The framework ROOT and its use in particle physics

Session 2: Histograms and graphs

Instructor: Gustavo Loachamin

Plan for today

- Today we are going to talk about the graphic classes: TGraph, histograms and functions.
- These objects are used and important for data visualization, qualitative and quantitative analysis of data, properties of data sets, fitting and other functions.

Histograms

- ROOT supports 1D, 2D and 3D histograms.
- Basic methods come from the TH1 class
- Basic constructor:

```
TH1F * h1 = new TH1F("h1", "h1 title", 100, 0.0, 4.0);
TH2F *h2 = new TH2F("h2", "h2 title", 40, 0.0, 2.0, 30, -1.5, 3.5);
 TH3D *h3 = new TH3D("h3", "h3 title", 80, 0.0, 1.0, 100, -2.0, 2.0, 50, 0.0, 3.0);
```

- Filling a histogram:

```
h1->Fill(x);
h1->Fill(x,w); // with weight
h2->Fill(x,y);
h2->Fill(x,y,w);
h3->Fill(x,y,z);
h3->Fill(x,y,z,w);
```

Histograms

- Histograms can also be created with variable width by specifying the bin limits:

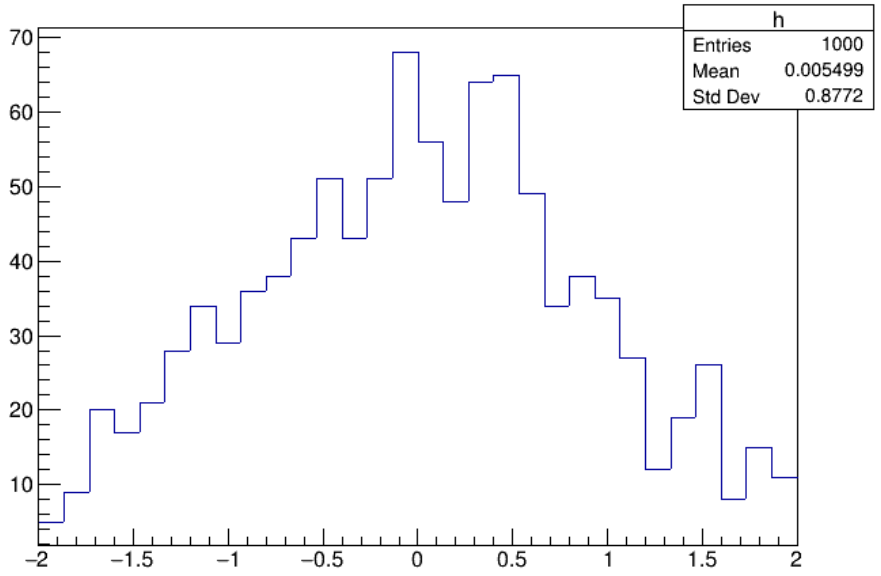
```
const Int_t NBINS = 5;
  Double_t edges[NBINS + 1] = {0.0, 0.2, 0.3, 0.6, 0.8, 1.0};
  // Bin 1 corresponds to range [0.0, 0.2]
  // Bin 2 corresponds to range [0.2, 0.3] etc...

  TH1* h = new TH1D(
     /* name */ "h",
     /* title */ "Hist with variable bin width",
     /* number of bins */ NBINS,
     /* edge array */ edges
   );
```

Histograms

- Histograms can be filled with random numbers from a given distribution. One way to do this is using the method FillRandom.

```
TH1F *h = new TH1F("h", "", 30, -2, 2);
h->FillRandom("gaus", 1000);
```



Histograms operations

- Different operations can be performed using histograms:

```
TH1F *h = new TH1F("h", "", 30, -2, 2);
TH1F *h2 = new TH1F("h2", "", 30, -2, 2);
h2->Add(h);  //add histograms.
h2->Multiply(h); //multiply histograms.
h2->Divide(h); //divide histograms.
h2->Scale(3); // Scale by a constant
```

Draw histograms

- Some drawing options:

“AXIS”: Draw only the axis.

“HIST”: When a histogram has errors, it is visualized by default with error bars. To visualize it without errors use HIST together with the required option (e.g. “HIST SAME C”).

“LEGO”: Draw a lego plot.

“SURF”: Draw a surface plot.

- For 1D histograms:

AH”: Draw the histogram, but not the axis labels and tick marks

“C”: Draw a smooth curve through the histogram bins

“E”: Draw the error bars

“L”: Draw a line through the bin contents

Display stats

mode = ksiourmen (default =000001111)

n = 1 the name of histogram is printed
e = 1 the number of entries
m = 1 the mean value
m = 2 the mean and mean error values
r = 1 the root mean square (RMS)
r = 2 the RMS and RMS error
u = 1 the number of underflows
o = 1 the number of overflows
i = 1 the integral of bins
s = 1 the skewness
s = 2 the skewness and the skewness error
k = 1 the kurtosis
k = 2 the kurtosis and the kurtosis error

- To turn off the stats use h.SetStats(kFALSE)
- Stats option gStyle->SetOptStat(ksiourmen)

Exercise

- We take the file in the github page of the course and plot a histogram.
- We use different drawing options.

Other important operations

- Change axis labels:

```
h->GetXaxis()->SetTitle("X axis title");
h->GetYaxis()->SetTitle("Y axis title");
h->GetZaxis()->SetTitle("Z axis title");
```

- Make copy of histograms:

```
TH1F *hnew = (TH1F*)h->Clone(); //renaming is recommended
hnew->SetName("hnew");
```

- Normalize histograms

```
Double_t scale = 1/h->Integral();
h->Scale(scale);
```

- Save histograms in files

```
h1->Write();
```

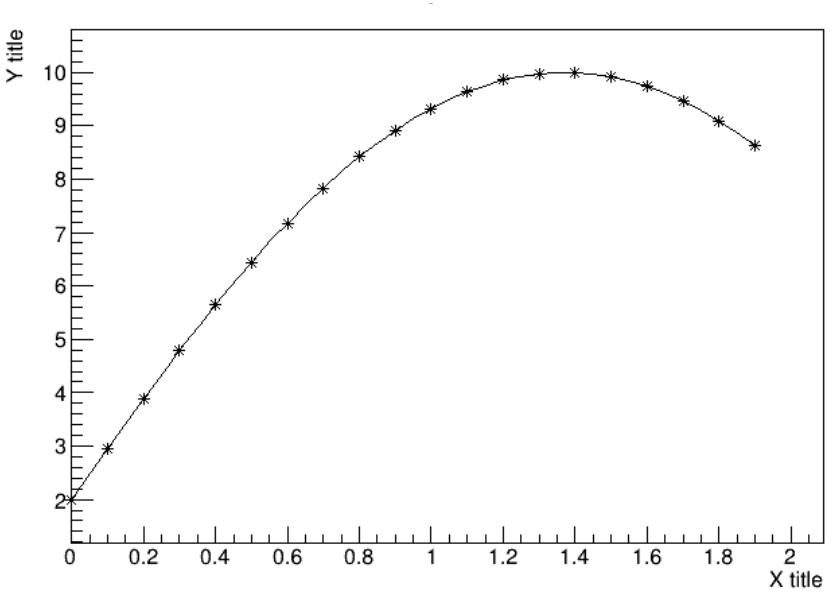
TGraphs

- The basic constructor for Tgraph:

```
auto g = new TGraph(n,x,y);
```

- Ideal for data sets that contain dependent-independent variables. For exampleÑ

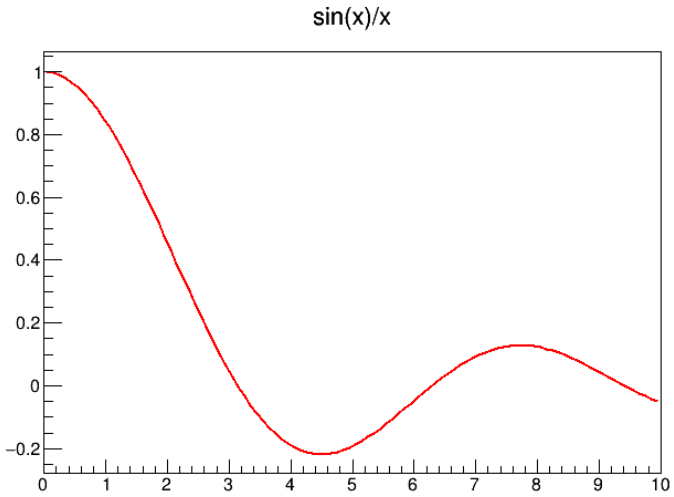
```
double x[100], y[100];
int n = 20;
for (int i=0;i<n;i++) {
  x[i] = i*0.1;
  y[i] = 10*sin(x[i]+0.2);
}
auto g = new TGraph(n,x,y);
g->SetTitle("Graph title;X title;Y title");
g->Draw("AC*");
```



TF classes

- TF define functions between a lower and upper limit.
- They define 1D, 2D and 3D functions.
- The function may have associated parameters.
- TF has the same drawing functionsas TH1 and Tgraph.

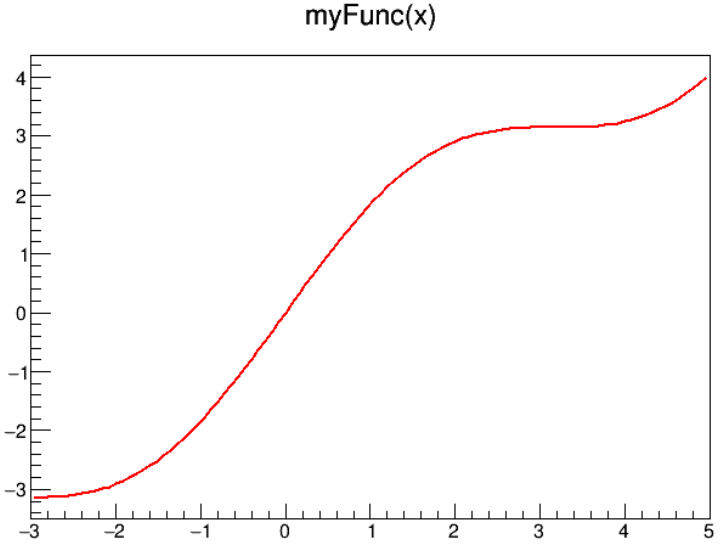
```
TF1 * fa1 = new TF1("fa1","sin(x)/x",0,10);
  fa1->Draw();
```



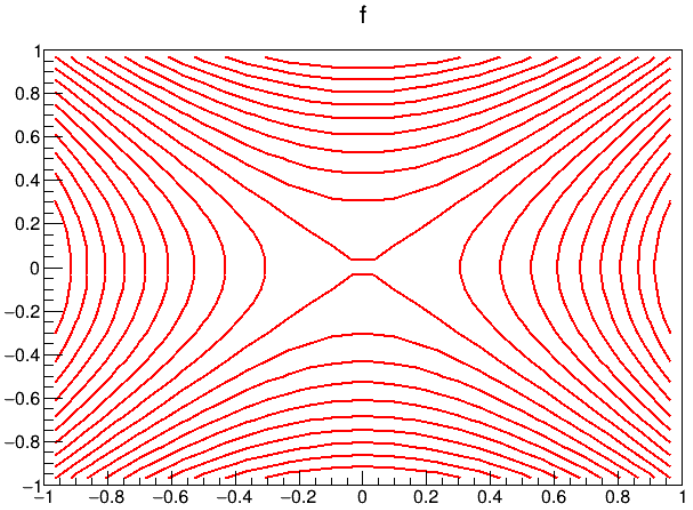
TF classes

- TF functions can also be initialize with user's defined functions:

```
Double_t myFunc(double x) { return x+sin(x);}
TF1 *fa3 = new TF1("fa3","myFunc(x)",-3,5);
fa3->Draw();
```




```
Double_t func(Double_t *val, Double_t *par)
{Float_t x = val[0];
   Float_t y = val[1];
   Double_t f = x*x-y*y;
   return f;
}
void fplot()
{auto f = new TF2("f",func,-1,1,-1,1);
   f->Draw("surf1");
}
```



Classes used in drawing

- TCanvas, TLegend, Tpad, TLine, TArrow, TFrame, TMarker, TPoints.