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/HowTo.html
- 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
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                | setenv 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.

```
TObject
 IS A
 Event
 HAS A  HAS A  HAS A
 Segment Track Vertex
 HAS A  HAS A  HAS A
 Momentum MassSquare InterceptAtVert
```
The Framework Organization

```
/* Optional Installation

libCint.so
libCore.so
libEG.so
*libEGPythia.so
*libEGPythia6.so
libEGVenus.so
libGpad.so
libGra.so
libGra6d.so
libGui.so
libGX11.so
*libGX11TTF.so
libHist.so
libHistPainter.so
libHtml.so
libMatrix.so
libMinuit.so
libNew.so
libPhysics.so
libPostscript.so
libProof.so
*libRFIO.so
*libRGL.so
libRint.so
*libThread.so
libTree.so
libTreePlayer.so
libTreeViewer.so
*libttf.so
libX3d.so
libXpm.a

The Framework Organization

$ROOTSYS
`
# User Interfaces

## User Interfaces

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.q</code></td>
<td>Quit</td>
</tr>
<tr>
<td><code>.L macro.C</code></td>
<td>Load a macro file</td>
</tr>
<tr>
<td><code>.x macro.C</code></td>
<td>Load and execute a macro file</td>
</tr>
<tr>
<td><code>.x macro.C++</code></td>
<td>Compile and execute</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>basic.dat</th>
<th>tree.C</th>
</tr>
</thead>
</table>
| -1.102279 -1.799389 4.452822 | `#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();
}`
| 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 |
| ... ... ... |
Read the Root file and the Tree

```
Read the Root file and the Tree

TBrowser

root [0] TBrowser

TFile* basic.root

ntuple> Scain();

root [2] ntuple->Scain()

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.18278</td>
<td>-1.799389</td>
<td>4.4528222</td>
</tr>
<tr>
<td>1</td>
<td>1.867179</td>
<td>-0.596621</td>
<td>3.8423130</td>
</tr>
<tr>
<td>2</td>
<td>-8.524181</td>
<td>1.8665829</td>
<td>3.7661398</td>
</tr>
<tr>
<td>3</td>
<td>-0.380611</td>
<td>0.9691280</td>
<td>1.0848740</td>
</tr>
<tr>
<td>4</td>
<td>0.5524539</td>
<td>-0.213289</td>
<td>0.358281</td>
</tr>
<tr>
<td>5</td>
<td>-0.184954</td>
<td>1.1870849</td>
<td>1.4493020</td>
</tr>
<tr>
<td>6</td>
<td>0.2056429</td>
<td>-0.767014</td>
<td>0.6354169</td>
</tr>
<tr>
<td>7</td>
<td>1.8792219</td>
<td>-0.327389</td>
<td>1.2719039</td>
</tr>
<tr>
<td>8</td>
<td>-0.27491</td>
<td>-1.721428</td>
<td>3.0388989</td>
</tr>
<tr>
<td>9</td>
<td>2.0477790</td>
<td>-0.062677</td>
<td>4.1973290</td>
</tr>
<tr>
<td>10</td>
<td>-0.45867</td>
<td>-1.443218</td>
<td>2.2932660</td>
</tr>
<tr>
<td>11</td>
<td>0.3047310</td>
<td>-0.884635</td>
<td>0.8754420</td>
</tr>
<tr>
<td>12</td>
<td>-0.712336</td>
<td>-0.223291</td>
<td>0.5668101</td>
</tr>
<tr>
<td>13</td>
<td>-0.271865</td>
<td>1.1817669</td>
<td>1.4704840</td>
</tr>
<tr>
<td>14</td>
<td>0.8862019</td>
<td>-0.65106</td>
<td>1.2132090</td>
</tr>
<tr>
<td>15</td>
<td>-2.035522</td>
<td>0.5276479</td>
<td>4.4128310</td>
</tr>
<tr>
<td>16</td>
<td>-1.495046</td>
<td>-0.463997</td>
<td>2.341131</td>
</tr>
<tr>
<td>17</td>
<td>1.2386160</td>
<td>-0.005650</td>
<td>1.545590</td>
</tr>
<tr>
<td>18</td>
<td>0.8878769</td>
<td>1.8853290</td>
<td>3.562369</td>
</tr>
<tr>
<td>19</td>
<td>-0.314153</td>
<td>-0.329160</td>
<td>0.2878399</td>
</tr>
<tr>
<td>20</td>
<td>-0.198253</td>
<td>0.6460700</td>
<td>0.4567120</td>
</tr>
<tr>
<td>21</td>
<td>-1.636217</td>
<td>1.0495510</td>
<td>3.7787621</td>
</tr>
<tr>
<td>22</td>
<td>1.2211989</td>
<td>0.8143829</td>
<td>2.1543269</td>
</tr>
<tr>
<td>23</td>
<td>1.4131350</td>
<td>1.5493369</td>
<td>4.3989419</td>
</tr>
<tr>
<td>24</td>
<td>-0.174493</td>
<td>-1.339397</td>
<td>1.8018410</td>
</tr>
</tbody>
</table>
```

Type <CR> to continue or q to quit =>

TBrowser t;

root [3] .ls

```
root [3] .ls basic.root
```

TFile* basic.root

KEY: TH1F h1xy x distribution

KEY: TNtuple ntuplo1 data from ascii file

root [4] ntuple->Print()

```
root [4] ntuple->Print()
```

```
Tree : ntuple : data from ascii file
*Entries : 1000 : Total = 14144 bytes File Size = 11792
*Tree compression factor = 1.00

*Entries : 1000 : Total Size= 4604 bytes One basket in memory
*Baskets : 0 : Basket Size= 32000 bytes Compression= 1.00
```

Type <CR> to continue or q to quit =>

```
Type <CR> to continue or q to quit =>
```
### What do we learn from this Macro

<table>
<thead>
<tr>
<th>How to</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a Root file?</strong></td>
<td><code>TFile *f = new TFile(&quot;basic.root&quot;,&quot;RECREATE&quot;);</code></td>
</tr>
<tr>
<td><strong>//option:</strong> NEW, CREATE, RECREATE, UPDATE, or READ</td>
<td></td>
</tr>
<tr>
<td><strong>//Book and fill histograms and trees</strong></td>
<td><code>//-----</code></td>
</tr>
<tr>
<td></td>
<td><code>f-&gt;Write();</code> //write the file</td>
</tr>
<tr>
<td></td>
<td><code>f-&gt;Close();</code> //close the file</td>
</tr>
<tr>
<td><strong>Book and fill a histogram?</strong></td>
<td><code>TH1F *h1 = new TH1F(&quot;h1&quot;,&quot;x distribution&quot;,100,-4,4);</code></td>
</tr>
<tr>
<td></td>
<td><code>/*do some calculation and get the parameter that you want to fill*/</code></td>
</tr>
<tr>
<td></td>
<td><code>h1-&gt;Fill(x);</code></td>
</tr>
<tr>
<td><strong>Book and fill a tree ?</strong></td>
<td><code>TNtuple *ntuple = new TNtuple(&quot;ntuple&quot;,&quot;data from ascii file&quot;,&quot;x:y:z&quot;);</code></td>
</tr>
<tr>
<td></td>
<td><code>/*do some calculation and get the parameter that you want to fill*/</code></td>
</tr>
<tr>
<td></td>
<td><code>ntuple-&gt;Fill(x,y,z);</code></td>
</tr>
<tr>
<td><strong>CINT Data types</strong></td>
<td><code>Int_t and Float_t</code></td>
</tr>
</tbody>
</table>
|                                             | (see http://root.cern.ch/root/html/ListOfTypes.html ) )
## Root Data Types

<table>
<thead>
<tr>
<th>C++ type</th>
<th>Size (bytes)</th>
<th>ROOT types</th>
<th>Size (bytes)</th>
<th>FORTRAN analog</th>
</tr>
</thead>
<tbody>
<tr>
<td>(unsigned)char</td>
<td>1</td>
<td>(U)Char_t</td>
<td>1</td>
<td>CHARACTER*1</td>
</tr>
<tr>
<td>(unsigned)short (int)</td>
<td>2</td>
<td>(U)Short_t</td>
<td>2</td>
<td>INTEGER*2</td>
</tr>
<tr>
<td>(unsigned)int</td>
<td>2 or 4</td>
<td>(U)Int_t</td>
<td>4</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>(unsigned)long (int)</td>
<td>4 or 8</td>
<td>(U)Long_t</td>
<td>8</td>
<td>INTEGER*8</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
<td>Float_t</td>
<td>4</td>
<td>REAL*4</td>
</tr>
<tr>
<td>double</td>
<td>8 (=4)</td>
<td>Double_t</td>
<td>8</td>
<td>REAL*8</td>
</tr>
<tr>
<td>long double</td>
<td>16 (= double)</td>
<td></td>
<td></td>
<td>REAL*16</td>
</tr>
</tbody>
</table>
## How to Compile a Macro?

### Running the macro

```
root [0] .x tree.C or,
root [0] .L tree.C
root [1] tree()
```

### Compiling the macro

```
root [0] .L tree1.C++
root [1] main()
```

```c
#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();
}
```
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1.GetMean()</td>
<td>Mean</td>
</tr>
<tr>
<td>h1.GetRMS()</td>
<td>Root of Variance</td>
</tr>
<tr>
<td>h1.GetMaximum();</td>
<td>Maximum bin content</td>
</tr>
<tr>
<td>h1.GetMaximumBin(int bin_number);</td>
<td>location of maximum</td>
</tr>
<tr>
<td>h1.GetBinCenter(int bin_number);</td>
<td>Center of bin</td>
</tr>
<tr>
<td>h1.GetBinContent(int bin_number);</td>
<td>Content of bin</td>
</tr>
</tbody>
</table>
## Histogram Cosmetics

<table>
<thead>
<tr>
<th>h1.SetMarkerStyle();</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Marker Styles" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Colors" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h1.SetFillColor();</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Fill Colors" /></td>
</tr>
</tbody>
</table>
## Histogram cosmetics: Lines

<table>
<thead>
<tr>
<th>LineStyle</th>
<th>LineColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1-&gt;SetLineStyle();</td>
<td>h1.SetLineColor();</td>
</tr>
</tbody>
</table>

![ROOT Color Wheel](image_url)
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>c1 = new TCanvas(&quot;c1&quot;,&quot;Title, w, h&quot;)</code></td>
<td>Creates a new canvas with width equal to ( w ) number of pixels and height equal to ( h ) number of pixels.</td>
</tr>
<tr>
<td><code>c1-&gt;Divide(2,2);</code></td>
<td>Divides the canvas to 4 pads.</td>
</tr>
<tr>
<td><code>c1-&gt;cd(3)</code></td>
<td>Select the 3(^{rd}) Pad</td>
</tr>
</tbody>
</table>
| `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 [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

AC*

AB

AF

AL
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

- \text{Float_t x[n]} = \{-0.22, 0.05, 0.25, 0.35, 0.5, 0.61, 0.7, 0.85, 0.89, 0.95\};
- \text{Float_t y[n]} = \{1, 2.9, 5.6, 7.4, 9, 9.6, 8.7, 6.3, 4.5, 1\};
- \text{Float_t ex[n]} = \{0.05, 1, 0.07, 0.07, 0.04, 0.05, 0.06, 0.07, 0.08, 0.05\};
- \text{Float_t ey[n]} = \{0.8, 0.7, 0.6, 0.5, 0.4, 0.4, 0.5, 0.6, 0.7, 0.8\};
- \text{gr = new TGraphErrors(n, x, y, ex, ey);}
• 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

<table>
<thead>
<tr>
<th>root [2] T-&gt;Print()</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Tree :T</td>
</tr>
<tr>
<td>*Entries : 1000</td>
</tr>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

| *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 |
```c
T->Scan(); // scan the Root rows and columns
```

```
<table>
<thead>
<tr>
<th>Root [3] T-&gt;Scan()</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Row * x * x * x *</td>
</tr>
<tr>
<td>0 * -1.102278 * -1.102278 * -1.102278 *</td>
</tr>
<tr>
<td>1 * 1.8671779 * 1.8671779 * 1.8671779 *</td>
</tr>
<tr>
<td>2 * -0.524181 * -0.524181 * -0.524181 *</td>
</tr>
<tr>
<td>3 * -0.380611 * -0.380611 * -0.380611 *</td>
</tr>
<tr>
<td>4 * 0.5524539 * 0.5524539 * 0.5524539 *</td>
</tr>
<tr>
<td>5 * -0.184954 * -0.184954 * -0.184954 *</td>
</tr>
<tr>
<td>6 * 0.2056429 * 0.2056429 * 0.2056429 *</td>
</tr>
<tr>
<td>7 * 1.0792219 * 1.0792219 * 1.0792219 *</td>
</tr>
<tr>
<td>8 * -0.274919 * -0.274919 * -0.274919 *</td>
</tr>
<tr>
<td>9 * 2.0477790 * 2.0477790 * 2.0477790 *</td>
</tr>
<tr>
<td>10 * -0.458676 * -0.458676 * -0.458676 *</td>
</tr>
<tr>
<td>11 * 0.3047310 * 0.3047310 * 0.3047310 *</td>
</tr>
<tr>
<td>12 * -0.712336 * -0.712336 * -0.712336 *</td>
</tr>
<tr>
<td>13 * -0.271865 * -0.271865 * -0.271865 *</td>
</tr>
<tr>
<td>14 * 0.8862019 * 0.8862019 * 0.8862019 *</td>
</tr>
<tr>
<td>15 * -2.035552 * -2.035552 * -2.035552 *</td>
</tr>
<tr>
<td>16 * -1.459046 * -1.459046 * -1.459046 *</td>
</tr>
<tr>
<td>17 * 1.2306610 * 1.2306610 * 1.2306610 *</td>
</tr>
<tr>
<td>18 * 0.0887869 * 0.0887869 * 0.0887869 *</td>
</tr>
<tr>
<td>19 * -0.314153 * -0.314153 * -0.314153 *</td>
</tr>
<tr>
<td>20 * -0.198253 * -0.198253 * -0.198253 *</td>
</tr>
<tr>
<td>21 * -1.636217 * -1.636217 * -1.636217 *</td>
</tr>
<tr>
<td>22 * 1.2211090 * 1.2211090 * 1.2211090 *</td>
</tr>
<tr>
<td>23 * 1.4131350 * 1.4131350 * 1.4131350 *</td>
</tr>
<tr>
<td>24 * -1.174493 * -1.174493 * -1.174493 *</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-&gt;Print();</td>
<td>Prints the content of the tree</td>
</tr>
<tr>
<td>T-&gt;Scan();</td>
<td>Scans the rows and columns</td>
</tr>
<tr>
<td>T-&gt;Draw(“x”);</td>
<td>Draw a branch of tree</td>
</tr>
<tr>
<td>How to apply cuts: T-&gt;Draw(“x”,”x&gt;0”); T-&gt;Draw(“x”,”x&gt;0 &amp; &amp; y&gt;0”);</td>
<td>Draw “x” when “x&gt;0” Draw “x” when both x &gt;0 and y &gt;0</td>
</tr>
<tr>
<td>T-&gt;Draw(“y”,”,””same”);</td>
<td>Superimpose “y” on “x”</td>
</tr>
<tr>
<td>T-&gt;Draw(“y:x”);</td>
<td>Make “y vs x” 2d scatter plot</td>
</tr>
<tr>
<td>T-&gt;Draw(“z:y:x”);</td>
<td>Make “z:y:x” 3d plot</td>
</tr>
<tr>
<td>T-&gt;Draw(“sqrt(x<em>x+y</em>y)”);</td>
<td>Plot calculated quantity</td>
</tr>
<tr>
<td>T-&gt;Draw(“x&gt;&gt;h1”);</td>
<td>Dump a root branch to a histogram</td>
</tr>
</tbody>
</table>
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");
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

Example of several fits in subranges

<table>
<thead>
<tr>
<th></th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entries</td>
<td>49</td>
</tr>
<tr>
<td>Mean</td>
<td>104.6</td>
</tr>
<tr>
<td>RMS</td>
<td>14.3</td>
</tr>
<tr>
<td>$\chi^2$/ndf</td>
<td>0.0848/7</td>
</tr>
<tr>
<td>Constant</td>
<td>4.967 ± 2.838</td>
</tr>
<tr>
<td>Mean</td>
<td>95.47 ± 12.41</td>
</tr>
<tr>
<td>Sigma</td>
<td>6.829 ± 7.572</td>
</tr>
</tbody>
</table>
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

Lorentzian Peak on Quadratic Background

- **Data**
- **Background fit**
- **Signal fit**
- **Global Fit**
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