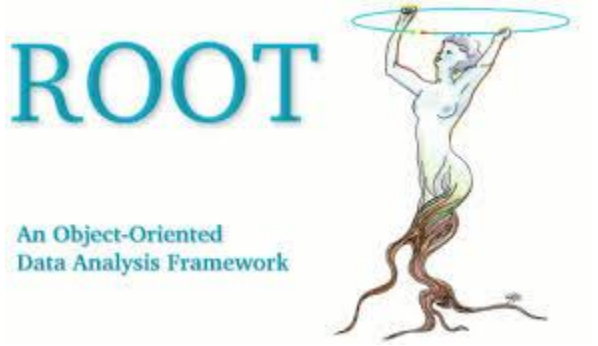


ROOT: Ntuple and Trees

Harinder Singh Bawa

California State University Fresno

April 27, 2016

Review of Exercise

Exercise 1

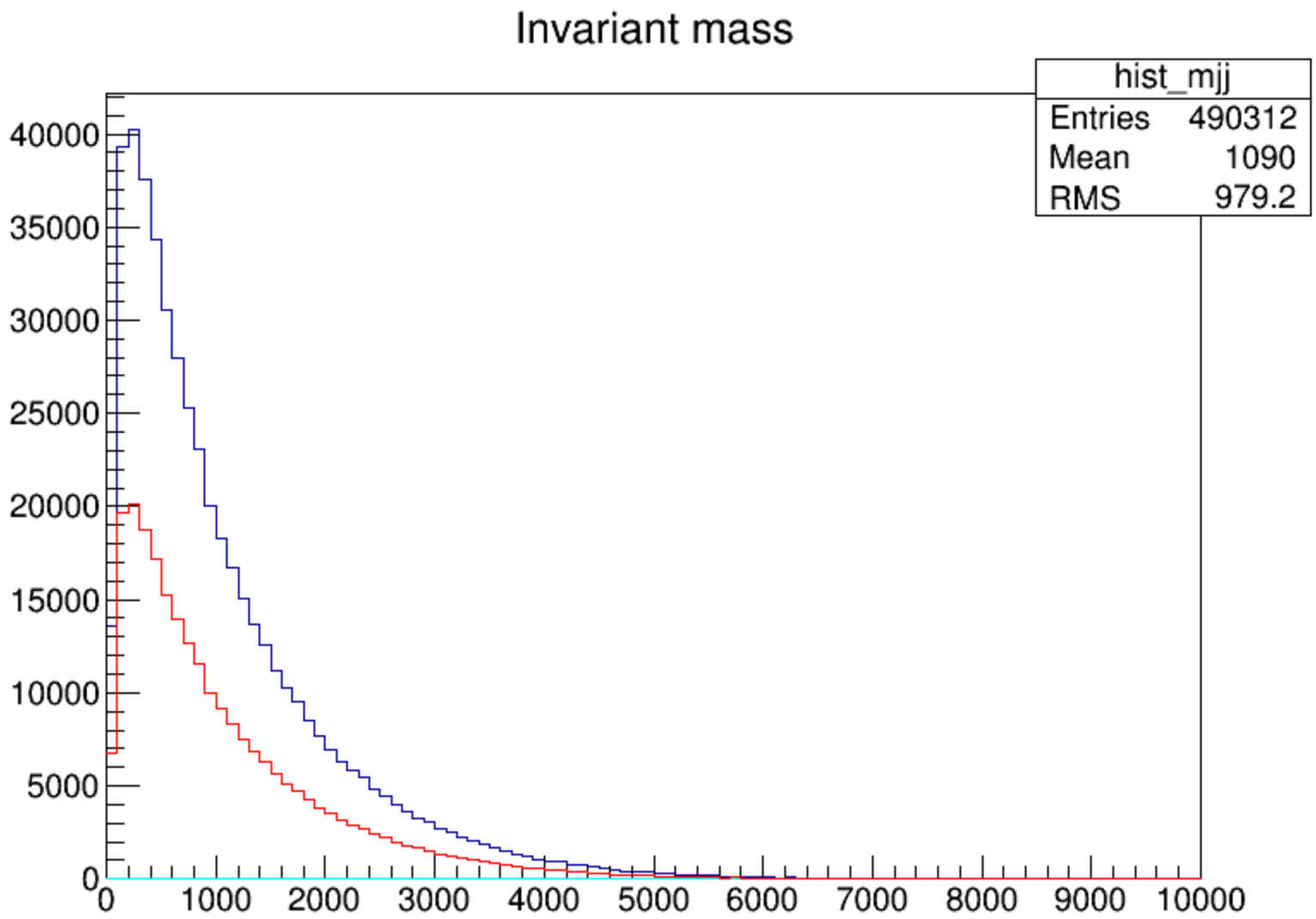
1. Download http://zimmer.fresnostate.edu/~hbawa/rootfiles/OutputJets.root
2. List the histograms inside the files
3. Get the histogram “hist_mjj” and draw on Canvas
4. Clone the histogram and scale to to 0.5 and Draw on the same canvas
5. Normalize the histogram to 1 and draw on the same canvas

Exercise 2

1. Download http://zimmer.fresnostate.edu/~hbawa/rootfiles/OutputJets.root
2. List the histograms inside the files
3. Add the histograms “histo_leadingjet” & “histo_subleadingjet” and draw on Canvas
4. Subtract the histogram “histo_leadingjet” from “histo_jetpt”
5. Save the histograms in outputfile in a separate directory called “NewHistos”

Exercise 1

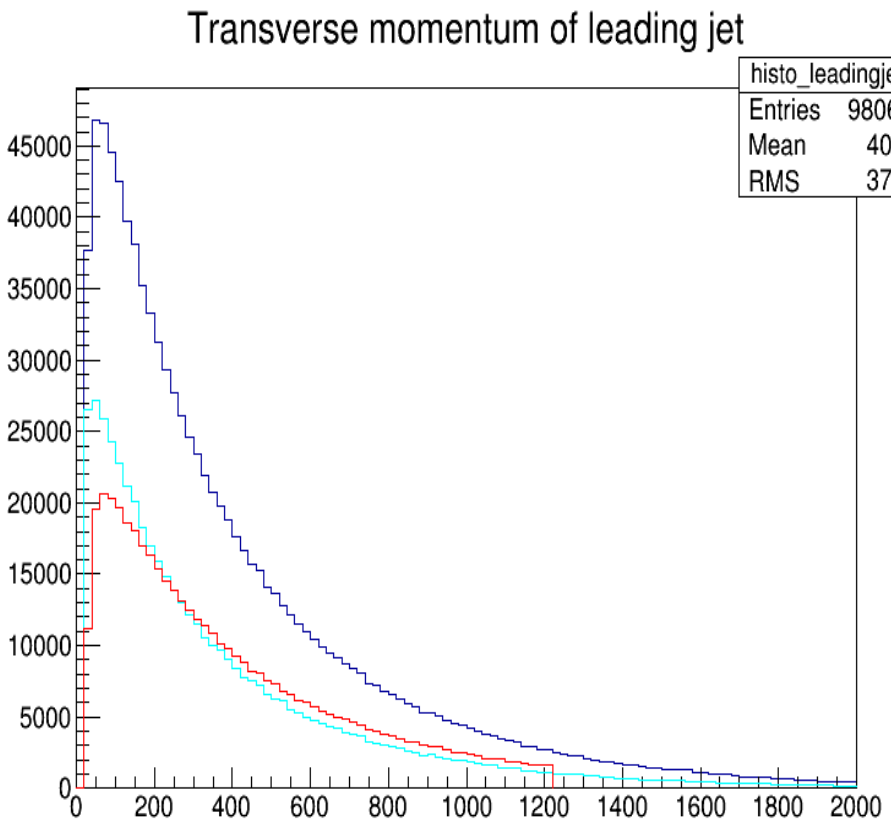
```
#include "TCanvas.h"
#include "TMath.h"
#include <TH1.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TDirectory.h>
#include<iostream>
#include "TFile.h"
#include "TH1F.h"

void Ap13(){
//in terminal, this was simply TFile f("OutputJets.root")
TCanvas *c1 = new TCanvas("c1", "Mjj", 800,600);
TFile* infile = new TFile("OutputJets.root");
//Get the histogram from the input root file
TH1F* histogram_mjj;
infile->GetObject("hist_mjj", histogram_mjj);
histogram_mjj->SetDirectory(0);
//Cloning the input read histogram
TH1F* hist_mjj_half = (TH1F*)hist_mjj->Clone(); hist_mjj_half->SetDirectory(0);
TH1F* hist_mjj_norm = (TH1F*)hist_mjj->Clone(); hist_mjj_norm->SetDirectory(0);
//Scale to 0.5
hist_mjj_half->Scale(0.5);
//Normalise to 1
hist_mjj_norm->Scale(1.0/(hist_mjj_norm->Integral()));
hist_mjj->Draw();
hist_mjj_norm->Draw("same");
hist_mjj_half->Draw("same");
hist_mjj_half->SetLineColor(kRed);
hist_mjj_norm->SetLineColor(kCyan);
```

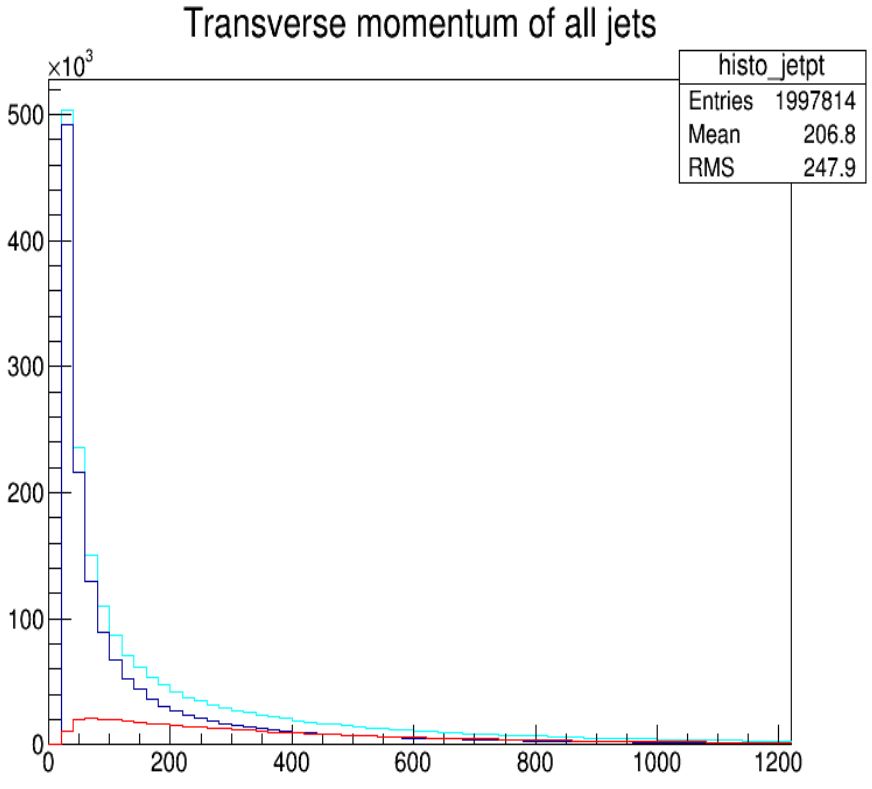

Invariant mass
hist_mjj
Entries 490312
Mean 1090
RMS 979.2
40000
35000
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Exercise 2

Continuing from previous code

```
TCanvas *c1 = new TCanvas("c2", "LeadingandSub", 800,600);
TH1F* histo = (TH1F*)histo_leadingjetpt->Clone(); hist_leadingjetpt->SetDirectory(0);
//Adding histogram
histo->Add(histo_subleadingjetpt);
histo->Draw();
//Superimposing now
histo_subleadingjetpt->Draw("same");
histo_leadingjetpt->Draw("same");
histo_subleadingjetpt->SetLineColor(kCyan);
histo_leadingjetpt->SetLineColor(kRed);

TCanvas *c3 = new TCanvas("c3", "MinusLeading", 800,600);
//Subtracting now
TH1F* histo2 = (TH1F*)histo_jetpt->Clone(); hist_jetpt->SetDirectory(0);
histo2->Add(histo_leadingjetpt, -1);
histo_jetpt->SetLineColor(kCyan);
histo_jetpt->Draw();
histo2->Draw("same");
histo_leadingjetpt->SetLineColor(kRed);
histo_leadingjetpt->Draw("same");

return ();
}
```


Transverse momentum of leading jet
histo_leadingje
Entries 980
Mean 40
RMS 37
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000

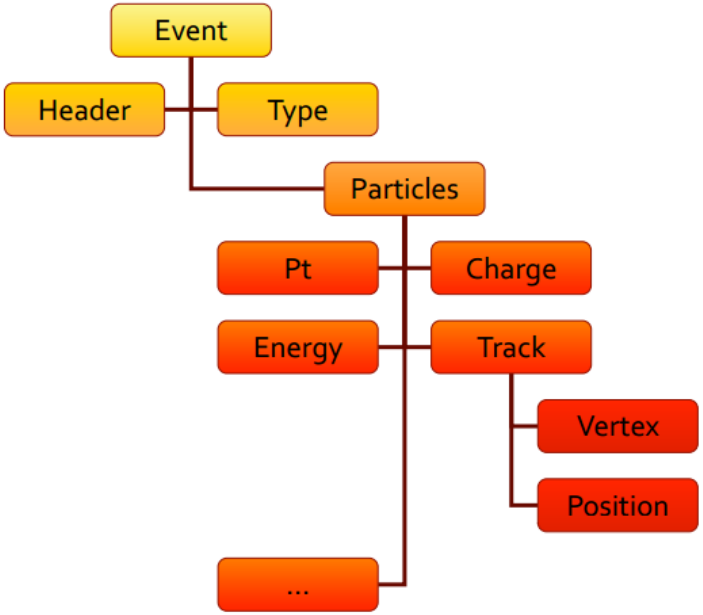

Transverse momentum of all jets
×10^3
histo_jetpt
Entries 1997814
Mean 206.8
RMS 247.9
500
400
300
200
100
0
0
200
400
600
800
1000
1200

Ntuples and Trees

- Ntuple class:
  - `TNtuple`
    - for storing tabular data
    - e.g. Excel Table with numbers

x	y	z
-1.10228	-1.79939	4.452822
1.867178	-0.59662	3.842313
-0.52418	1.868521	3.766139
-0.38061	0.969128	1.084074
0.552454	-0.21231	0.350281
-0.18495	1.187305	1.443902
0.205643	-0.77015	0.635417
1.079222	-0.32739	1.271904
-0.27492	-1.72143	3.038899
2.047779	-0.06268	4.197329
-0.45868	-1.44322	2.293266
0.304731	-0.88464	0.875442
-0.71234	-0.22239	0.556881
-0.27187	1.181767	1.470484
0.886202	-0.65411	1.213209
-2.03655	0.527648	4.421883
-1.45905	-0.464	2.344113
1.230661	-0.00565	1.514559
		3.562347

- Tree class
  - `TTree`
    - for storing complex data types
    - e.g. DataBase tables



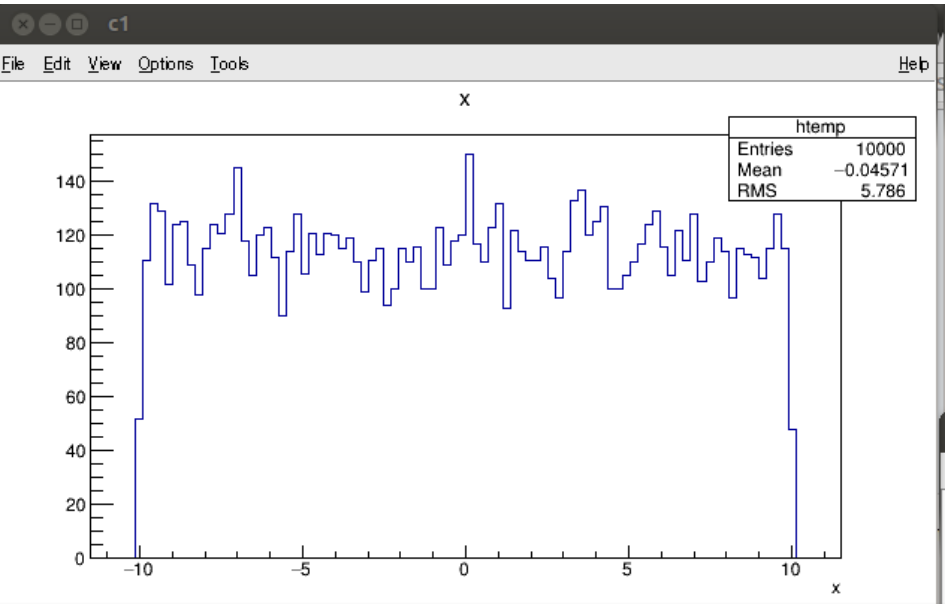
Ntuples

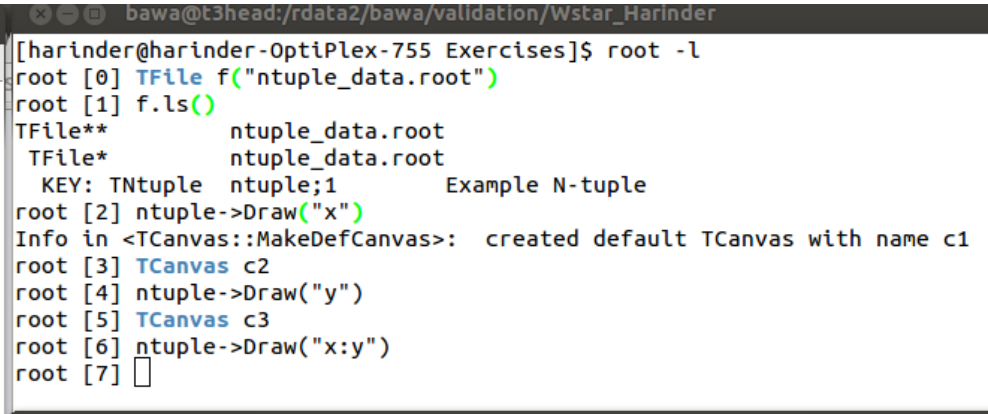
```
#include "TRandom.h"
#include "TFile.h"
#include "TNtuple.h"

void test() {

   TNtuple data("ntuple","Example N-tuple","x:y:z:t");

// fill it with random data
   for (int i = 0; i<10000; ++i) {
      float x = gRandom->Uniform(-10,10);
      float y = gRandom->Gaus(0,5);
      float z = gRandom->Exp(10);
      float t = gRandom->Landau(0,2);

      data.Fill(x,y,z,t);
   }
// write in a file
   TFile f("ntuple_data.root","RECREATE");
   data.Write();
   f.Close();

}
```

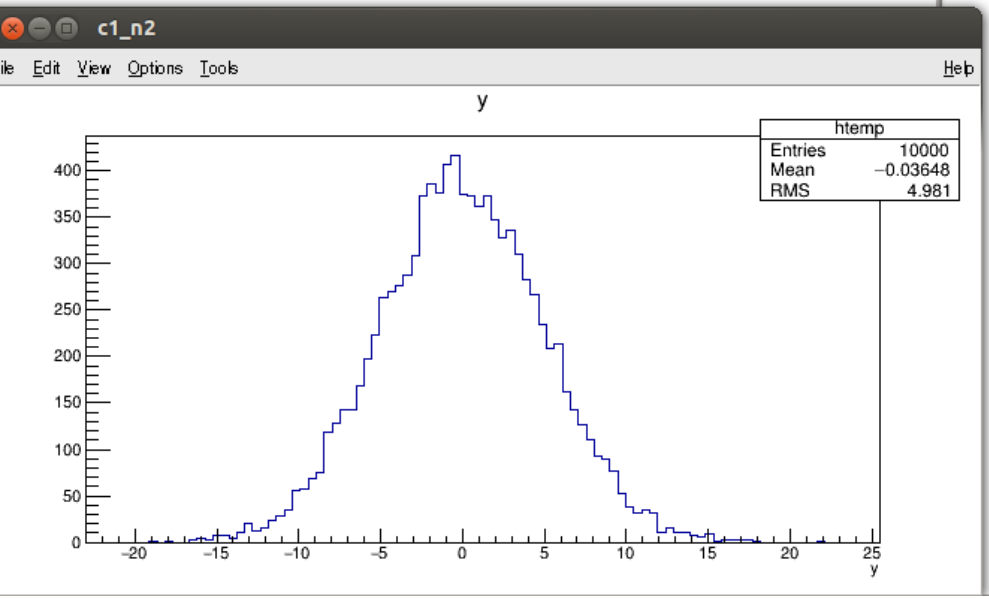
Ntuple: Scan

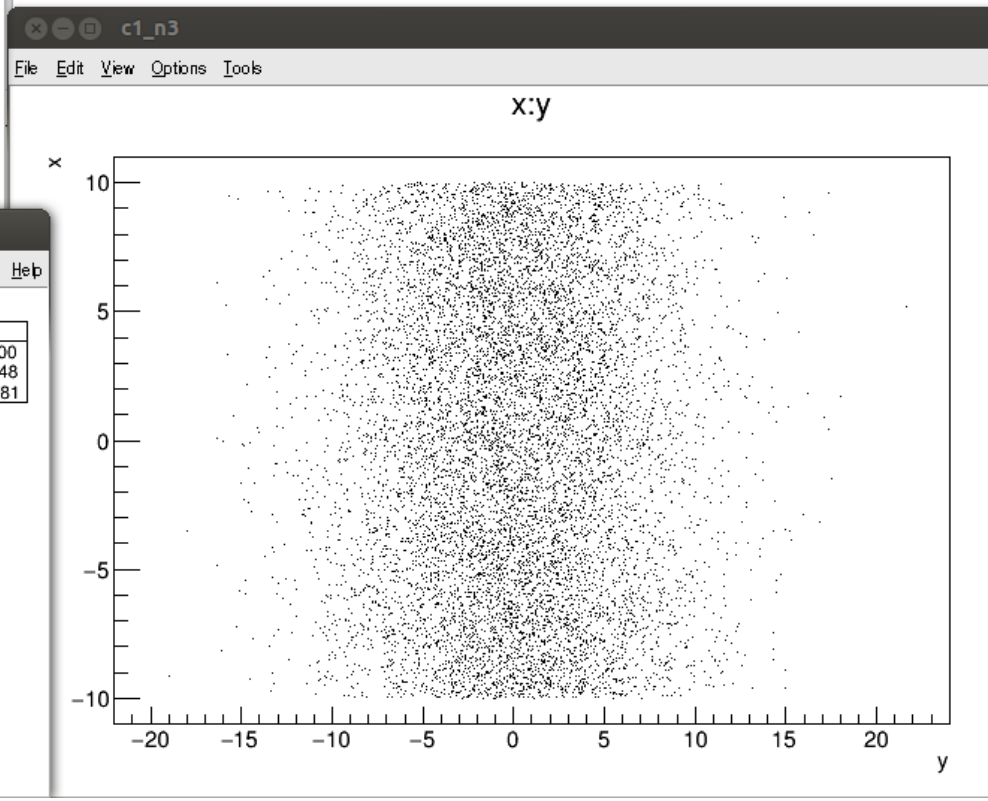
```
[harinder@harinder-OptiPlex-755 Exercises]$ root -l
root [0] TFile f("ntuple_data.root")
root [1] f.ls()
TFile**         ntuple_data.root
 TFile*         ntuple_data.root
  KEY: TNtuple  ntuple;1        Example N-tuple
root [2] ntuple->Scan()
************************************************************
*    Row   *         x *         y *         z *         t *
************************************************************
*        0 * 9.9948349 * -2.173821 * 0.5424387 * -0.612444 *
*        1 * -0.300527 * 4.1213183 * 2.9530391 * 3.3599093 *
*        2 * 4.7990598 * 0.0395610 * 4.1758332 * 0.3248392 *
*        3 * 6.0880603 * -4.925330 * 17.803897 * 2.3463683 *
*        4 * -2.153720 * -2.868095 * 34.954464 * 0.5302660 *
*        5 * -6.117022 * 3.8369829 * 5.4484496 * 22.924655 *
*        6 * 3.3112785 * 10.304511 * 12.156219 * -1.950201 *
*        7 * -8.741646 * -2.271012 * 3.2100651 * 5.3128857 *
*        8 * 4.2777013 * 9.2215652 * 3.5772235 * -2.077455 *
*        9 * -7.415144 * 11.302228 * 24.868108 * -1.812724 *
*       10 * 0.9474282 * 7.6234602 * 12.305172 * 21.557962 *
*       11 * -5.457643 * -6.407330 * 9.1531086 * 6.6468873 *
*       12 * -6.586581 * -5.453365 * 11.670097 * 2.8072593 *
*       13 * -6.351783 * -12.29070 * 18.275390 * 2.1263477 *
*       14 * 2.5567293 * -1.661886 * 3.5925467 * 2.3901383 *
*       15 * -8.197848 * -5.952137 * 8.2887077 *  2.333956 *
*       16 * -1.615081 * 0.4030978 * 4.7062692 * 109.15314 *
*       17 * -2.591395 * 1.5039703 * 19.603691 * 8.6652317 *
*       18 * 6.5947938 * 1.2195639 * 10.592968 * -3.074309 *
*       19 * 2.2613263 * -3.939634 * 6.9536156 * 7.8710861 *
*       20 * -7.108989 * -6.082385 * 11.686473 * -1.895370 *
*       21 * -5.158606 * -0.598714 * 5.1116585 * 5.4938359 *
*       22 * -1.347475 * -4.175319 * 3.3290097 * 5.4394669 *
*       23 * 3.7155344 * 8.9050216 * 20.961120 * -2.862777 *
*       24 * 6.4014411 * -1.842549 * 17.116537 * -1.637794 *
```

Ntuple: Draw()


c1
File Edit View Options Tools Help
x
htemp
Entries 10000
Mean −0.04571
RMS 5.786
x


bawa@t3head:/rdata2/bawa/validation/Wstar_Harinder
[harinder@harinder-OptiPlex-755 Exercises]$ root -l
root [0] TFile f("ntuple_data.root")
root [1] f.ls()
TFile** ntuple_data.root
TFile* ntuple_data.root
KEY: TNtuple ntuple;1 Example N-tuple
root [2] ntuple->Draw("x")
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [3] TCanvas c2
root [4] ntuple->Draw("y")
root [5] TCanvas c3
root [6] ntuple->Draw("x:y")
root [7]


c1_n2
File Edit View Options Tools Help
y
htemp
Entries 10000
Mean −0.03648
RMS 4.981
y


c1_n3
File Edit View Options Tools
x:y
x
y

Reading ntuple & Writing on Screen

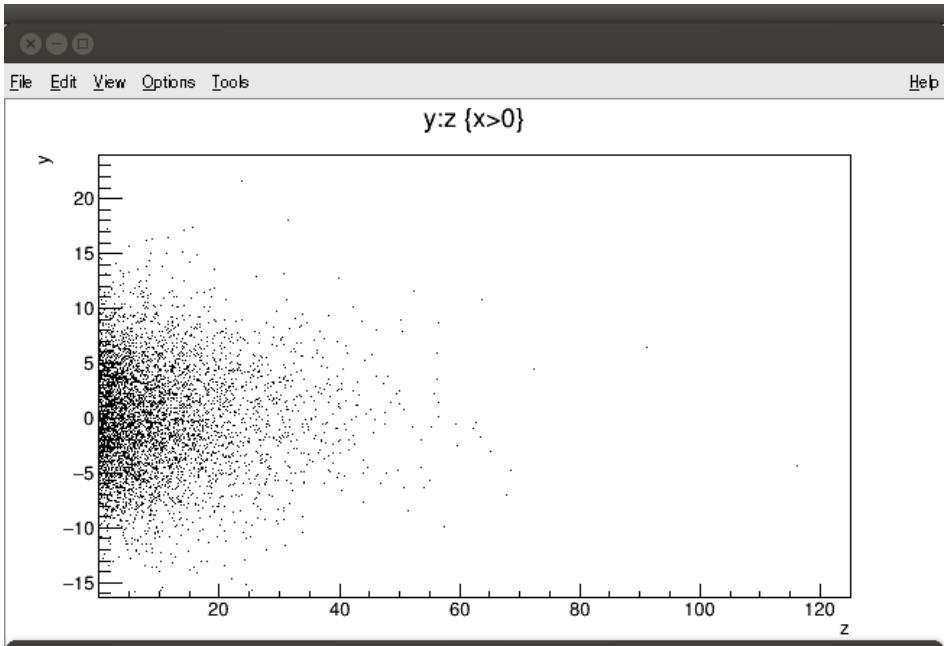
bawa@t3head:/rdata2/bawa/validation/Wstar_Harinder

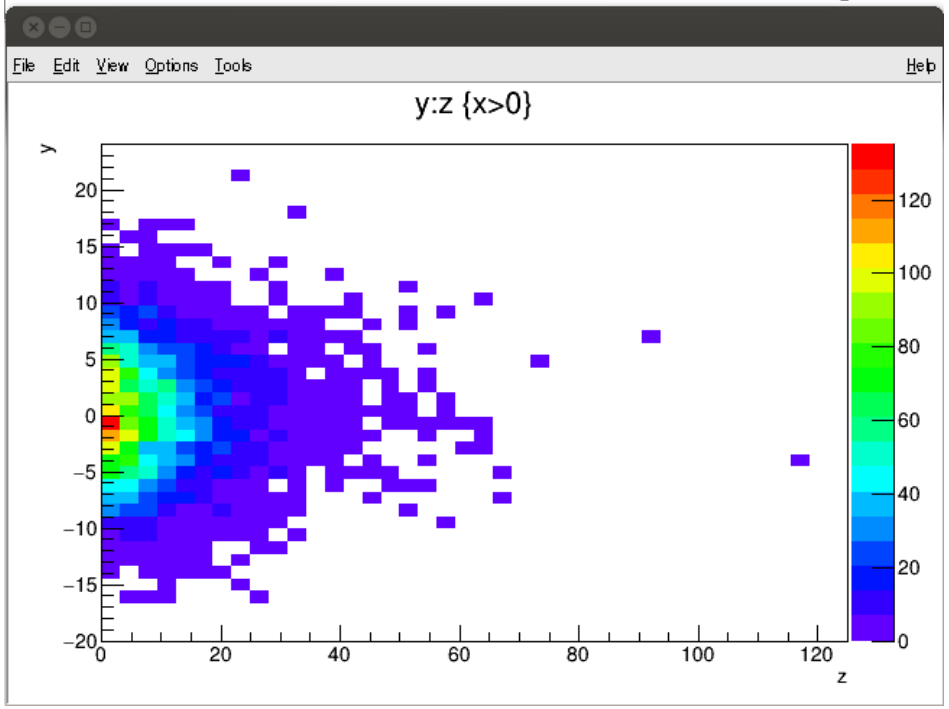
```
void exampleNtupleRead() {
   TFile f("ntuple_data.root");
   TNtuple *ntuple=0;
   f.GetObject("ntuple",ntuple);
   // loop on the ntuple entries

   for (int i = 0; i < ntuple->GetEntries(); ++i) {
      ntuple->GetEntry(i);
      float * raw_content = ntuple->GetArgs();
      float x = raw_content[0];
      float y = raw_content[1];
      float z = raw_content[2];
      float t = raw_content[3];
      if (i%1000==0) std::cout << x << "  " << y << "   " << z << "  " << t  << std::endl;
   }
// write in a file

   f.Close();

}
```

```
[harinder@harinder-OptiPlex-755 Exercises]$ root -l exampleNtupleRead.C
root [0]
Processing exampleNtupleRead.C...
9.99483  -2.17382   0.542439  -0.612445
5.26687  -0.567785   11.5874  -0.610093
0.521705  -1.77127   14.9077  0.442004
-9.28038  3.90928   0.152558  0.665459
-4.33669  -1.49484   2.11666  -0.778403
5.24322  -2.51007   10.8218  4.71695
5.43867  -4.75767   1.01158  -2.5583
-9.6626  -2.28142   0.0127684  8.1902
-4.96916  -7.83322   14.7144  -1.456
6.17546  -3.47435   6.39128  28.7686
root [1]
```

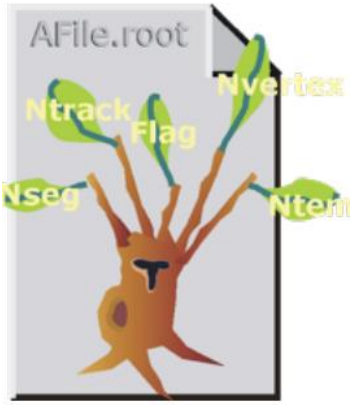
Ntuple:Draw()

```
void exampleNtupleDraw() {
        TFile f("ntuple_data.root");
        TNtuple *ntuple=0;
        f.GetObject("ntuple",ntuple);
        TCanvas *c=new TCanvas("c");
        ntuple->Draw("t");
        // to add a selection cut and a graphic option
        TCanvas *c1=new TCanvas("c1");
        ntuple->Draw("y:z","x>0","colz");
        TCanvas *c2=new TCanvas("c2");
        ntuple->Draw("y:z","x>0");
}
```





ROOT Trees?



- ROOT N-tuple can store only floating point variables
- For storing complex types, i.e. objects we can use the ROOT tree class, TTree –TNtuple is a special case of a TTree (a derived class)
- The ROOT Tree is
  - extremely efficient write once, read many.
  - Designed to store >109 (HEP events) with same data structure.
  - Trees allow fast direct and random access to any entry (sequential access is the best).
  - Optimized for network access (read-ahead)

Creating a ROOT Tree

```
#include "TRandom.h"
#include "TFile.h"
#include "TTree.h"

void ExampleTree(const char * filename= "tree.root") {

   TTree data("tree","Example TTree");
   double x, y, z, t;
   data.Branch("x",&x,"x/D");
   data.Branch("y",&y,"y/D");
   data.Branch("z",&z,"z/D");
   data.Branch("t",&t,"t/D");

// fill it with random data

   for (int i = 0; i<10000; ++i) {
      x = gRandom->Uniform(-10,10);
      y = gRandom->Gaus(0,5);
      z = gRandom->Exp(10);
      t = gRandom->Landau(0,2);

      data.Fill();
   }
// write in a file

   TFile f(filename,"RECREATE");
   data.Write();
   f.Close();

}
```

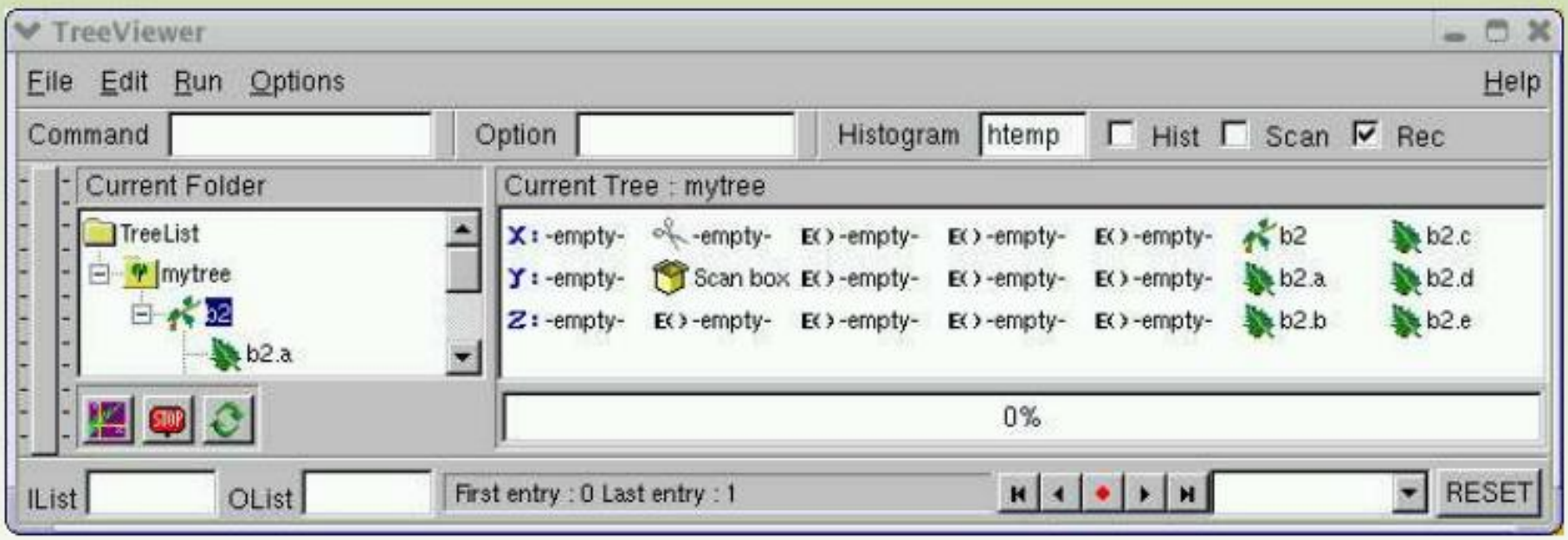
Scanning and Browsing a Tree

```
[harinder@harinder-OptiPlex-755 Exercises]$ root -l
root [0] TFile f("tree.root")
root [1] f.ls()
TFile**         tree.root
 TFile*         tree.root
  KEY: TTree    tree;1  Example TTree
root [2] tree->Scan()
************************************************************
*    Row   *         x *         y *         z *         t *
************************************************************
*        0 * 9.9948349 * -2.173821 * 0.5424387 * -0.612444 *
*        1 * -0.300527 * 4.1213184 * 2.9530392 * 3.3599092 *
*        2 * 4.7990596 * 0.0395610 * 4.1758331 * 0.3248392 *
*        3 * 6.0880602 * -4.925330 * 17.803898 * 2.3463682 *
*        4 * -2.153720 * -2.868095 * 34.954463 * 0.5302660 *
*        5 * -6.117022 * 3.8369828 * 5.4484498 * 22.924656 *
*        6 * 3.3112786 * 10.304510 * 12.156219 * -1.950201 *
*        7 * -8.741646 * -2.271012 * 3.2100651 * 5.3128856 *
*        8 * 4.2777012 * 9.2215651 * 3.5772235 * -2.077455 *
*        9 * -7.415144 * 11.302227 * 24.868109 * -1.812724 *
*       10 * 0.9474282 * 7.6234604 * 12.305172 * 21.557962 *
*       11 * -5.457643 * -6.407330 * 9.1531087 * 6.6468874 *
*       12 * -6.586581 * -5.453365 * 11.670097 * 2.8072593 *
*       13 * -6.351783 * -12.29070 * 18.275389 * 2.1263476 *
*       14 * 2.5567294 * -1.661886 * 3.5925466 * 2.3901384 *
*       15 * -8.197848 * -5.952137 * 8.2887077 * 2.3339561 *
*       16 * -1.615081 * 0.4030978 * 4.7062692 * 109.15314 *
*       17 * -2.591395 * 1.5039703 * 19.603690 * 8.6652320 *
*       18 * 6.5947936 * 1.2195639 * 10.592969 * -3.074309 *
*       19 * 2.2613264 * -3.939634 * 6.9536157 * 7.8710863 *
*       20 * -7.108989 * -6.082385 * 11.686474 * -1.895370 *
*       21 * -5.158606 * -0.598714 * 5.1116584 * 5.4938359 *
*       22 * -1.347475 * -4.175319 * 3.3290098 * 5.4394669 *
*       23 * 3.7155345 * 8.9050217 * 20.961120 * -2.862777 *
*       24 * 6.4014408 * -1.842549 * 17.116538 * -1.637794 *
Type <CR> to continue or q to quit ==> q
************************************************************
(Long64_t)25
root [3] tree->StartViewer()
```

Browsing the Tree

A graphical interface to play with a tree is started using:

```
mytree->StartViewer()
```



Tree on a Command Line

The contents of the tree can be drawn from the commandline:

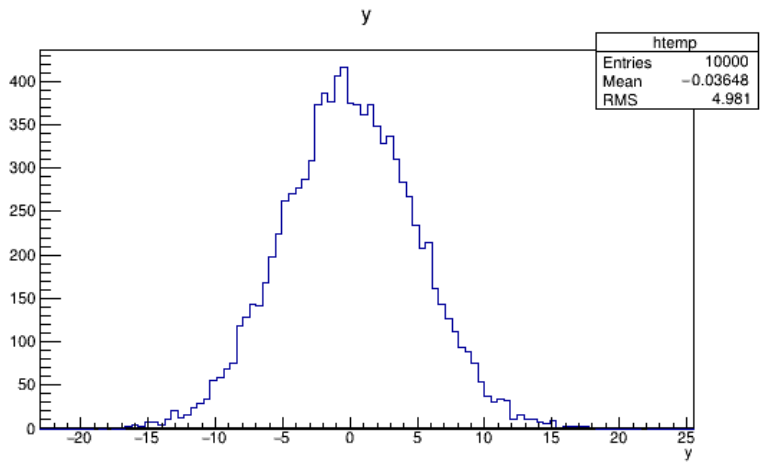
```
Draw(const char *exp, const char *cut, Option_t *option, Long64_t nent, Long64_t first)
```

exp: expression describing what to draw, *e.g.* "y:x", or "sqrt(x/y*z*z)". For 2-D (or 3-D) plots, expressions are separated by a ":". Convention: $z : y : x$. Statement "x»histoname" will save to predefined histogram.

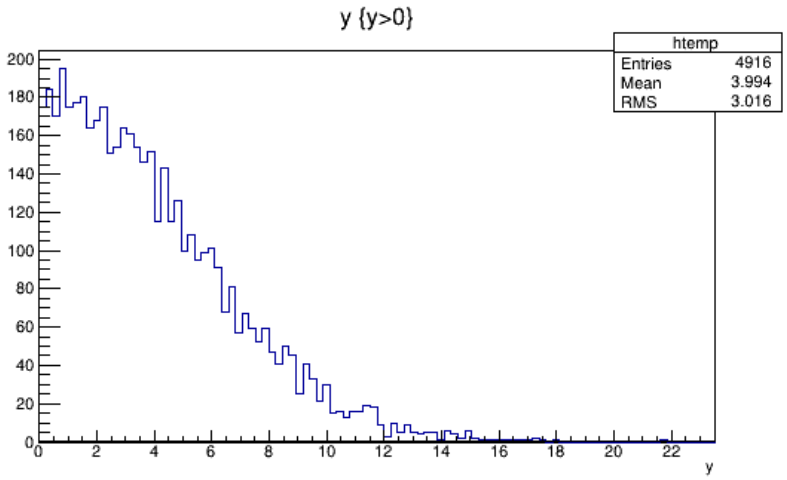
cut: expression describing some conditions, *e.g.* "z>0"

option: drawing option (see histograms)

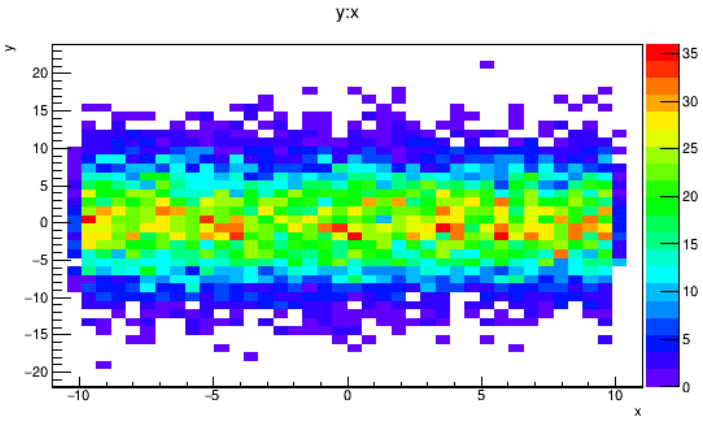
tv__tree->Draw("y","","", 10000, 0);



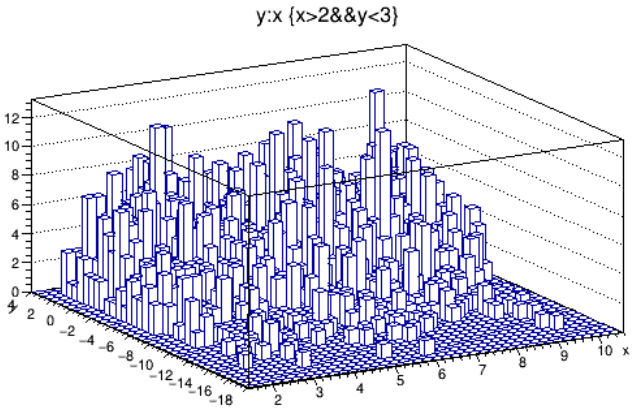
tv__tree->Draw("y","y>0","", 10000, 0);



tv__tree->Draw("y:x","","colz", 10000, 0);



tv__tree->Draw("y:x","x>2&&y<3","LEGO", 10000, 0);



Exercise

- Create a ntuple and fill with px,py,pz and events as random values

- Create a ROOT tree and fill with px,py,pz and events as random values

Hint: Use Rannor function to get px&py as gaussian random value
gRandom->Rannor(px,py)

Also use pz=sqrt(px*px+py*px)

- Use tree/ntuple StartViewer and plot px Vs py with pz>10
- Plot py with px>5 or pz<10