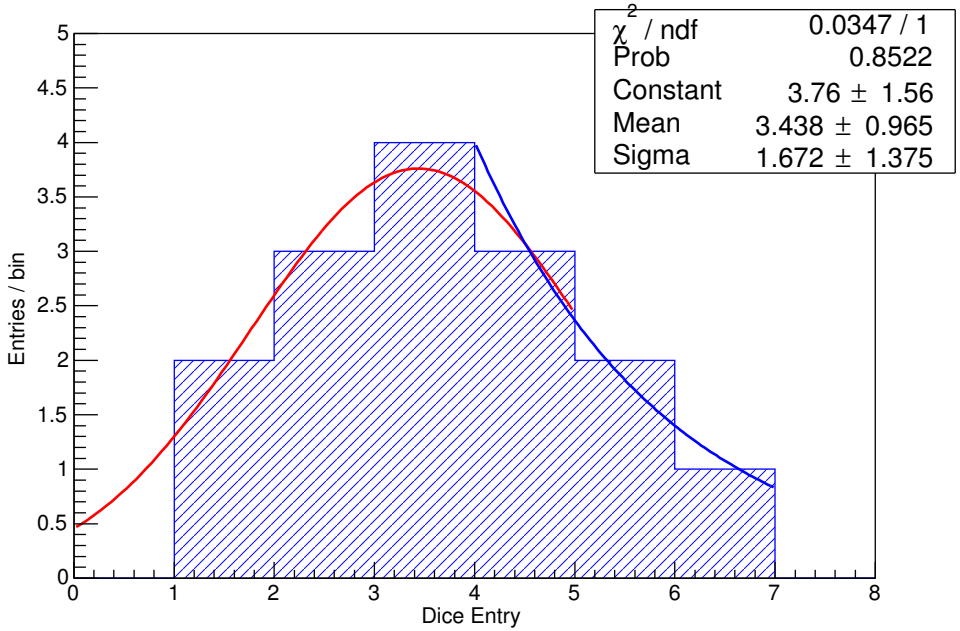
- Turn-off stat-box:

```
gStyle→SetOptStat(0);
```

- Turn-on fit sat-box:

```
gStyle→SetOptFit(1111);
```

- Fit statistics are now visible in the stat-box:

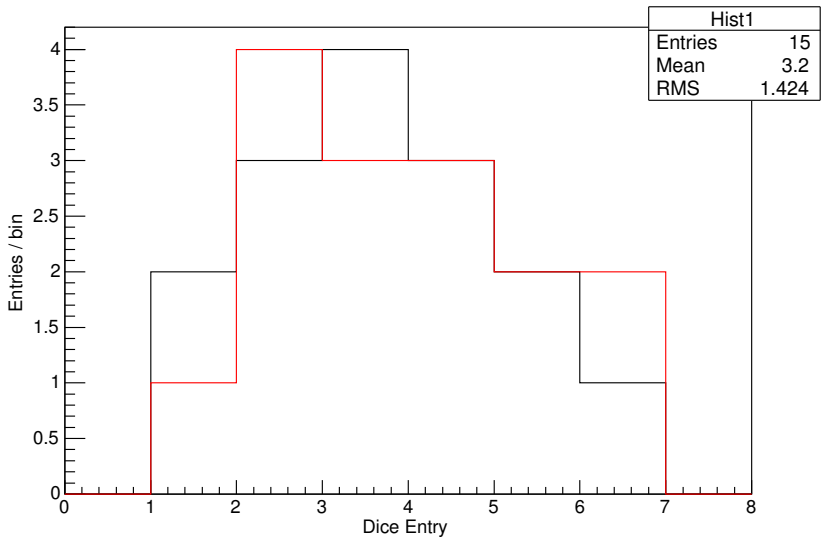


Two Histograms on a same canvas

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 (MC): {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->GetXaxis()->SetTitle("Dice Entry");
Hist1->GetYaxis()->SetTitle("Entries/bin");
Hist1->GetXaxis()->CenterTitle();
Hist1->GetYaxis()->CenterTitle();
Hist1->SetLineColor(4);
Hist1->Draw();
Hist2->SetLineColor(4);
Hist2->Draw("same");
}
```

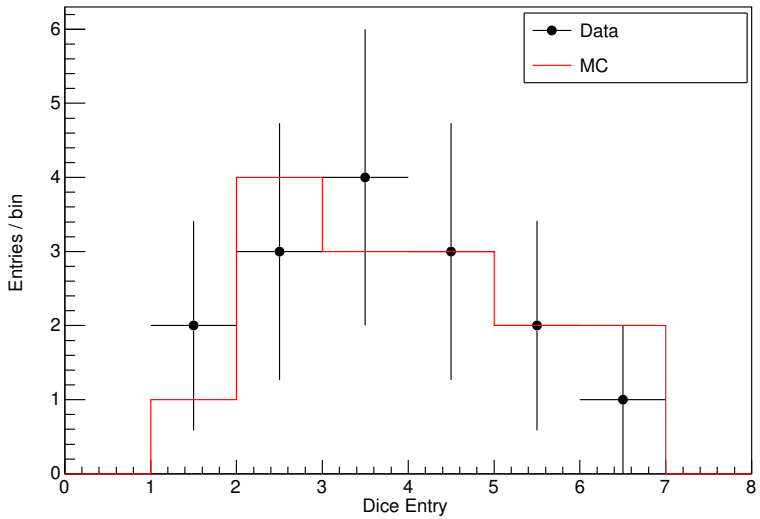
For more draw options, please check http://root.cern.ch/root/html/THistPainter.html

Two Histograms on a same canvas→ needs a Legend

Let's remove the stat box and add a legend to the plot.

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

MC : {1,3,2,6,5,2,2,4,3,2,4,6,4,3,5}

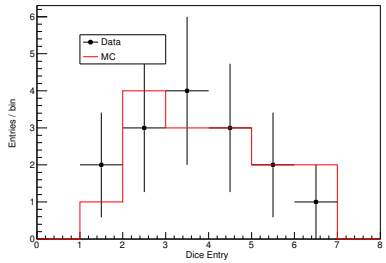



```
void TwoDicesHisto() {
gStyle→SetOptStat(0); // Removes stat box
const int n = 15;
.
.
for(int i = 0; i < n; i++){ // Event Loop starts
Hist1→Fill(occurence1[i]);
Hist2→Fill(occurence2[i]);
} // Event Loop ends
.
.
Hist1→Draw("E");
Hist2→Draw("same");
TLegend *leg = new TLegend(0.6,0.8,0.9,0.9);
leg→AddEntry( Hist1, "Data", "lp");
leg→AddEntry( Hist2, "Simulation", "l");
leg→SetTextFont(42);
leg→SetTextSize(0.035);
leg→SetFillColor(0);
leg→SetBorderSize(1);
leg→Draw("");
}
```

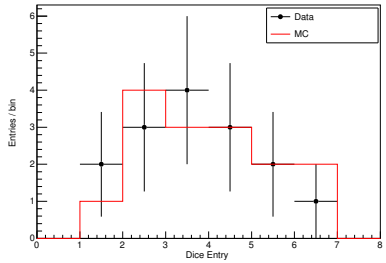
TLegend *leg = new TLegend(x1,y1,x2,y2);

A trick to set coordinates of TLegend

- Run the code with arbitrary coordinates value. For example, you will see legend at any place in canvas:



- Put it at the desired place by hand.



- Now save the plot as "test.C", and then inside "test.C" find the legend's coordinates and copy them.
- Replace the arbitrary coordinates in your code by the copied one.

Adding/Dividing two histograms

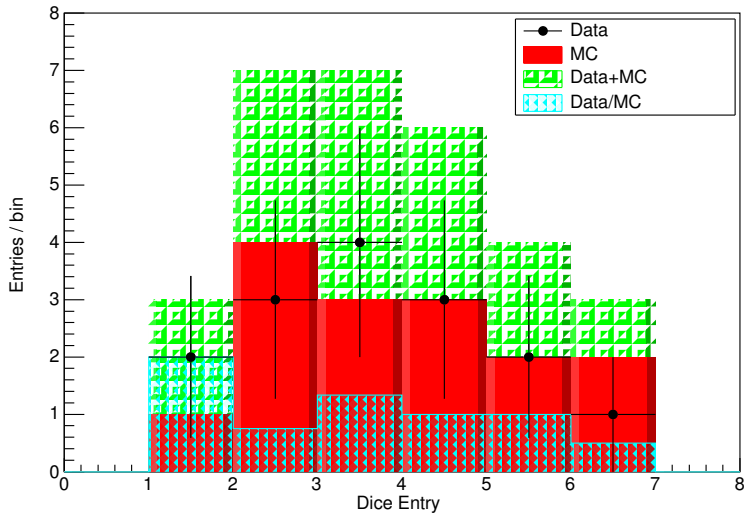
- You can add/divide two histograms of same binning:

```
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:



Note: Practically, data and MC are not added. They are divided to see their agreement. You can also subtract two histos by Add(Hist1,Hist2,1.,-1.).

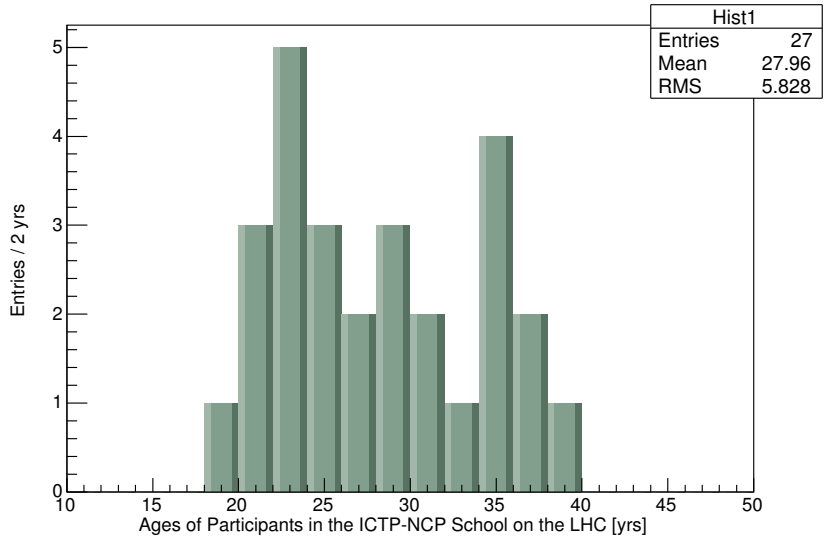
Binning in a Histogram

Visualize number of events in a certain range, called bin.

- Typically bins have an equal size.
- There may also be a histogram with different size bins.

For example, there are 27 participants in this ICTP-NCP school in the age range 18-40 years. We collect the data of ages of these 27 participants:{19,20,21,21,22,23,23,22,23,25,25,25,27,27,29,29,28,31,31,33,35,35,34,34,37,37,39}.

Age (yrs)	Number
18-20	1
20-22	3
22-24	5
24-26	3
26-28	2
28-30	3
30-32	2
32-34	1
34-36	4
36-38	2
38-40	1



TH1F *Hist1 = new TH1F("Hist1","Title ",20, 10, 50);

Let's store this histogram in a root file "aginghisto.root" to use it later.

The statistics box also gives other informations about the distributions.

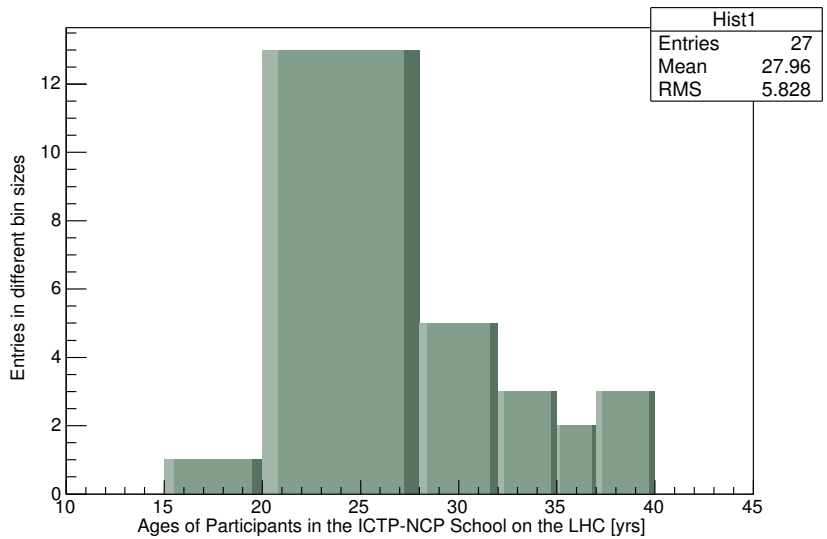
Binning in a Histogram

Visualize number of events in a certain range, called bin.

- There may also be a histogram with different size bins.

For example, there are 27 participants in this ICTP-NCP school in the age range 18-40 years. We collect the data of ages of these 27 participants:{19,20,21,21,22,23,23,22,23,25,25,25,27,27,29,29,28,31,31,33,35,35,34,34,37,37,39}.
This time let's put the data in 8 different-sized bins.

Bin#	Age (yrs)	Number
1	10-15	0
2	15-20	1
3	20-28	13
4	28-32	5
5	32-35	3
6	35-37	1
7	37-40	3
8	40-45	0



TH1F *Hist1 = new TH1F("Hist1","Title ",8, age_bins);

Float_t age_bins[9]={10.,15.,20.,28.,32.,35.,37.,40.,45.};

Except some special cases, we don't consider different bin-widths. Let's go back to the equal bin-width histograms.

Extracting information from a Histogram

- Let's access previously created "aginghisto.root" file:

```
root[ ] TFile *f=new TFile("aginghisto.root");
```

- To the list of contents in above file:

```
root[ ] f.ls()
TFile** aginghisto.root
TFile* aginghisto.root
KEY: TH1F Hist1;1
```

- Now you can draw histogram in prompt:

```
root[ ] Hist1→Draw();
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
```

Now we are ready to extract information from this histogram in root-prompt.

Please also remember that we can do all this in a code.

Extracting information from a Histogram

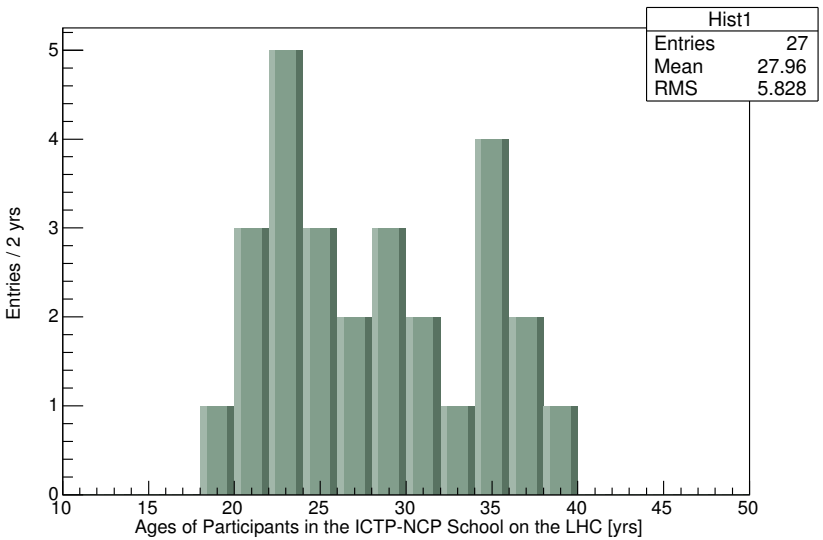
Many informations can be extracted from a histogram.

To get total entries:

```
root[ ] Hist1→GetEntries()
(const Double_t)2.70000000000000000e+01
```

To get total mean:

```
root[ ] Hist1→GetMean()
(const Double_t)2.79629629629629619e+01
```



To get total RMS:

```
root[ ] Hist1→GetRMS()
(const Double_t)5.82765743852970086e+00
```

To find bin number at any x-axis value:

```
root[ ] Hist1→FindBin(25)
(Int_t)8
```

To find bin contents of any bin:

```
root[ ] Hist1→GetBinContent(8)
(const Double_t)3.00000000000000000e+00
```

To find bin center:

```
root[ ] Hist1→GetBinCenter(8)
(const Double_t)2.50000000000000000e+01
```

To get bin error:

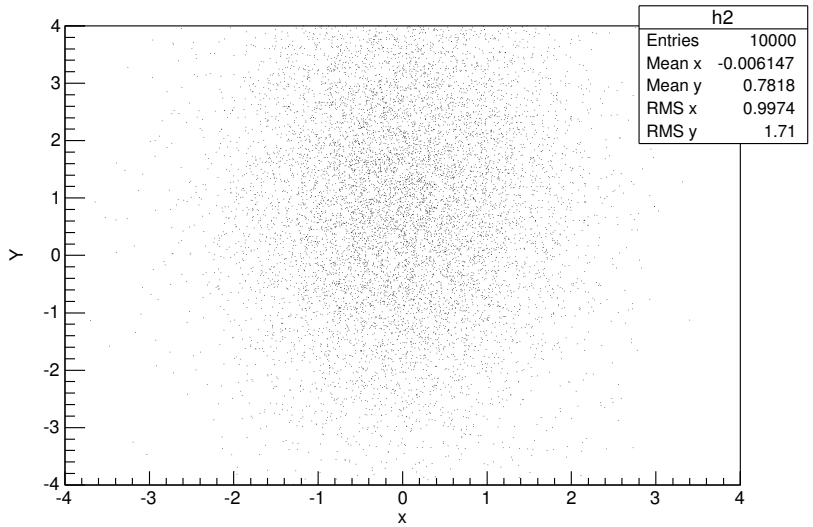
```
root[ ] Hist1→GetBinError(8)
(const Double_t)1.73205080756887719e+00
```

2D histograms

TH2D * h2 = new TH2D("h2,"A 2D Hist,40,-4.,4.,40,-4.,4);

- Filling a 2D histogram for 1000 events is like this:

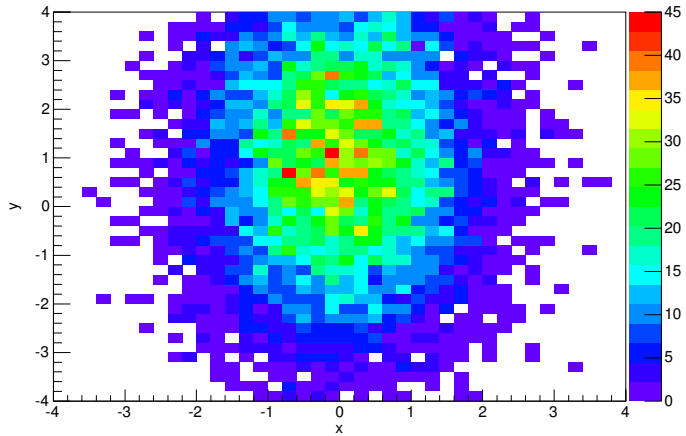
```
for (int i = 0; i<10000; ++i) {
double x = gRandom→Gaus(0,1);
double y = gRandom→Gaus(1,2);
h2→Fill(x,y);
}
```



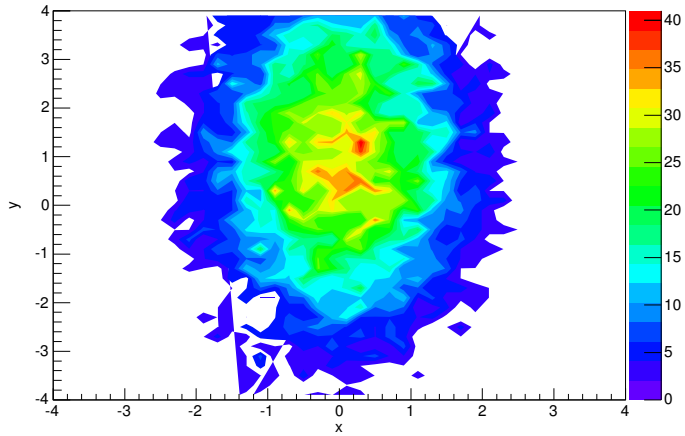
Let's try some famous draw options....

2D histograms

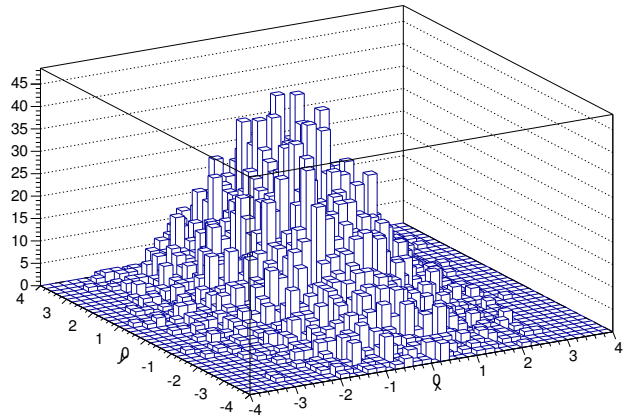
- Draw("colz"):



- Draw("contz"):



- Draw("LEGO"):



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 2D adding weight is like following:

  ```
  h2D→Fill(x, y, 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);
  ```

Exercises

- Filling/Adding/Fitting histograms

Exercise#1 Make a histogram (h1) for 10000 entries containing for a gaussian of mean 3 and sigma 1 from bin 0 to 8 for 80 bins. Make another histogram (h2) for gaussian of mean 2 and sigma 1 for the same binning. Add these two histograms into a histogram (h3).

Exercise#2 Fit a gaussian function to histogram h3 in previous exercise and obtain the value of chi2, mean and sigma from fit.

Exercise#3 Fit two exponential and a gaussian functions to h3 created in the first exercise. Put the gaussian in the center and exponential function on each side. Decide the function ranges by yourself, and assign different colours to each function. Also add legend and fit statistics to the final plot.

- 2D histograms

Exercise#4 If x = gRandom→Rndm()*Gaus(2,3)**ientry*
and y = x = gRandom→Rndm()*Gaus(3,2)**ientry*
where function Rndm() generates random numbers between 0 and 1 and {0<entry<1000}. Then make a 2D histogram of these two variables and find the (x,y) coordinates of highest statistics region. *Hint* : Use Draw("colz") option.

Thanks