ROOT Analysis Tool

Introduction

- What is ROOT ?
  - ROOT is an object-oriented C++ analysis package
  - User-compiled code can be called to produce 1-d, 2-d, and 3-d graphics and histograms…

ROOT web-page: http://root.cern.ch/

ROOT can be downloaded from
http://root.cern.ch/twiki/bin/view/ROOT/Download

ROOT Tutorials:

- http://root.cern.ch/root/Tutorials.html
- Babar ROOT Tutorial I
- Babar ROOT Tutorial II

Start the ROOT

Start ROOT:

- type "root"
- (to skip the introductory welcome type "root -l")

For help: type ".?", ".h"

Quit ROOT: type ".q"

A typical ROOT analysis could be:

- Take variables in an n-tuple, make histograms, calculate quantities, and perform fits…
  - How does one make a histogram ?
  - What is an n-tuple ?
  - How are calculations done ?
  - How does one fit ?

Exercise-1

Problem-1: Please write your first ROOT program which is supposed to print "Hello World" to the screen. To start your program (call it first.C) in ROOT, you type simply ".x first.C" or "root –l first.C" on your terminal.

Problem-2: Now the program should ask the person how often it should print "Hello World" to the screen. Use a variable *ntimes* to store this information. Then the program should output the line *ntimes* times. [use for loop]

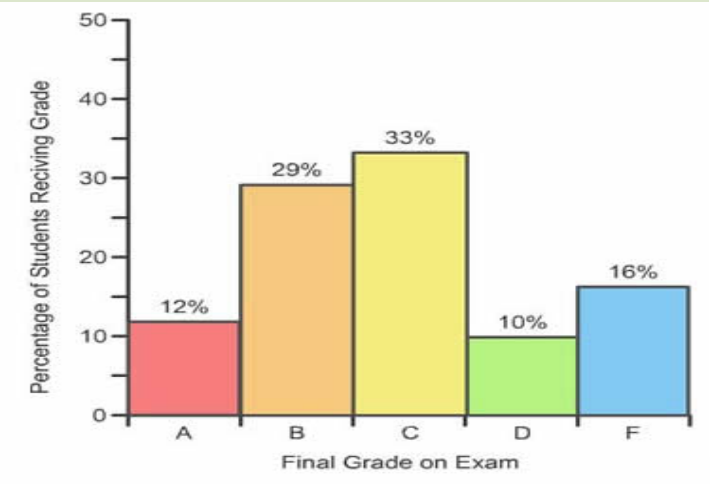
ROOT histograms

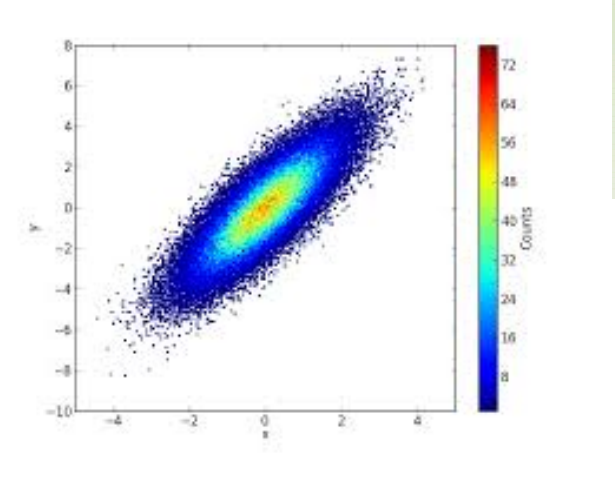
- ✓ A histogram is a display of statistical information to show the frequency of data items in successive numerical intervals, e.g. see the picture below

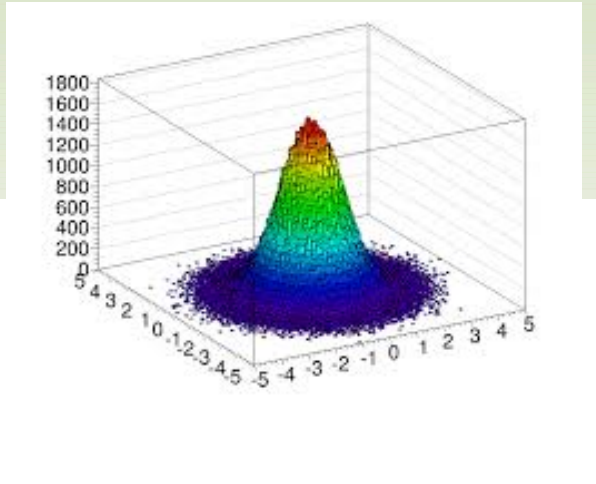
There are several histograms classes available in ROOT, which contain data in the form of a number of (weighted) counts N for a collection of consecutive bins in x (1-D), (x, y) (2-D) or (x, y, z) (3-D). The corresponding classes are

THNS where $N = 1, 2, 3$ for 1-D, 2-D and 3-D and $S = "C, S, I, F, D"$

for 1 (Char_t), 2 (Short_t), 4 (Int_t), 4 (Float_t) or 8 (Double_t) bytes of storage volume per bin.







Example of creating 1D & 2D histograms

```
TH1F *histName = new TH1F("histName", "histTitle", num_bins,x_low,x_high)
```

```
TH1F *my_hist = new TH1F("my_hist", "My First Histogram", 100, 2, 200)
```

Note: Bin 0 → underflow (i.e. entries < x_low)

Bin (num_bins+1) → overflow (i.e. entries > x_high)

2-d and 3-d histograms can be booked similarly…

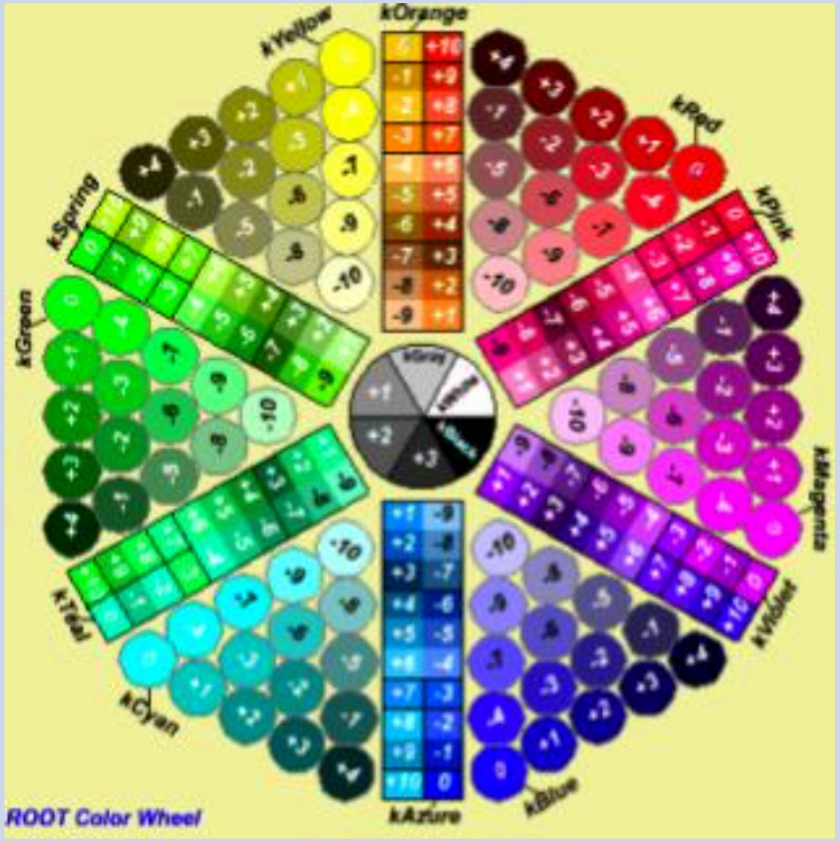
```
TH2F *myhist = new TH2F("myhist", "My Hist", 100, 2, 200, 200, 0,500)
```

Histogram cosmetics

h1.SetMarkerStyle();

h1.SetFillColor();

Histogram cosmetics

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

Histogram informations

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

ROOT Canvas

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.

How to fill histograms

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);

Lets try filling 1D or 2D histograms using Random Number Generator (RNG) in ROOT. What is RNG ? How to fill histogram with RNG ?

Exercise

Problem-3: I assume you know basic C++ programing. Please create an array of data points (integer numbers). Read it from root command line and print it to the screen.

Problem-4: Now fill a 1d histogram with these data points. Play with the histogram by changing linecolor, fillcolor, linewidth of the histogram

Problem-5: Now create a two column text file (data.txt) as follows.

```
// x     y      z
   1     5      10
   2     7      15
   3     14     11
   4     3      11
   5     8      12
   .     .
   .     .
```

Read the text file from your macro, fill and draw the histogram.

Lets generate large sample size using RNG

A random number generator (RNG) is a computer algorithm that can be used to generate a sequence of numbers which appear to be distributed randomly.
Question: How random is it ? After how many events does the sequence start over again ? How fast can the random number be generated ?

Class TRandom : public TNamed;

TRandom(***UInt_t*** seed=65539)

- ❑ The default RNG in ROOT
- ❑ The fastest RNG in ROOT
- ❑ Periodicity = 10^8 events

Class TRandom2 : public TRandom;

TRandom2(***UInt_t*** seed=65539)

- ❑ Slower than TRandom.
- ❑ Periodicity > 10^{14} events

Class TRandom3 : public TRandom;

TRandom3(***UInt_t*** seed=65539)

- ❑ Nearly as fast as TRandom, faster than TRandom2
- ❑ Periodicity = 2^{19937}-1 events

Different distribution using RNG

```
class TRandom : public TNamed {

protected:
   UInt_t   fSeed;  //Random number generator seed

public:
   TRandom(UInt_t seed=65539);
   virtual ~TRandom();
   virtual  Int_t    Binomial(Int_t ntot, Double_t prob);
   virtual  Double_t BreitWigner(Double_t mean=0, Double_t gamma=1);
   virtual  Double_t Exp(Double_t tau);
   virtual  Double_t Gaus(Double_t mean=0, Double_t sigma=1);
   virtual  UInt_t   GetSeed() {return fSeed;}
   virtual  UInt_t   Integer(UInt_t imax);
   virtual  Double_t Landau(Double_t mean=0, Double_t sigma=1);
   virtual  Int_t    Poisson(Double_t mean);
   virtual  Double_t PoissonD(Double_t mean);
   virtual  void     Rannor(Float_t &a, Float_t &b);
   virtual  void     Rannor(Double_t &a, Double_t &b);
   virtual  void     ReadRandom(const char *filename);
   virtual  void     SetSeed(UInt_t seed=65539);
   virtual  Double_t Rndm(Int_t i=0);
   virtual  void     RndmArray(Int_t n, Float_t *array);
   virtual  void     RndmArray(Int_t n, Double_t *array);
   virtual  void     Sphere(Double_t &x, Double_t &y, Double_t &z, Double_t xlong);
   virtual  Double_t Uniform(Double_t x1=1);
   virtual  Double_t Uniform(Double_t x1, Double_t x2);
   virtual  void     WriteRandom(const char *filename);
```

How to use TRandom2/3?

What?	How?
Generate uniformly distributed random number between 0..1	Double_t **Rndm**()
Generate uniformly distributed random number between x1..x2	Double_t **Uniform**(x1, x2)
Generate random numbers according to exponential distribution with slope t	Double_t **Exp**(t)
Generate random numbers according to a Gaussian distribution	Double_t **Gaus**(mean,sigma)
Generate random numbers according to a Poisson distribution	Double_t **Poisson**(mean)

…… (many more)

Comparing different RNG

Try to write this code and compare which one has is faster ?

Create 3 RNGs

Create 3 Histograms

Setup stopwatch

Fill histogram 1 with numbers from RNG 1

Fill histogram 2 with numbers from RNG 2

Fill histogram 3 with numbers from RNG 3

```
void testRandom(Int_t nrEvents=500000000)
{
 TRandom *r1=new TRandom();
 TRandom2 *r2=new TRandom2();
 TRandom3 *r3=new TRandom3();

 TH1D *h1=new TH1D("h1","TRandom",500,0,1);
 TH1D *h2=new TH1D("h2","TRandom2",500,0,1);
 TH1D *h3=new TH1D("h3","TRandom3",500,0,1);

 TStopwatch *st=new TStopwatch();

 st->Start();
 for (Int_t i=0; i<nrEvents; i++) { h1->Fill(r1->Uniform(0,1)); }
 st->Stop(); cout << "Random:  " << st->CpuTime() << endl;

 st->Start();
 for (Int_t i=0; i<nrEvents; i++) { h2->Fill(r2->Uniform(0,1)); }
 st->Stop(); cout << "Random2: " << st->CpuTime() << endl;

 st->Start();
 for (Int_t i=0; i<nrEvents; i++) { h3->Fill(r3->Uniform(0,1)); }
 st->Stop(); cout << "Random3: " << st->CpuTime() << endl;
}
```

Exercise

Problem-3: Use random number generator to make a Gaussian distribution (with mean 0.0 and sigma =0.2) in 1D with range [1., 1.] Generate 100,000 events but draw the histogram in a canvas for every 200 events.

Problem-4: Make 2D histograms with Gaussian RNG in both x and y variables. Generate 100,000 events but draw the histogram in a canvas for every 200 events.

Reading and drawing Histogram from a File

Option 1: in the browser, click on the histogram

Option 2: the pro root [1] new TBrowser ro version

```
root [1] TFile* f = new TFile("/tmp/timeScan_fitcbo27.root")
root [2] TH1D* h = (TH1D*)f->Get("fastRot24")
(class TH1D*)0x8d9fdd8
root [3] h->Draw()
```

Get uses the *name* to look up an object in the file, reads it into memory and returns a pointer to it.

Option 3: the sloppy version

```
root [1] TFile* f = new TFile("/tmp/timeScan_fitcbo27.root")
root [3] fastRot24->Draw()
```

Here, ROOT uses the *name* of the object to find it by itself

```
Double_t scale = norm/hist->Integral();
hist->Scale(scale);
```

Exercise

Problem-5: I am providing you the root file named "test.root" which includes few histograms. Find the object "hBmass" in the file. Print all the info on the screen about the histogram [before that please Change the x-axis range between (5, 5.4) & draw the histogram]

GetBinContent(), GetBinError(), GetMinimum(), GetMaximum(), GetMinimumBin(), GetMaximumBin(), GetEntries(), GetSum(), Integral (bin1, bin2), GetMean(), GetRMS().

Change the FillColor(), SetLineColor() and draw the histogram again.

Problem-6: Lets get two histograms from the test.root (objects named as "hBpt, "hDimupt", then add the two histograms).

How to create a file

```
sanjays-MacBook-Pro-2:SANJAY sanjay$ root
   ******************************************
   *                                        *
   *        W E L C O M E  to  R O O T      *
   *                                        *
   *   Version   5.34/30     23 April 2015  *
   *                                        *
   *  You are welcome to visit our Web site *
   *          http://root.cern.ch           *
   *                                        *
   ******************************************

ROOT 5.34/30 (v5-34-30@v5-34-30, Apr 23 2015, 18:31:46 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TFile *f = new TFile("my.root","recreate")
root [1] TH1F *myhis = new TH1F("my his","A 1d histogram",100, -1., 1.)
root [2] f->Write();
root [3] f->Close();
root [4] .q
sanjays-MacBook-Pro-2:SANJAY sanjay$ root -l
root [0] new TBrowser
```

Fill your 1D histograms (A Gaussian distribution using random number generator). See if the file "my.root" exists in your working directory and does it have appropriate histogram inside the file.

How to fit a 1D histogram

First get the name of the defaulter fitter in ROOT from gEnv(an instance of TEnv class).

```
sanjays-MacBook-Pro-2:SANJAY sanjay$ root
  ********************************************
  *                                          *
  *        W E L C O M E  to  R O O T        *
  *                                          *
  *   Version   5.34/30     23 April 2015    *
  *                                          *
  *  You are welcome to visit our Web site   *
  *          http://root.cern.ch             *
  *                                          *
  ********************************************

ROOT 5.34/30 (v5-34-30@v5-34-30, Apr 23 2015, 18:31:46 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] gEnv->GetValue("Root.Fitter"," ")
(const char* 0x7ff533d15509)"Minuit"
root [1]
```

Lets try to fit a 1D histogram. Then figure out how to get error matrix (what is error matrix or covariance matrix ?)

Fitting Histogram

- ✓ One need to find out what kind of function you have to use to fit the data to.
- ✓ There are many predefined function available in ROOT. For example:

- " `gaus` " Gaussian function with 3 parameters: `f(x) = p0*exp(-0.5*((x-p1)/p2)^2)`
- " `expo` "An Exponential with 2 parameters: `f(x) = exp(p0+p1*x)`
- " `pol` `N` " A polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*x2 +...`
- " `chebyshev` `N` " A Chebyshev polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*(2*x2-1) +...`

- There are additional options for the fit method.
  - hist->Fit("func","opt","gopt",llim,ulim);
- GOpt : Graphical Options.
  - As for plotting histograms
- Opt: Fitting Options
  - L : Likelihood fit (default is χ^2).
  - W : Set all errors to 1
  - E : Use Minos error estimation
  - Q/V :Quiet/ Verbose mode.
  - M : Improve fit result.

User defined function

✓ One can write his own user defined function, such as

```
TF1 *f1 = new TF1("f1","sin(x)/x", 0, 10)
      f1->Draw()
```

One can even add more parameters to the function such as

```
TF1 *f1 = new TF1("f1","[0]*x*sin([1]*x)", -3, 3)
f1->SetParameters(10,5)
f1->Draw()
```

How to write a first order polynomial and Gaussian user-defined function ?

```
Double_t ff1(Double_t *x, Double_t *par){

   return par[0] + par[1]*x[0];
}
```

```
Double_t ff(Double_t *x, Double_t *par){
  if (par[2]!0) arg = ((x[0] - par[1])/par[2]);
  Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
  return fitval;
}
```

Exercise

Problem-7: I am providing you the root file named "test.root" which includes few histograms. Find the object "hBmass" in the file. Fit the distribution with pre-defined Gaussian function in the range 5.2-5.35.

Part-II: Write your own Gaussian function (rather than predefined root function "gaus". Fit the distribution "hBmass" with your own user-defined function in the same range as above.

Part-III: Add a first order polynomial to the user defined function. So, now you have gaussian + poly1 user-defined function. Fit the distribution with both functions in separate ranges, i.e, Gaussian function to be fitted between 5.2- 5.35 and the poly1 to be fitted between 5.35 – 5.45.

Part-IV: Write both user defined function together and call it during your data fit.

Fitting a 1D histogram

```
Double_t myfitfunction(Double_t *x, Double_t *par){
  Double_t peak = par[0]*TMath::Gaus(x[0],par[1],par[2]);
  Double_t background = par[3]+ par[4]*x[0] + par[5]*x[0]*x[0];
  return (peak + background);
}

void zero4(){   //The Macro " Zero4"

  TFile *f = new TFile("./test.root");
  TH1D *h= (TH1D*)f->Get("hBmass");                    //obtain histograms
  h->GetXaxis()->SetRangeUser(5.15, 5.4);


  TF1 *fSpectrum = new TF1("fSpectrum", myfitfunction, 5.15, 5.4, 6); //define
  TF1 *fBackground = new TF1("fBackground","pol2", 5.15, 5.4);        //functions

  fSpectrum->SetParameters(200, 5.24, 0.03, 0,0);  //initialize fit parameters
  h->Fit("fSpectrum");    //fit the histogram

  Double_t fitParameters[6];
  fSpectrum->GetParameters(fitParameters);          //set background function
  fBackground->SetParameters(&fitParameters[3]);

  h->Draw("e");                                    //obtain histograms

  fBackground->SetLineColor(kBlue);fBackground->Draw("same");

}
```

The Fit Function: Gaussian + 2nd order Polynomial