C++ AND ROOT

J. STROLOGAS

INTRODUCTION TO ROOT

- ROOT is the main data analysis package for particle physics and beyond
- It is especially able to process massive multidimensional data (big data)
- It can be used in compiled C++ code or interpreted macros (.C)
  - Also in Python, but we will not cover that
- ROOT includes a number of classes that inherit from other classes etc.
  - ROOT classes start with “T”, e.g., TH1D or TCanvas

WHAT ROOT OBJECTS WE WILL COVER

- Histograms (TH1D and TH2D)
  - A way to group events in one or two dimensions
- Functions (TF1)
  - We will use them to fit histograms
- Trees (TTree)
  - A way to analyze multidimensional data
- Random numbers (TRandom3)
  - Production of random distributions
- Visualization (TCanvas)
  - A way to present plots

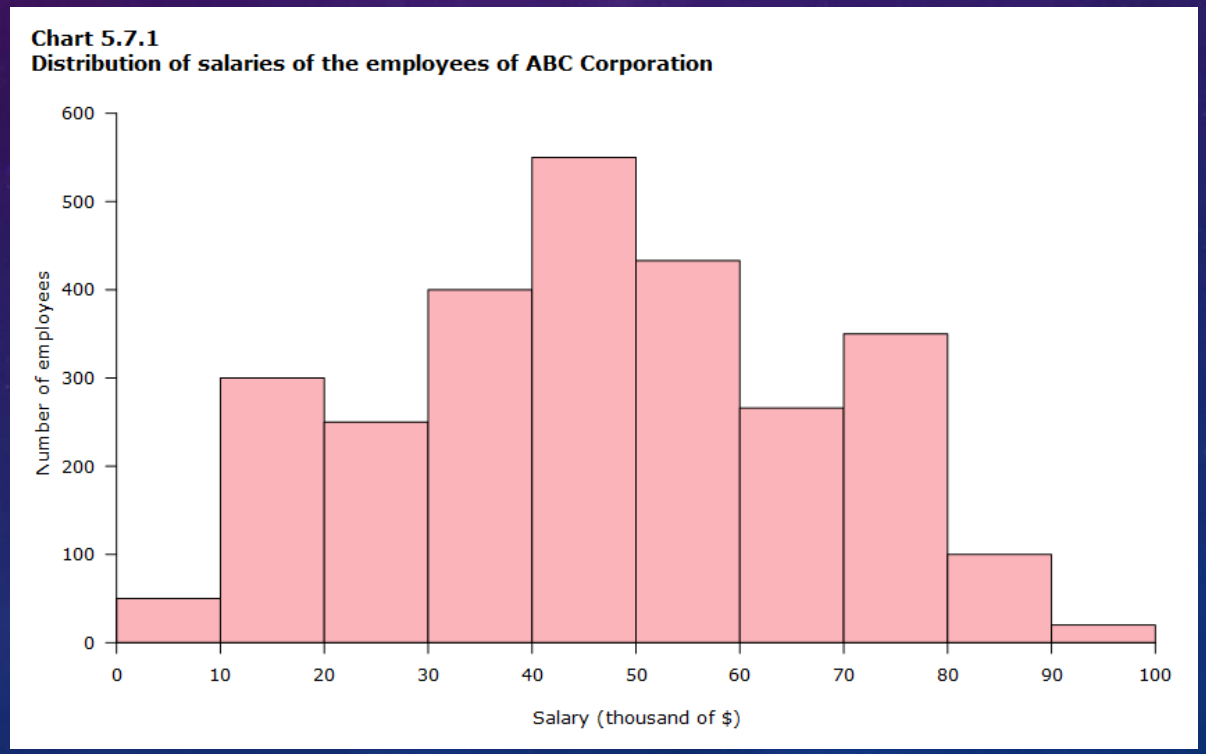
HISTOGRAM

- A 1D histogram groups events in one dimension
  - If an event gives you value within a bin, you increment the number of entries in that bin

HISTOGRAM

- A 1D histogram groups events in one dimension
  - If an event gives you value within a bin, you increment the number of entries in that bin

Y-axis: Number of events (or people here)



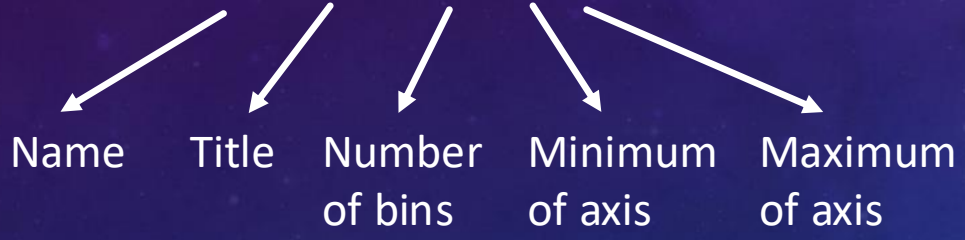
10 bins of size 10 each

X-axis: The quantity

Government of Canada | Gouvernement du Canada

HISTOGRAM

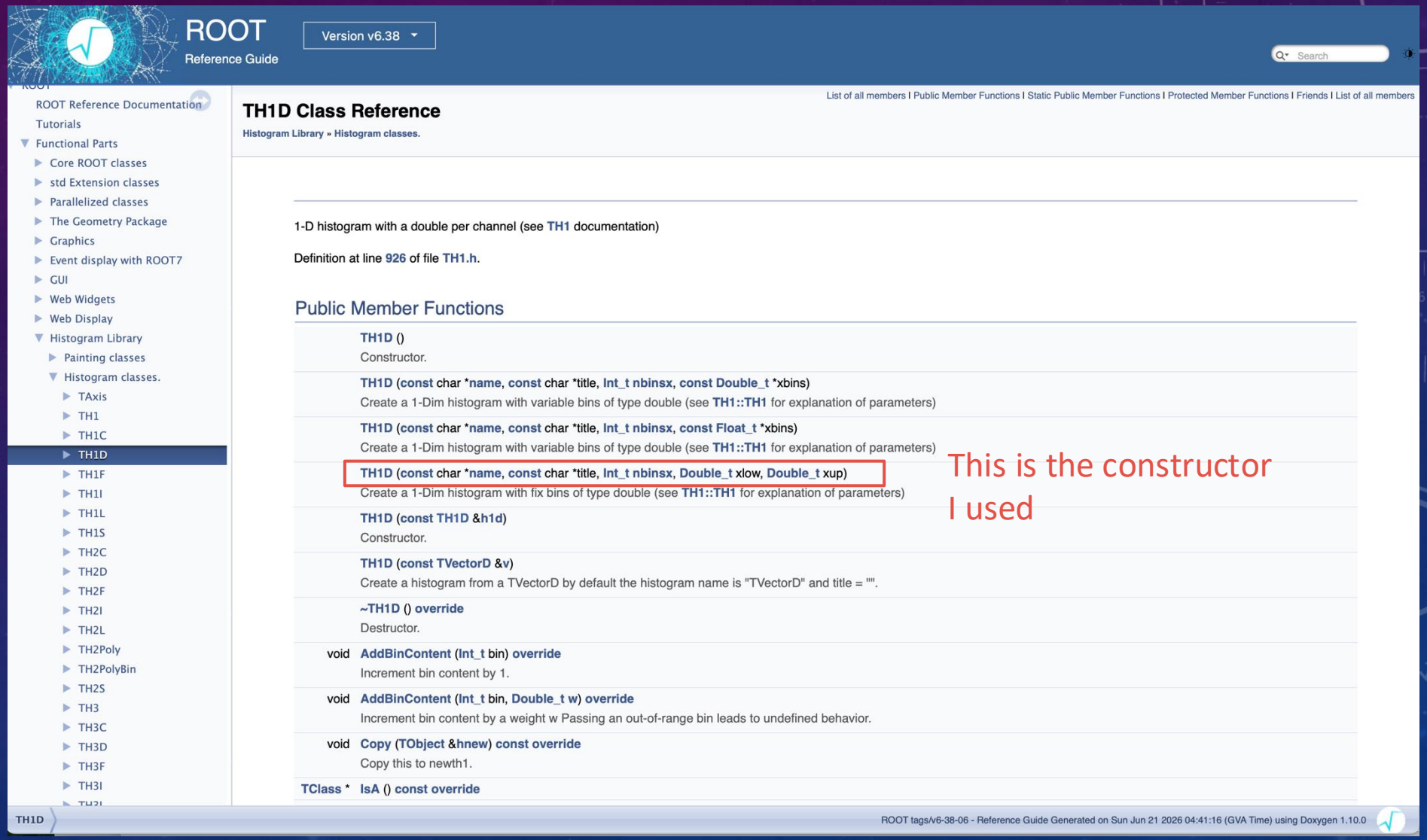
- A 1D histogram groups events in one dimension
  - If an event gives you value within a bin, you increment the number of entries in that bin
- The ROOT class is TH1D
  - D means “double”, one can use TH1F or TH1I for float and integer entries
- The usual constructor for this class is :
  TH1D(“h”,“h”,100,0,50)



- The most important methods of TH1D are Fill() and Draw()
  - To fill histogram with one entry and to draw the histogram

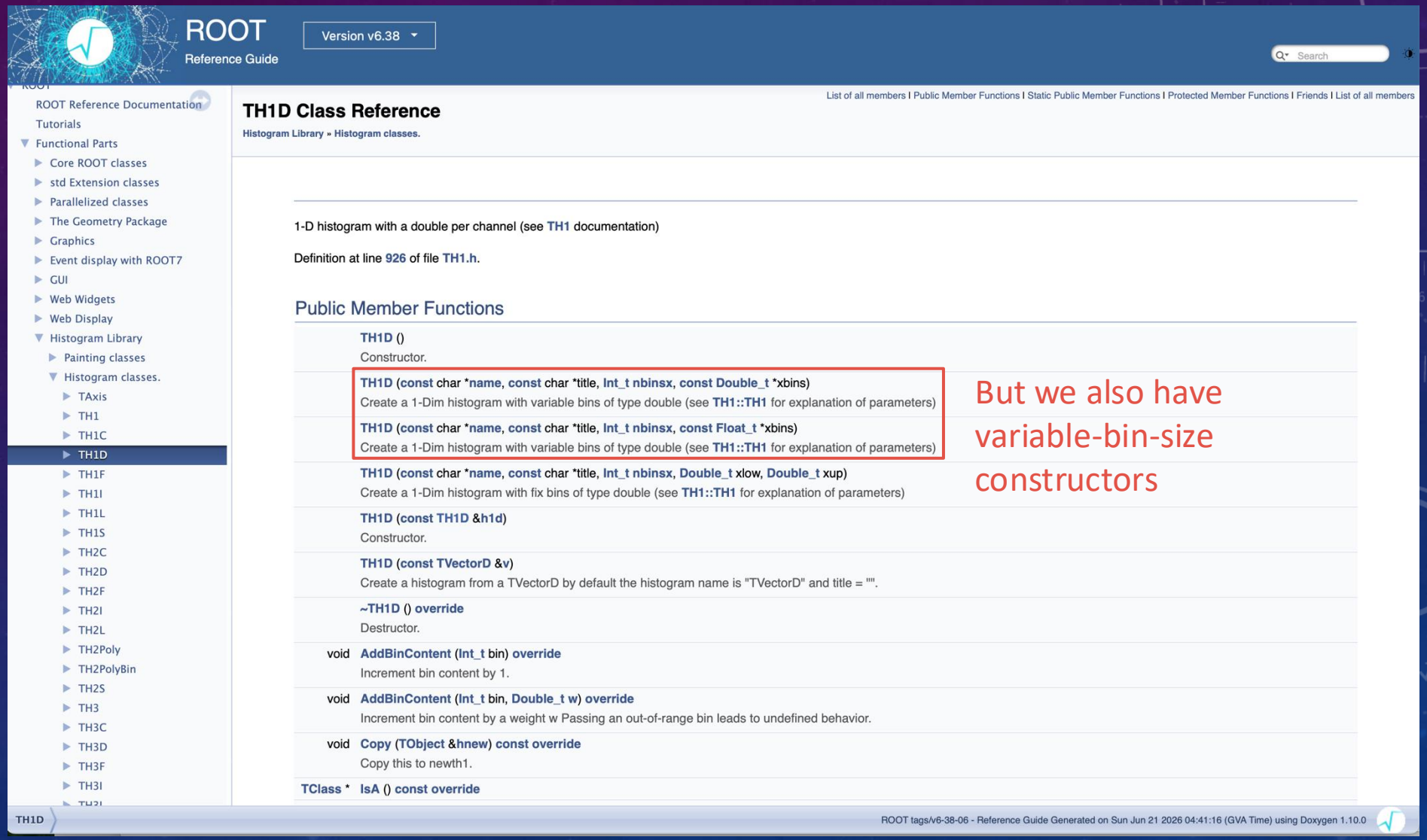
HOW DO YOU KNOW THE ORDER AND KIND OF METHOD ARGUMENTS?

- Check the manual



HOW DO YOU KNOW THE ORDER AND KIND OF METHOD ARGUMENTS?

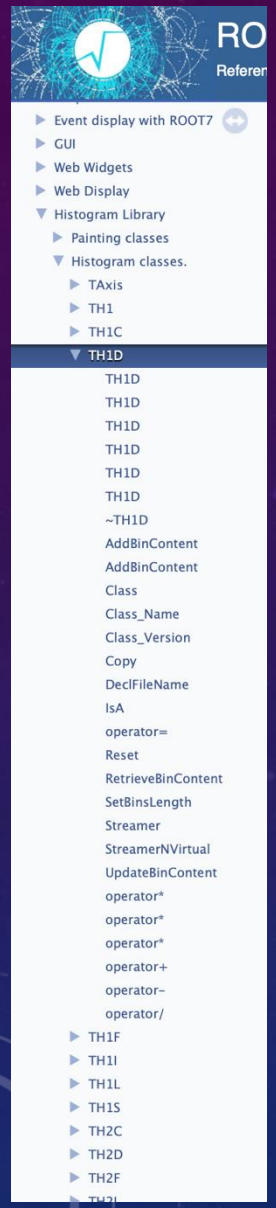
- Check the manual



SIMILARLY YOU CAN FIND ALL METHODS OF ANY OBJECT AND THEIR FUNCTIONALITY

- Commonly used methods of the object TH1D:
  - Fill
  - Draw
  - Print
  - GetBinContent
  - SetBinContent
  - Fit

YOU CAN FIND ALL METHODS YOU NEED AT IN THE MANUAL

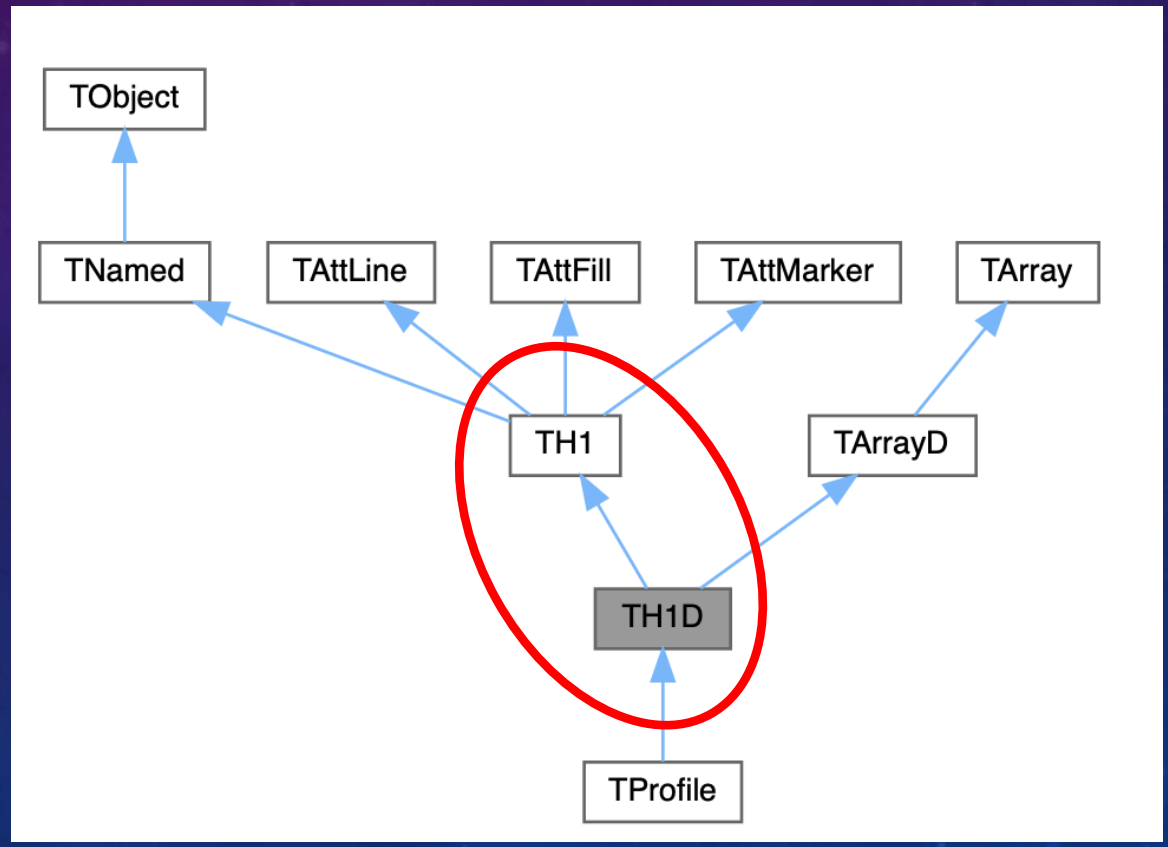


- But where is Draw or Fill ???

YOU CAN FIND ALL METHODS YOU NEED AT IN THE MANUAL

TH1
TH1
TH1
TH1
TH1
~TH1
Add
Add
Add
AddBinContent
AddBinContent
AddDirectory
AddDirectoryStatus
AndersonDarlingTest
AndersonDarlingTest
AutoP2FindLimits
AutoP2GetBins
AutoP2GetPower2
AxisChoice
Browse
BufferEmpty
BufferFill
Build
CanExtendAllAxes
CheckAxisLimits
CheckBinLabels
CheckBinLimits
CheckConsistency
CheckConsistentSubAxes
CheckEqualAxes
Chi2Test
Chi2TestX
Chisquare
Class
Class_Name
Class_Version
ClearUnderflowAndOverflow
Clone
ComputeIntegral
Copy
DeclFileName
DirectoryAutoAdd
DistancetoPrimitive
Divide
Divide
Divide
DoFillN
DoIntegral
Draw

- It is inherited from TH1 !!!



```
 ------------------------------------------------------------------
  | Welcome to ROOT 6.34.04                          https://root.cern |
  | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Feb 11 2025, 13:05:44                   |
  | From tags/6-34-04@6-34-04                                        |
  | With Apple clang version 16.0.0 (clang-1600.0.26.6)              |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
 ------------------------------------------------------------------

root [0] TH1D *h = new TH1D("h","h",10,0,50)
(TH1D *) 0x14e90b090
root [1] h->Fill(23)
(int) 5
root [2] h->Print("all")
TH1.Print Name  = h, Entries= 1, Total sum= 1
 fSumw[0]=0, x=-2.5
 fSumw[1]=0, x=2.5
 fSumw[2]=0, x=7.5
 fSumw[3]=0, x=12.5
 fSumw[4]=0, x=17.5
 fSumw[5]=1, x=22.5
 fSumw[6]=0, x=27.5
 fSumw[7]=0, x=32.5
 fSumw[8]=0, x=37.5
 fSumw[9]=0, x=42.5
 fSumw[10]=0, x=47.5
 fSumw[11]=0, x=52.5
root [3]
root [3]
root [3] h->Draw()
```

I have 10 histogram bins with index 1-10. Index 0 shows the underflows and index 11 the overflows

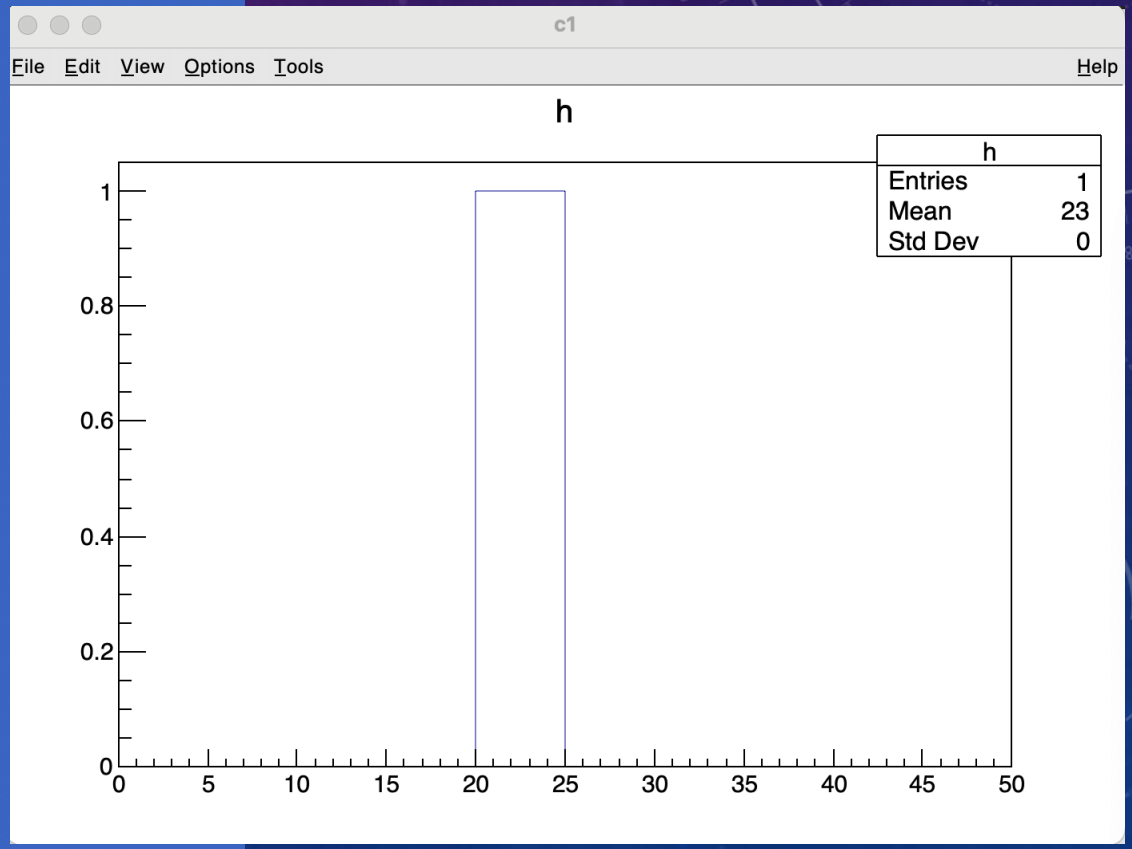
HOW WE WILL WRITE ROOT CODE

- One can start a root session by typing “root” and then one can type commands sequentially
- Here I create a pointer to TH1D with the constructor and the “new” command
- 10 bins from 0 to 50
- I then fill 1 event with value 23
- And then I Print the content of the histogram and then Draw it

HOW WE WILL WRITE ROOT CODE

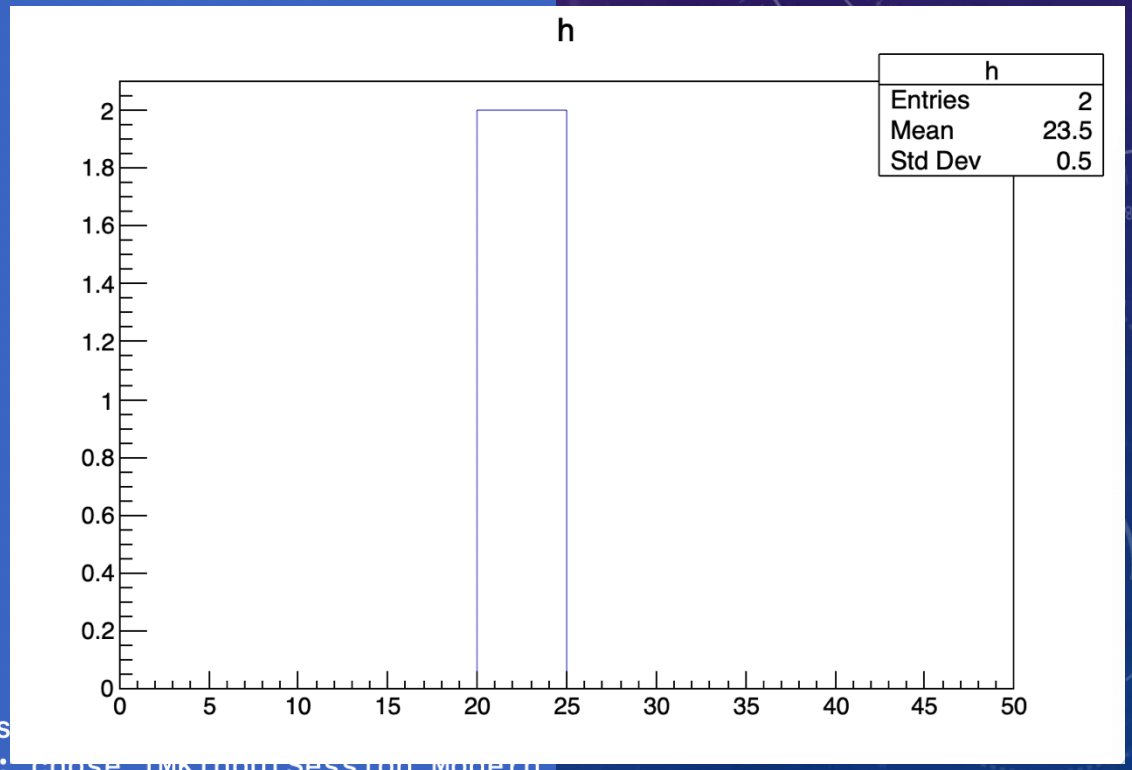
```
 ------------------------------------------------------------------
 | Welcome to ROOT 6.34.04                          https://root.cern |
 | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
 | Built for macosxarm64 on Feb 11 2025, 13:05:44                   |
 | From tags/6-34-04@6-34-04                                        |
 | With Apple clang version 16.0.0 (clang-1600.0.26.6)              |
 | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
 ------------------------------------------------------------------

root [0] TH1D *h = new TH1D("h","h",10,0,50)
(TH1D *) 0x14e90b090
root [1] h->Fill(23)
(int) 5
root [2] h->Print("all")
TH1.Print Name  = h, Entries= 1, Total sum= 1
 fSumw[0]=0, x=-2.5
 fSumw[1]=0, x=2.5
 fSumw[2]=0, x=7.5
 fSumw[3]=0, x=12.5
 fSumw[4]=0, x=17.5
 fSumw[5]=1, x=22.5
 fSumw[6]=0, x=27.5
 fSumw[7]=0, x=32.5
 fSumw[8]=0, x=37.5
 fSumw[9]=0, x=42.5
 fSumw[10]=0, x=47.5
 fSumw[11]=0, x=52.5
root [3]
root [3]
root [3] h->Draw()
```

I have 10 histogram bins with index 1-10. Index 0 shows the underflows and index 11 the overflows



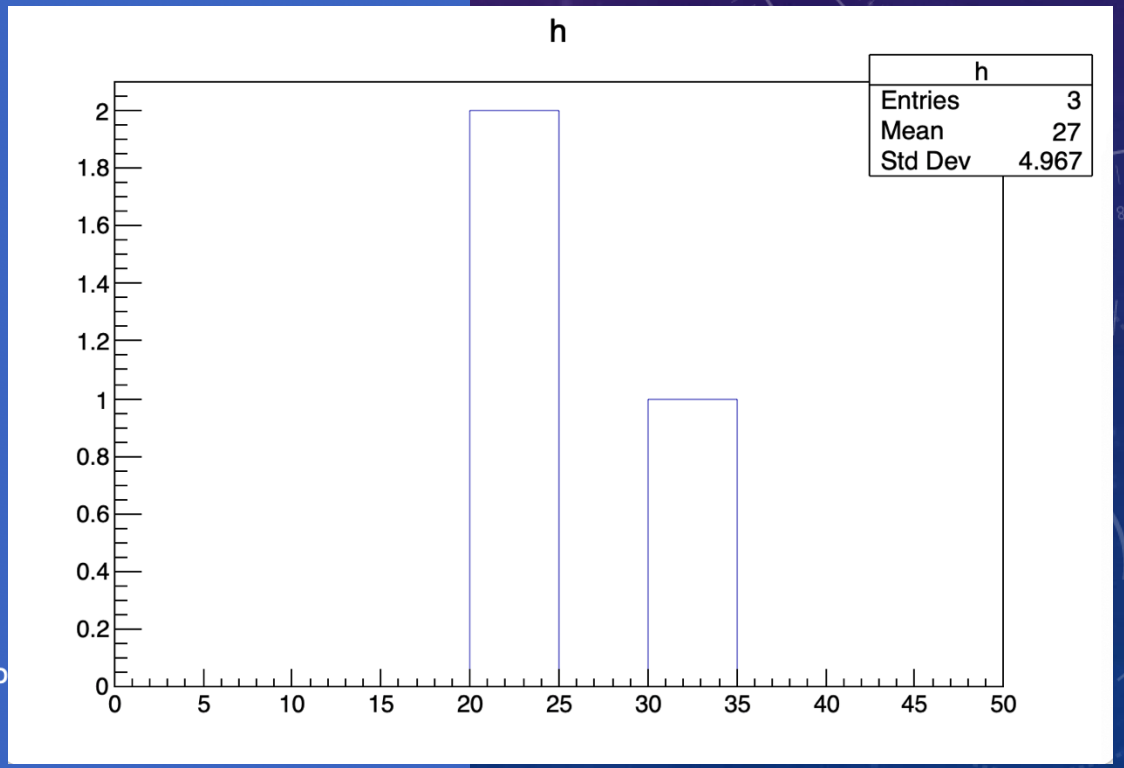
HOW WE WILL WRITE ROOT CODE

```
 ------------------------------------------------------------------
  | Welcome to ROOT 6.34.04                          https://root.cern |
  | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Feb 11 2025, 13:05:44                   |
  | From tags/6-34-04@6-34-04                                        |
  | With Apple clang version 16.0.0 (clang-1600.0.26.6)              |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0] TH1D *h = new TH1D("h","h",10,0,50)
(TH1D *) 0x14e90b090
root [1] h->Fill(23)
(int) 5
root [2] h->Print("all")
TH1.Print Name  = h, Entries= 1, Total sum= 1
 fSumw[0]=0, x=-2.5
 fSumw[1]=0, x=2.5
 fSumw[2]=0, x=7.5
 fSumw[3]=0, x=12.5
 fSumw[4]=0, x=17.5
 fSumw[5]=1, x=22.5
 fSumw[6]=0, x=27.5
 fSumw[7]=0, x=32.5
 fSumw[8]=0, x=37.5
 fSumw[9]=0, x=42.5
 fSumw[10]=0, x=47.5
 fSumw[11]=0, x=52.5
root [3]
root [3]
root [3] h->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [4] 2026-07-02 01:22:41.481 root.exe[39664:1109390] +[IMKClient subclas
2026-07-02 01:22:41.481 root.exe[39664:1109390] +[IMKInputSession subclass]: chose IMKInputSession_Modern

root [4]
root [4] h->Fill(24)
(int) 5
root [5] h->Draw()
```



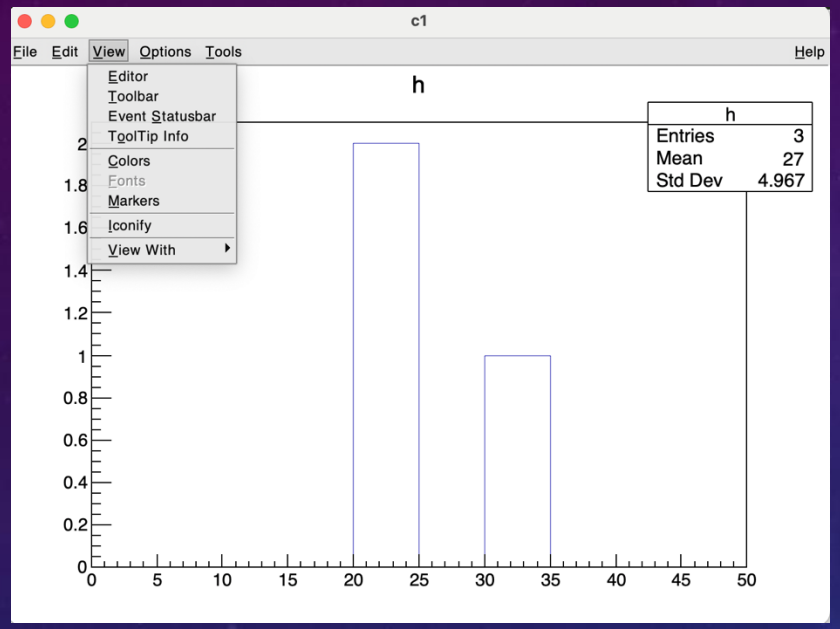
HOW WE WILL WRITE ROOT CODE

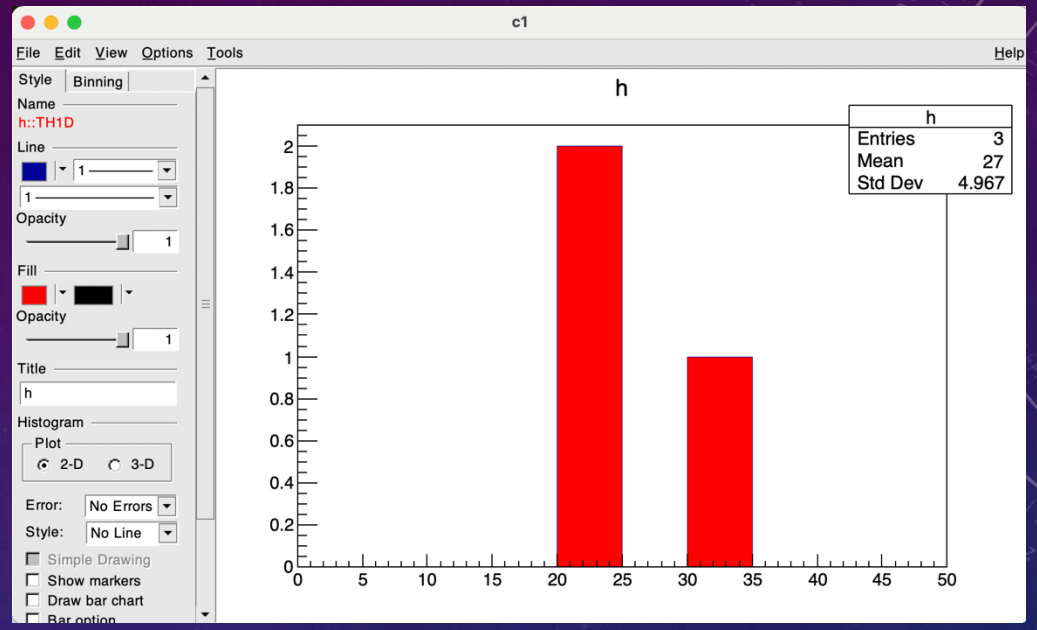
```
   ------------------------------------------------------------------
  | Welcome to ROOT 6.34.04                          https://root.cern |
  | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Feb 11 2025, 13:05:44                   |
  | From tags/6-34-04@6-34-04                                        |
  | With Apple clang version 16.0.0 (clang-1600.0.26.6)              |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0] TH1D *h = new TH1D("h","h",10,0,50)
(TH1D *) 0x14e90b090
root [1] h->Fill(23)
(int) 5
root [2] h->Print("all")
TH1.Print Name  = h, Entries= 1, Total sum= 1
 fSumw[0]=0, x=-2.5
 fSumw[1]=0, x=2.5
 fSumw[2]=0, x=7.5
 fSumw[3]=0, x=12.5
 fSumw[4]=0, x=17.5
 fSumw[5]=1, x=22.5
 fSumw[6]=0, x=27.5
 fSumw[7]=0, x=32.5
 fSumw[8]=0, x=37.5
 fSumw[9]=0, x=42.5
 fSumw[10]=0, x=47.5
 fSumw[11]=0, x=52.5
root [3]
root [3]
root [3] h->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [4] 2026-07-02 01:22:41.481 root.exe[39664:1109390] +[IMKClient subclass]:
2026-07-02 01:22:41.481 root.exe[39664:1109390] +[IMKInputSession subclass]: cho

root [4]
root [4] h->Fill(24)
(int) 5
root [5] h->Draw()
root [6] h->Fill(34)
(int) 7
root [7] h->Draw()
```



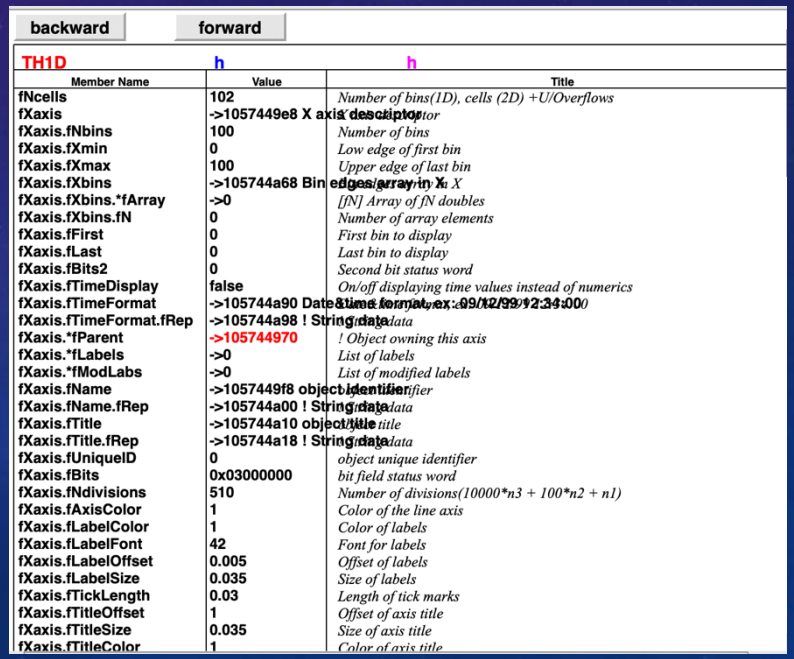
USEFUL TRICKS

- Open the editor to edit any object





- With right click while pointing to any object you can select “inspect” and see all member variables of the object and its value



ROOT MACROS

- Usually, we don’t type the commands at the root prompt, but we save them in macros that are interpreted
  - If we want, we can also compile the root code, but we won’t do that today

- With macros, we will violate some rules
  - We will not have the main() function
    - Instead, the main function will have the same name as our macro. E.g., if our macro is named histoGauss.C, the main function will be void histoGauss()
  - We don’t need all include commands

  - We run the macro by issuing the command: root histoGauss.C

OUR FIRST ROOT MACRO EXAMPLE

- We will write a macro that
  1. Generates 1M random numbers between 0 and 100 that follow a Gaussian distribution
  2. Fills a histogram with these numbers
  3. Creates a canvas
  4. Draws the histogram

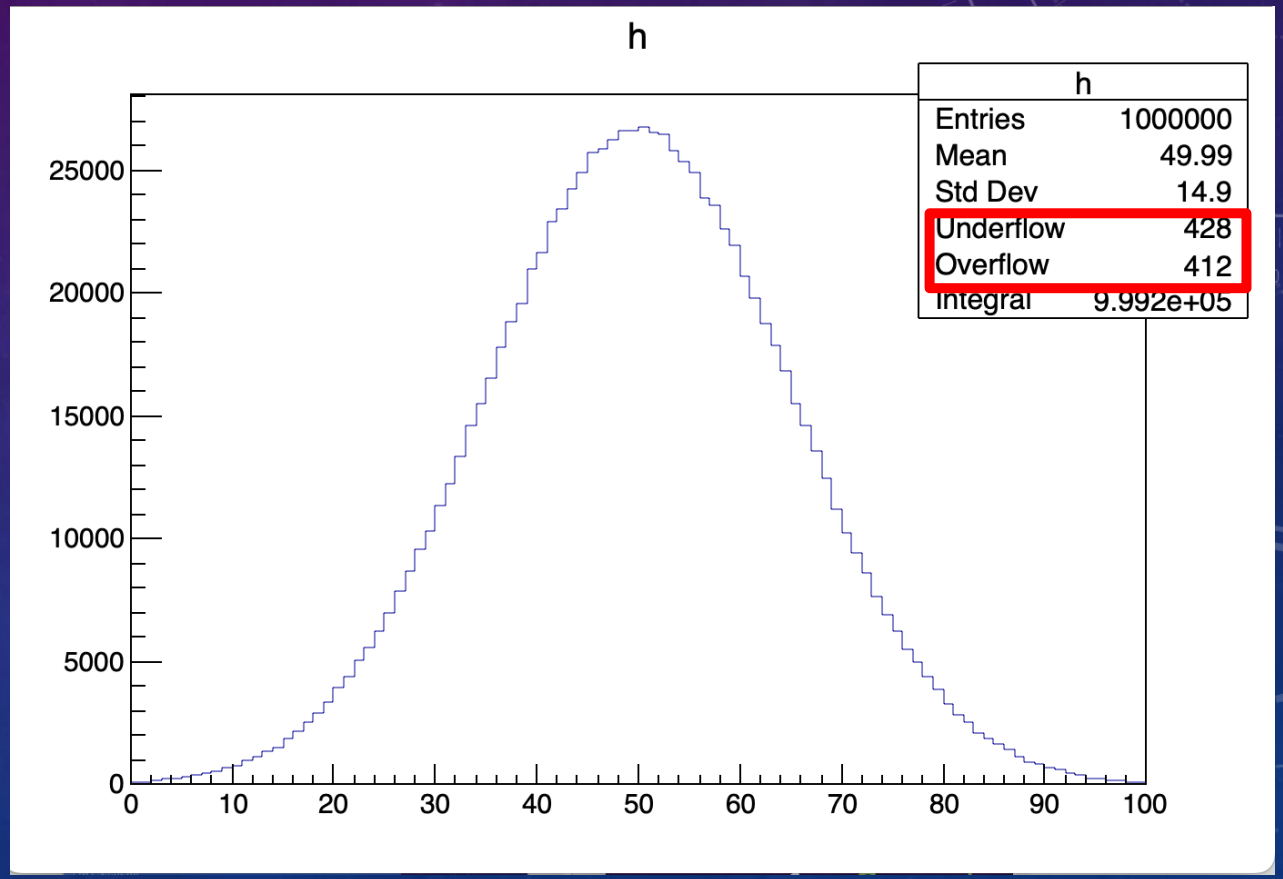
OUR FIRST ROOT MACRO EXAMPLE

- We will write a macro that
  1. Generates 1M random numbers between 0 and 100 that follow a Gaussian distribution
  2. Fills a histogram with these numbers
  3. Creates a canvas
  4. Draws the histogram

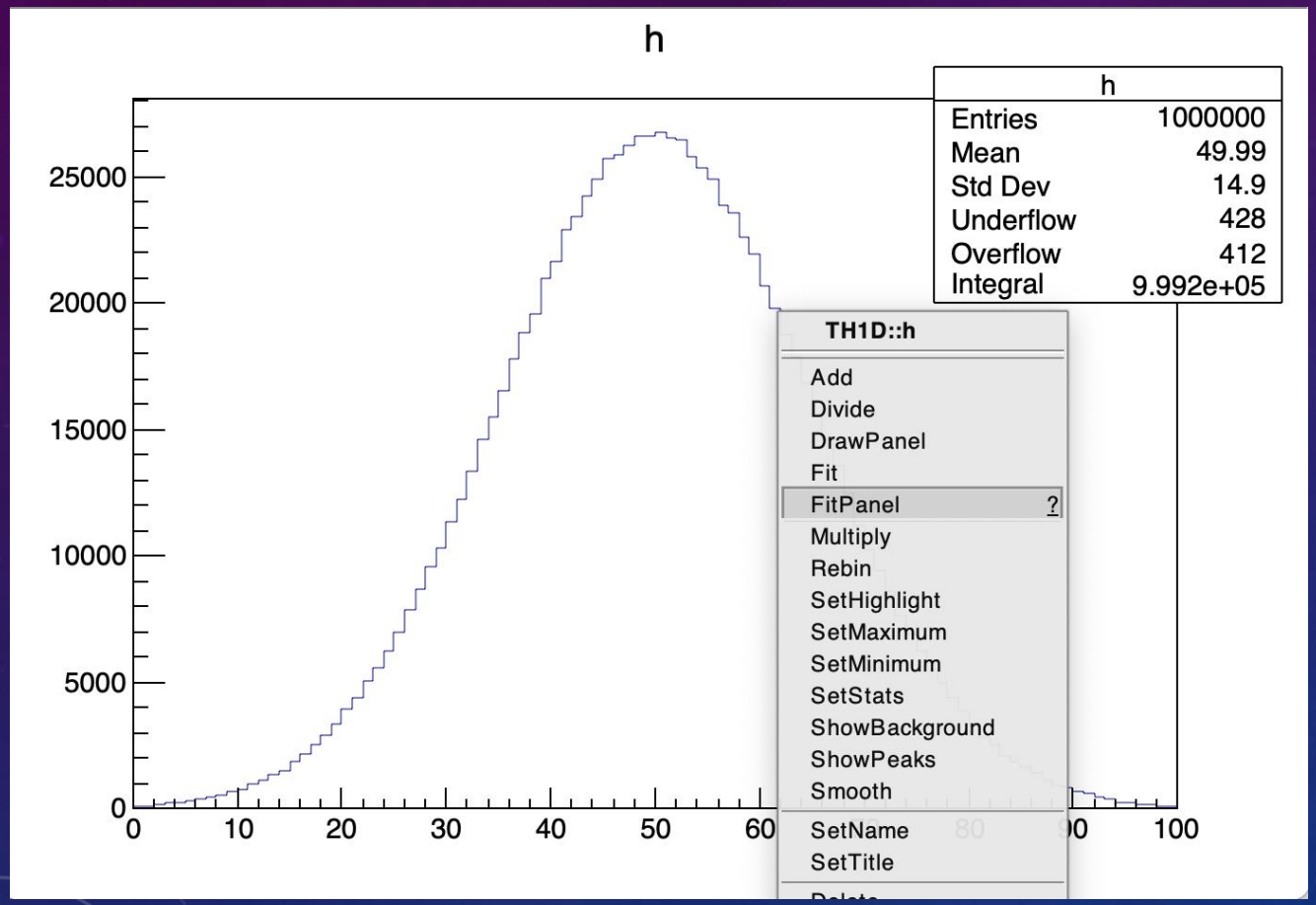
```
void histoGauss(){

  TH1D *h = new TH1D("h","h",100,0,100);

  TRandom3 *r = new TRandom3(343);

  for (int i=0; i<1000000; i++){

    h->Fill(r->Gaus(50,15));

  }

  TCanvas *c = new TCanvas();
  c->cd();

  h->Draw();


}
```

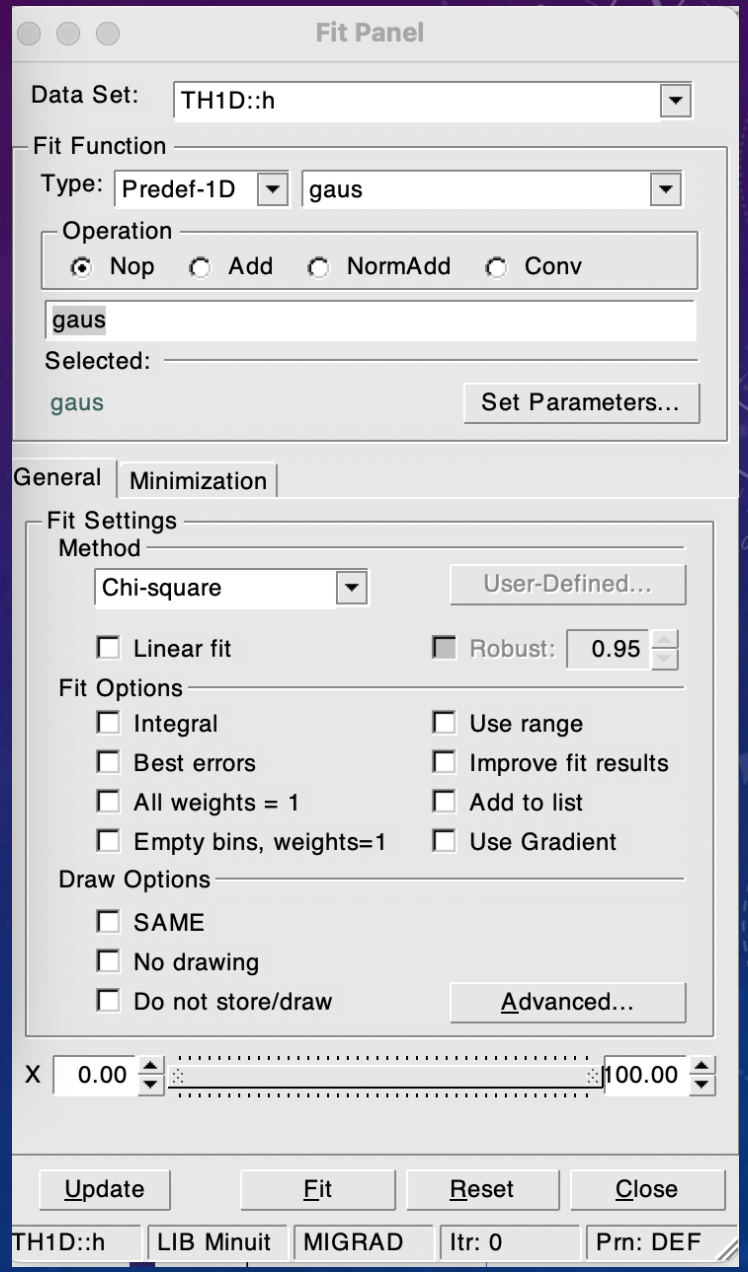
OUR FIRST ROOT MACRO EXAMPLE

```
void histoGauss(){

  TH1D *h = new TH1D("h","h",100,0,100);

  TRandom3 *r = new TRandom3(343);

  for (int i=0; i<1000000; i++){

    h->Fill(r->Gaus(50,15));

  }

  TCanvas *c = new TCanvas();
  c->cd();

  h->Draw();


}
```

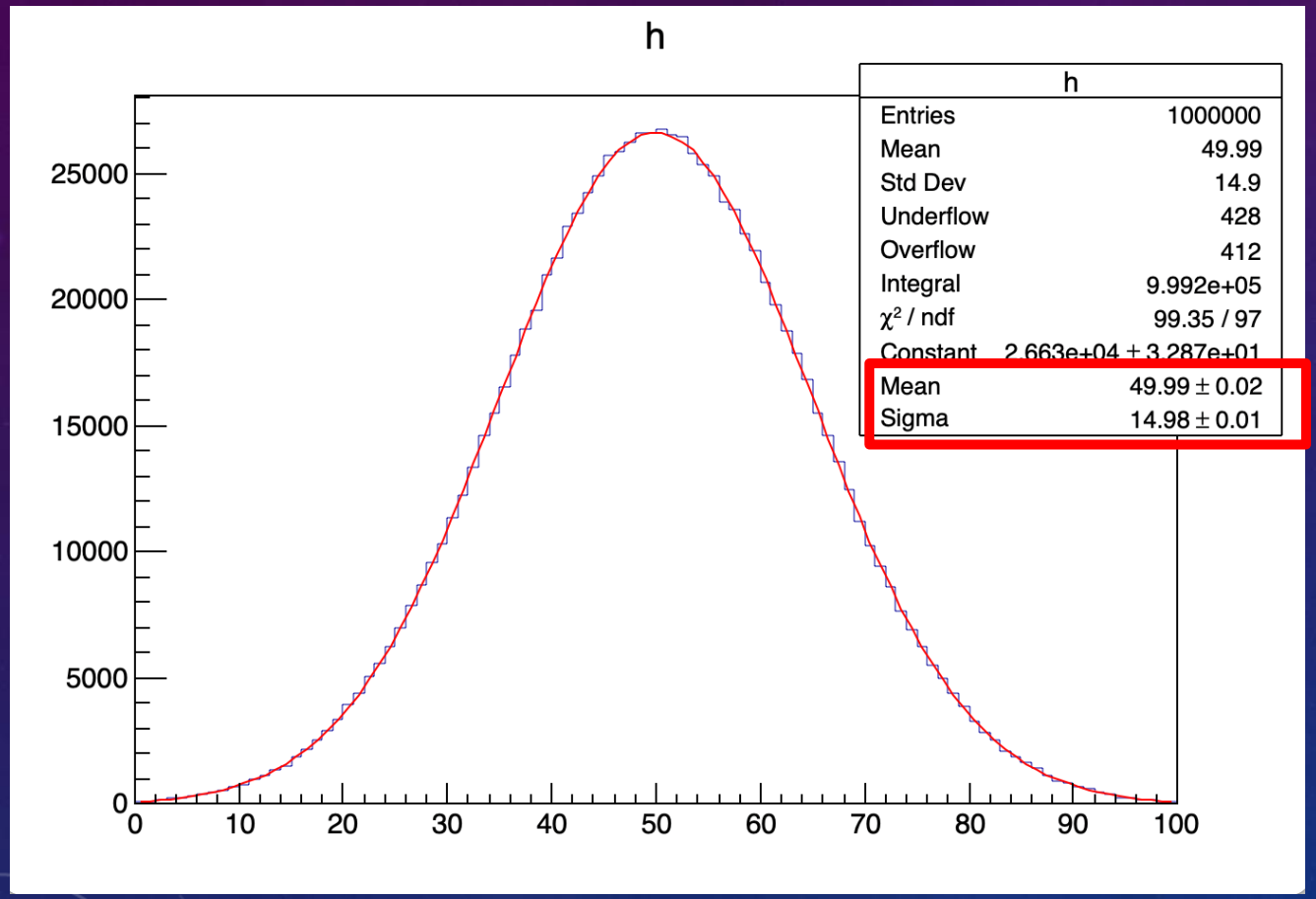


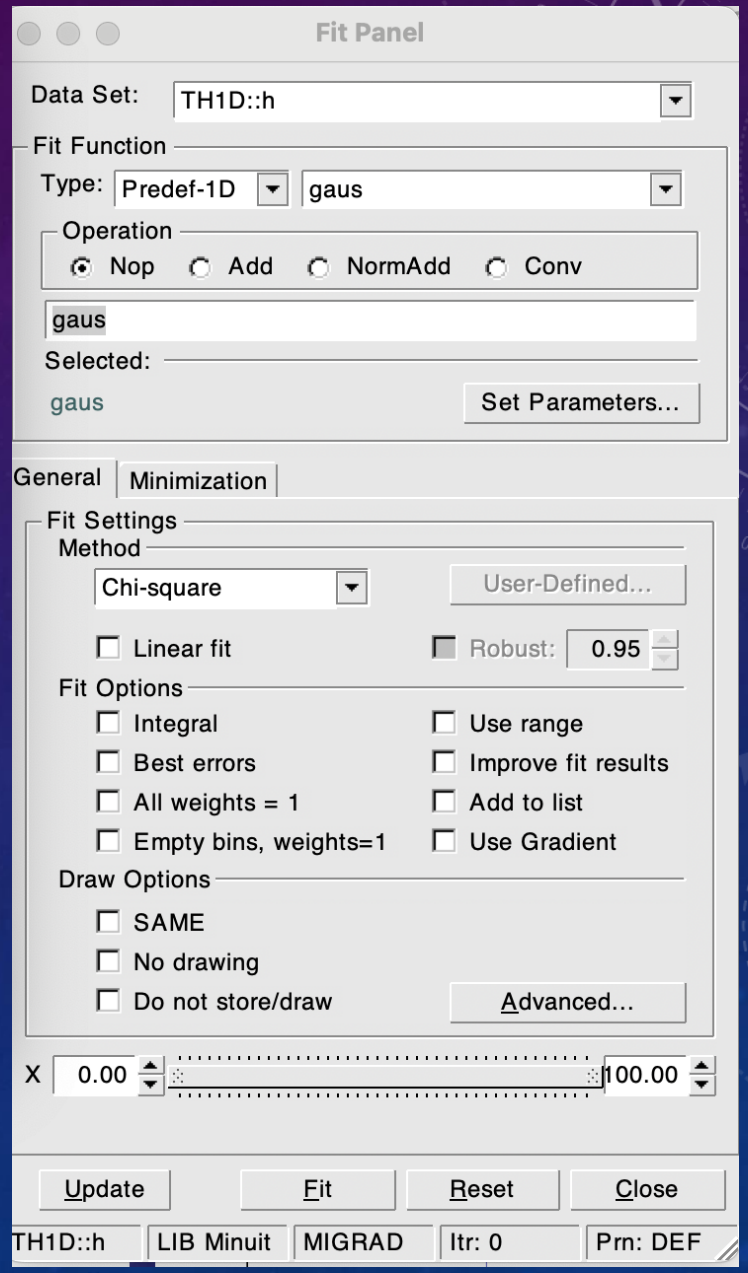
FIRST FIT USING FIT PANEL





FIRST FIT USING FIT PANEL





We will revisit the fits after we discuss the functions

ROOT FUNCTIONS

- Function of one parameter are described by class TF1
- There are different constructors
  - TF1(“f”,“gaus”,0,100)
    - This is a gaussian from 0 to 100 (the 3 parameters are hidden)
  - TF1("f","[0]*exp(-0.5*((x-[1])/[2])**2)",0,100);
    - This is also a gaussian from 0 to 100 explicitly written (parameters [0] (factor), [1] (mean), [2] (sigma)
  - TF1 *f = new TF1("f",myFunc,0,100,3);
    - A C++ function myFunc is used, that has 3 parameters

TF1 *F = NEW TF1("F",MYFUNC,0,100,3);

- If we want to fit the histogram with a gaussian defined by my own C++ function, the function has the form:

```
double myFunc(Double_t *x, Double_t *par){
 double xx=x[0];
 double result = par[0]*exp(-0.5*pow(((xx-par[1])/par[2]),2));
 return result;
}
```

- Also a gaussian, only it is now defined by us

CONTINUATION OF CODE WITH THE FIT

```
double myFunc(Double_t *x, Double_t *par);

void histoGauss(){

  TH1D *h = new TH1D("h","h",100,0,100);

  TRandom3 *r = new TRandom3(343);

  for (int i=0; i<1000000; i++){

    h->Fill(r->Gaus(50,15));

  }

  TCanvas *c = new TCanvas();
  c->cd();

  h->Draw();

  //TF1 *f = new TF1("f","gaus",0,100);
  //TF1 *f = new TF1("f","[0]*exp(-0.5*((x-[1])/[2])**2)",0,100);
  TF1 *f = new TF1("f",myFunc,0,100,3);
  f->SetParLimits(1,0,100);
  f->SetParLimits(2,0,100);

  h->Fit(f);

}

double myFunc(Double_t *x, Double_t *par){

  double xx=x[0];
  double result = par[0]*exp(-0.5*pow(((xx-par[1])/par[2]),2));
  return result;

}
```

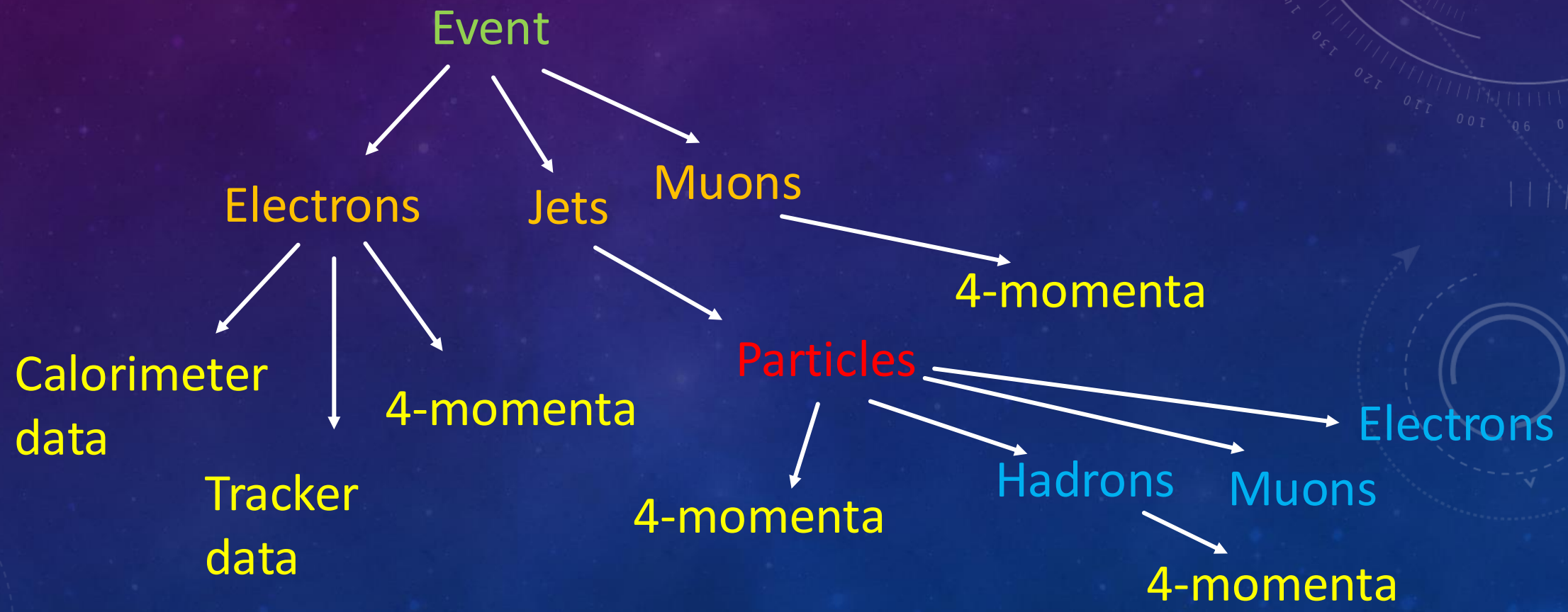
- We will uncomment one of the three TF1 options to perform the fit

TTREE OBJECT

- In Experimental Particle Physics we usually study collisions of particles that produce new particles
- One event is such a collision
  - For example, two protons collide and we detect an electron and a positron. These two particles may have been the result of the decay of a Z boson
- This event is characterized by objects that we may be able to detect (electron and positron) or may not be able to detect directly (outgoing quarks and gluons or the Z boson → we have those in Monte Carlo simulations).
  - These objects are characterized by other objects (e.g., detector tracks, calorimetry energies, reconstructed momenta etc.)
- So, each event can be described by many variables that have a tree structure

TREE STRUCTURE OF AN EVENT

- Each event is described by branches that may have other branches etc and the end up with leaves

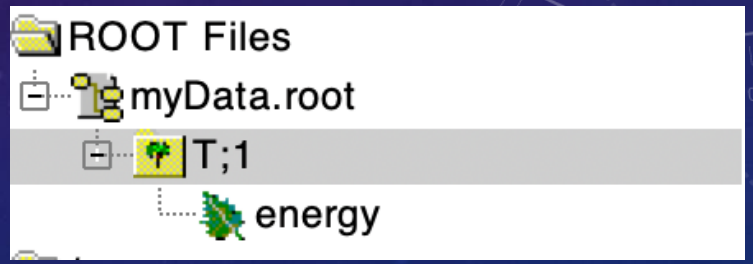


THE TTREE ROOT CLASS

- There are many ways to create a TTree
- We will try some of them today
- We will fill the values of the variables with random numbers following specific distributions

CASE 1

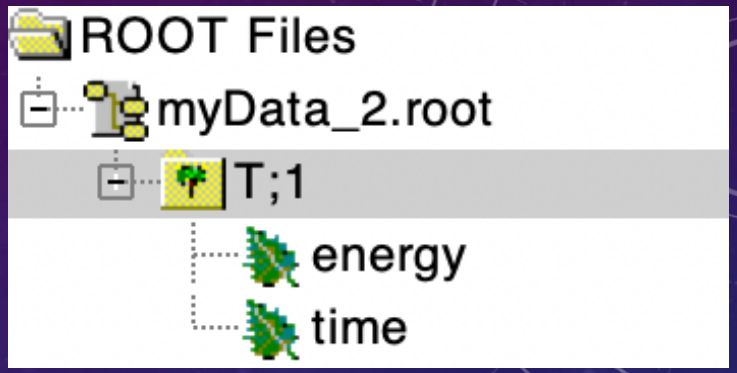
```
void makeTree(){

  TFile *f = new TFile("myData.root", "recreate");
  TTree *tree = new TTree("T", "My First Tree");
  Double_t energy;
  tree->Branch("energy", &energy, "energy/D");
  for (int i=0; i<100; i++) {
    energy = gRandom->Gaus(50, 5);
    tree->Fill();
  }
  tree->Write();
  f->Close();
}
```

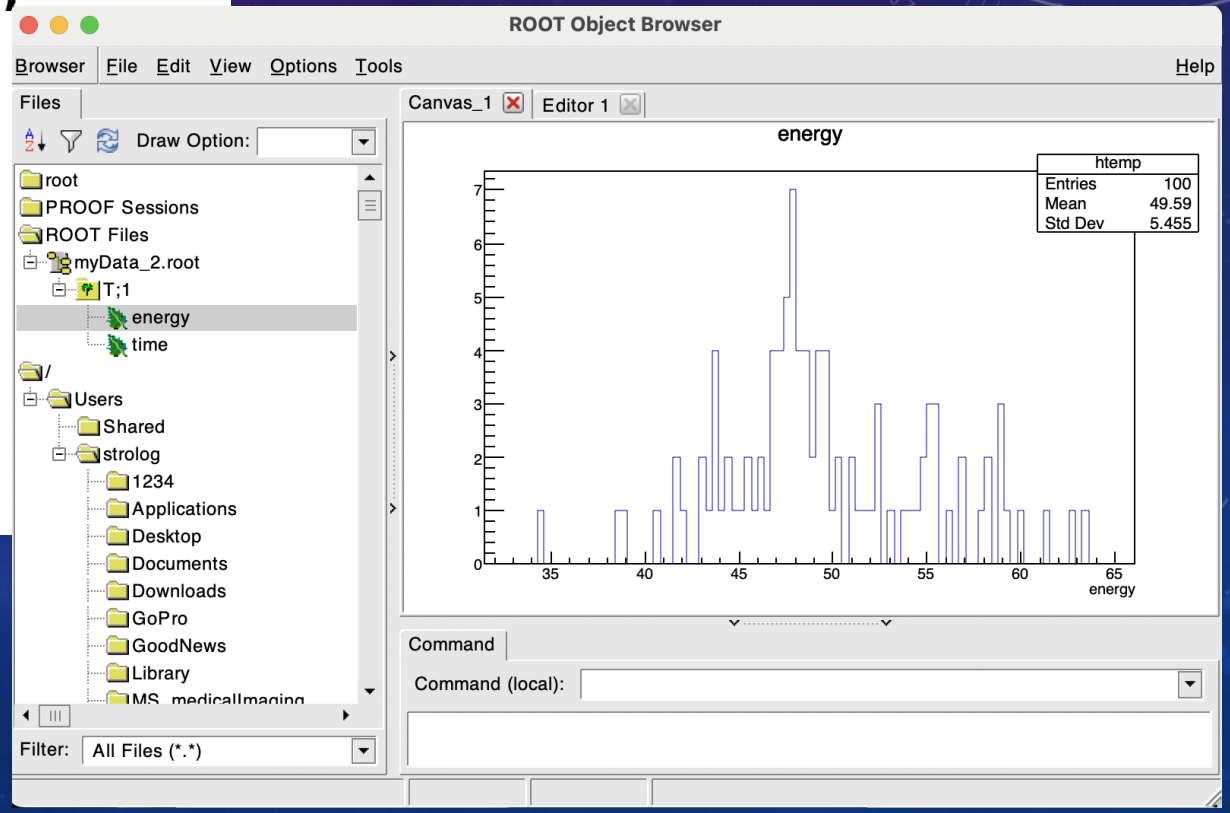


- Each event has one leaf → energy

CASE 2

```
void makeTree_2(){

  TFile *f = new TFile("myData_2.root", "recreate");
  TTree *tree = new TTree("T", "My First Tree");
  Double_t energy, time;
  tree->Branch("energy", &energy, "energy/D");
  tree->Branch("time", &time, "time/D");
  for (int i=0; i<100; i++) {
    energy = gRandom->Gaus(50, 5);
    time = gRandom->Uniform(2,5);
    tree->Fill();
  }
  tree->Write();
  f->Close();
}
```



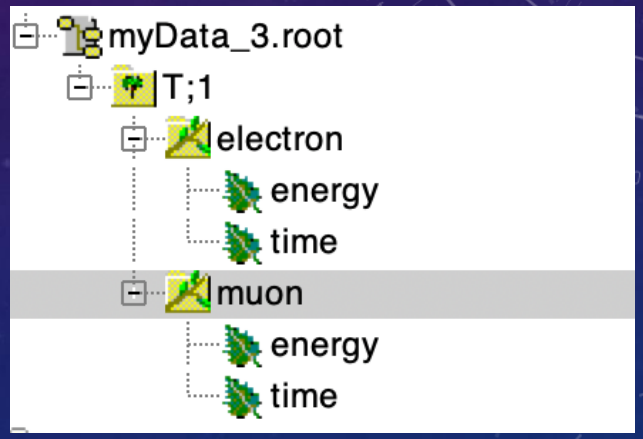


- Each event has two leaves → energy, time of detection

CASE 3

Each event has an electron and a muon

```
void makeTree_3(){

  TFile *f = new TFile("myData_3.root", "recreate");
  TTree *tree = new TTree("T", "My First Tree");
  Double_t a[2],b[2];
  tree->Branch("electron", a, "energy/D:time/D");
  tree->Branch("muon", b, "energy/D:time/D");
  for (int i=0; i<100; i++) {
    a[0] = gRandom->Gaus(50, 5);
    a[1] = gRandom->Uniform(2,5);
    b[0] = gRandom->Gaus(100, 10);
    b[1] = gRandom->Uniform(2,5);
    tree->Fill();
  }
  tree->Write();
  f->Close();
}
```

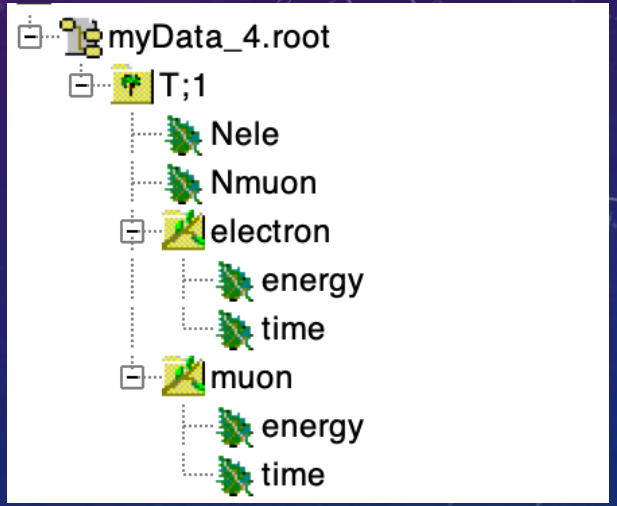


- Each event has two branches → electron and muon. Each branch has two leaves → energy and time

CASE 4

```
void makeTree_4(){

  TFile *f = new TFile("myData_4.root", "recreate");
  TTree *tree = new TTree("T", "My First Tree");
  Double_t a[2],b[2];
  Int_t aa=0,bb=0;
  tree->Branch("Nele", &aa, "Nele/I");
  tree->Branch("Nmuon", &bb, "Nmuon/I");
  tree->Branch("electron", a, "energy/D:time/D");
  tree->Branch("muon", b, "energy/D:time/D");
  for (int i=0; i<100; i++) {
    a[0] = gRandom->Gaus(50, 5);
    a[1] = gRandom->Uniform(2,5);
    aa=1;
    if (i%2==0){//every second event has a muon
      b[0] = gRandom->Gaus(100, 10);
      b[1] = gRandom->Uniform(2,5);
      bb=1;
    }
    else{
      b[0]=-1;
      b[1]=-1;
      bb=0;
    }

    tree->Fill();
  }
  tree->Write();
  f->Close();

}
```

Each event has an electron and either one or no muon



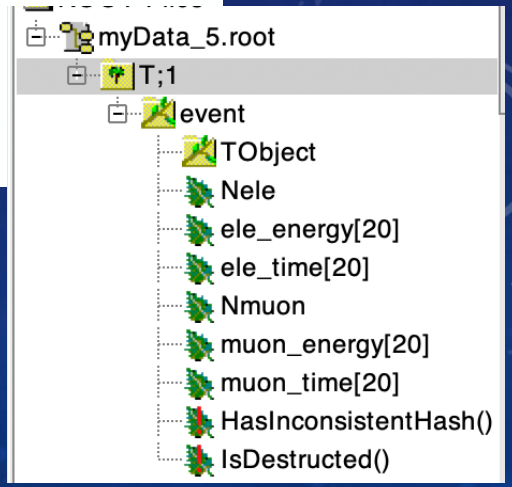
- Each event has two leaves (Number of electrons and muons) and two branches (electron and muon). Each branch has two leaves (energy and time)

CASE 5

```
class event : public TObject {
 public:

  Int_t Nele;
  Double_t ele_energy[20];
  Double_t ele_time[20];
  Int_t Nmuon;
  Double_t muon_energy[20];
  Double_t muon_time[20];

  event(){

    Nele=-1;
    Nmuon=-1;
    for (int i=0; i<20; i++) {

      ele_energy[i] = -1;
      ele_time[i]=-1;
      muon_energy[i]=-1;
      muon_time[i]=-1;

    }
  }
  ClassDefOverride(event,1);//
};
```

```
void makeTree_5(){

  TFile *f = new TFile("myData_5.root", "recreate");
  TTree *tree = new TTree("T", "My First Tree");
  event *ee = new event();

  tree->Branch("event", &ee, 80000, 1);

  for (int i=0; i<100; i++) {
    ee = new event();
    int NNe = (int)gRandom->Uniform(0,20);
    ee->Nele = NNe;
    for (int ii=0; ii<NNe; ii++){
      ee->ele_energy[ii] = gRandom->Gaus(50, 5);
      ee->ele_time[ii] = gRandom->Uniform(2,5);
    }
    int NNm = (int)gRandom->Uniform(0,20);
    ee->Nmuon = NNm;
    for (int ii=0; ii<NNm; ii++){
      ee->muon_energy[ii] = gRandom->Gaus(100, 5);
      ee->muon_time[ii] = gRandom->Uniform(2,5);
    }
    tree->Fill();
  }
  tree->Write();
  f->Close();
}
```

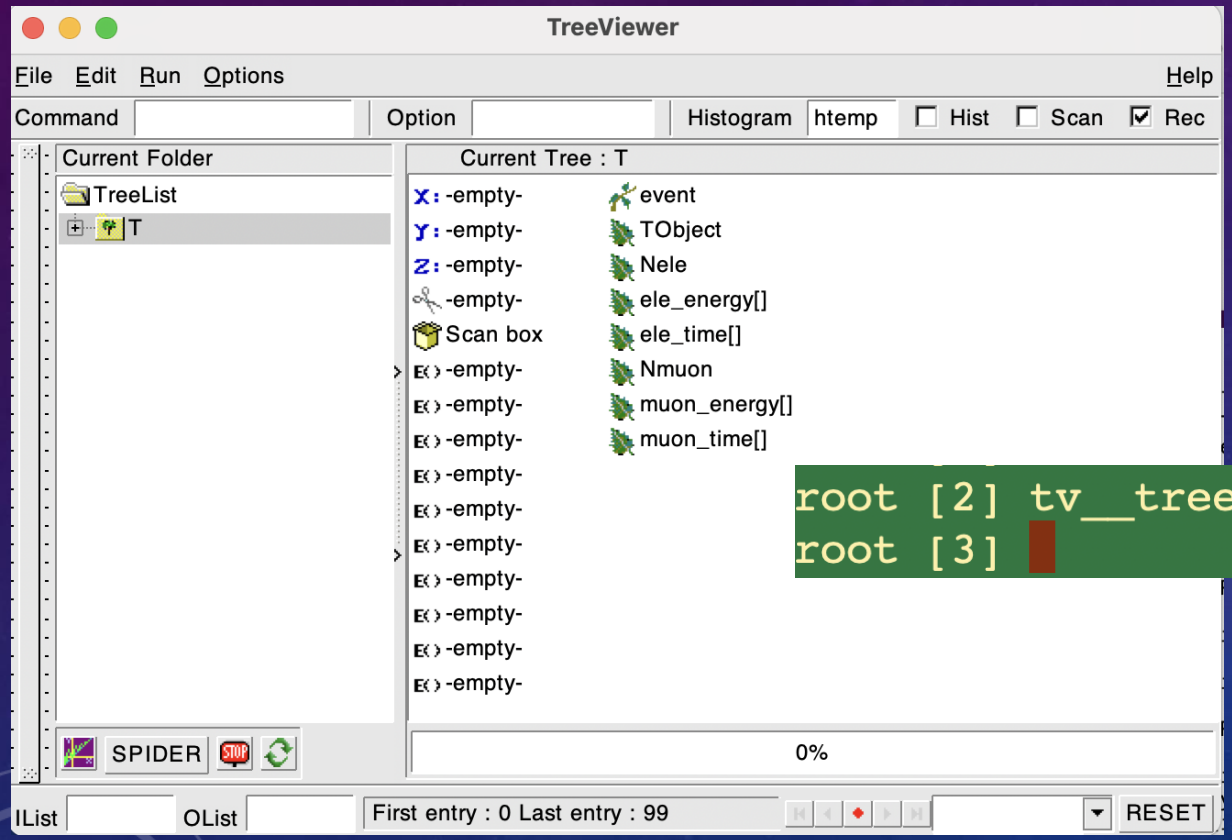
Each event has a random number of electrons and muons



- Each event has leaves of Number of electrons and muons and arrays that hold the energy and time of detection of each electron and muon

PLOTTING TTREE VARIABLES WITH TREEVIEWER

- The easiest way to plot a variable is to open the tree viewer



Quantity to plot

Selection (cut)

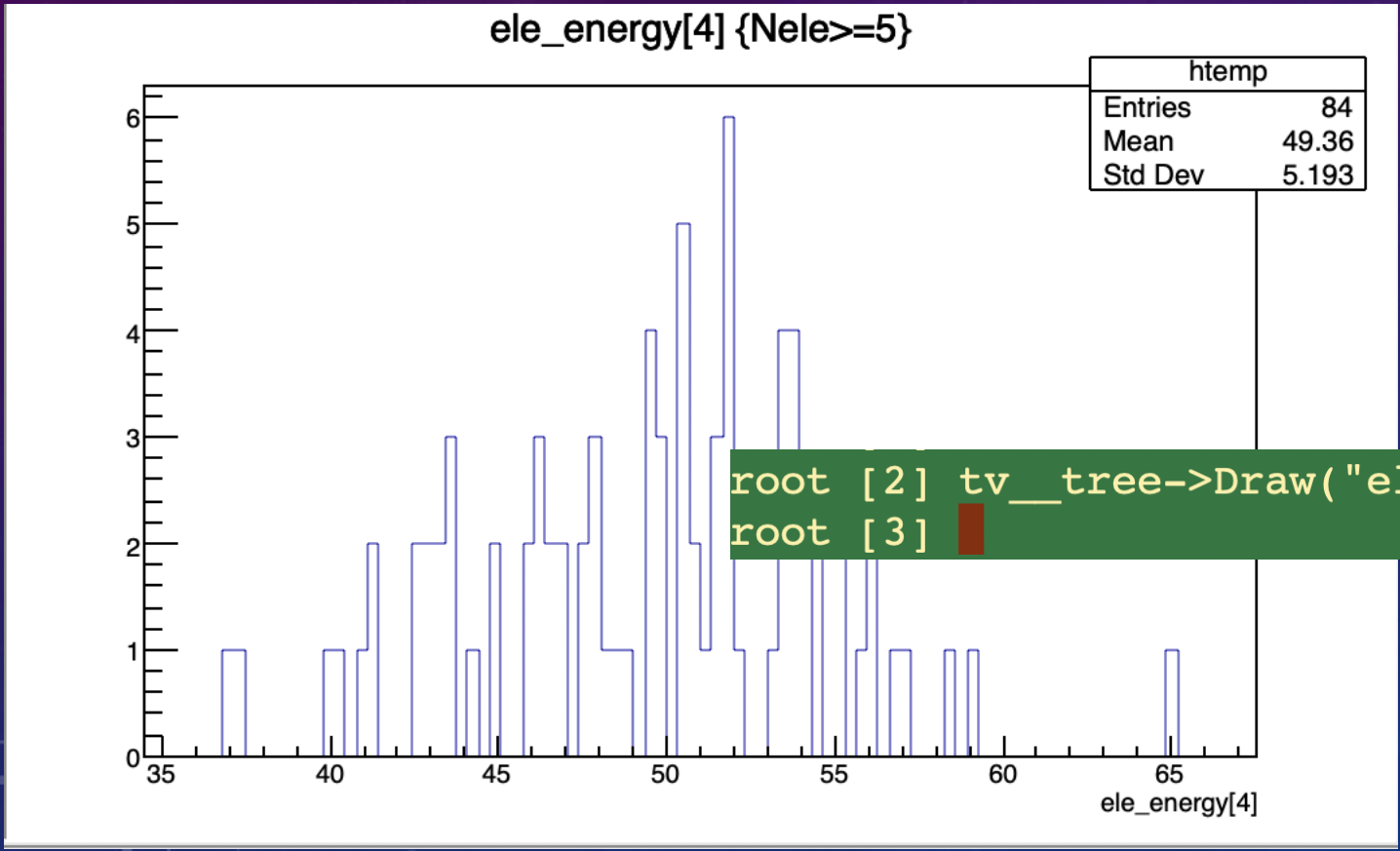
First event

```
root [2] tv__tree->Draw("ele_energy[4]","Nele>=5","", 100, 0);
root [3]
```

N of events

PLOTTING TTREE VARIABLES WITH TREEVIEWER

- The easiest way to plot a variable is to open the tree viewer




```
root [2] tv__tree->Draw("ele_energy[4]","Nele>=5","", 100, 0);
root [3]
```

Quantity to plot

Selection (cut)

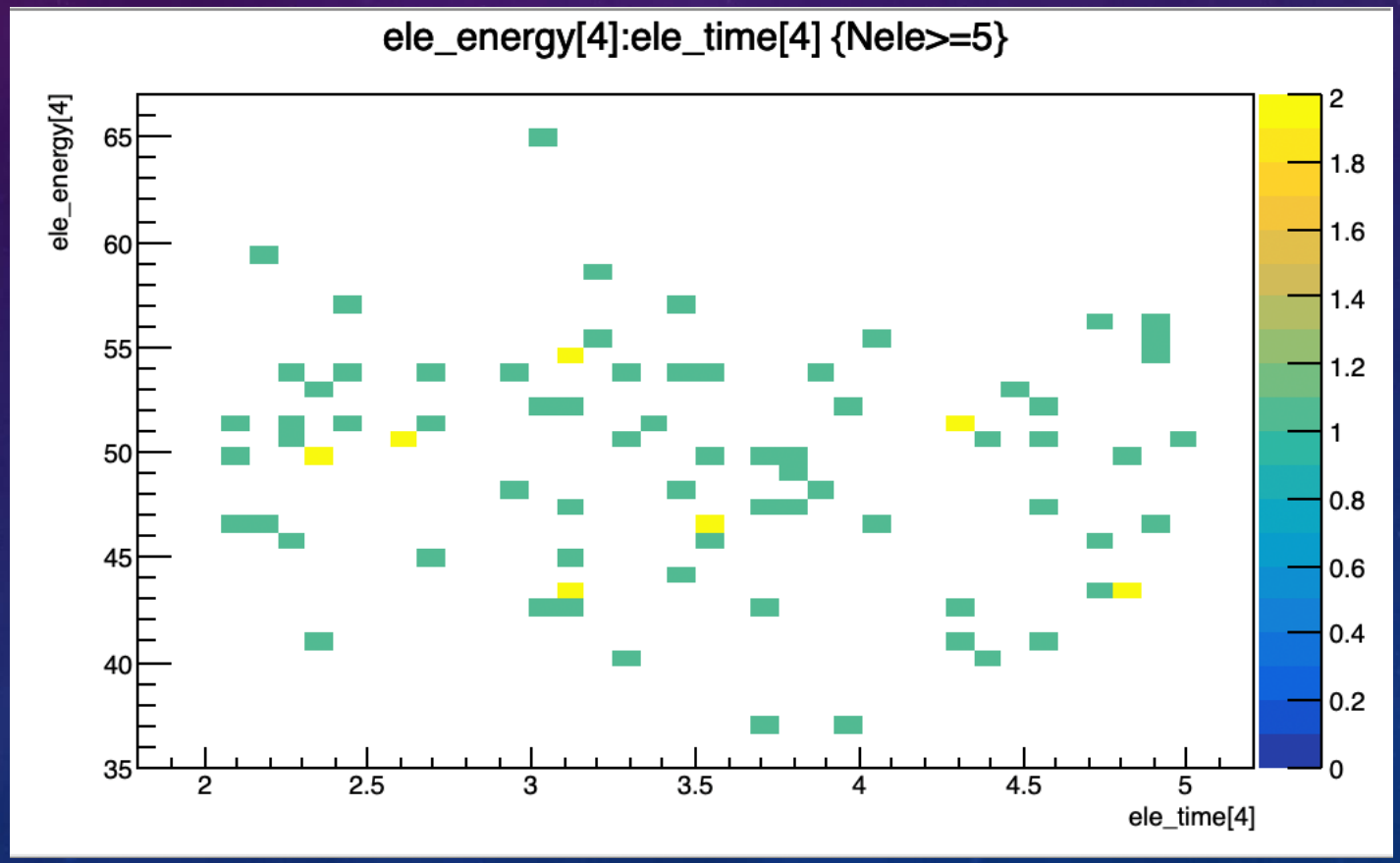
First event

N of events

PLOTTING TTREE VARIABLES WITH TREEVIEWER

- You can do it for 2D plots

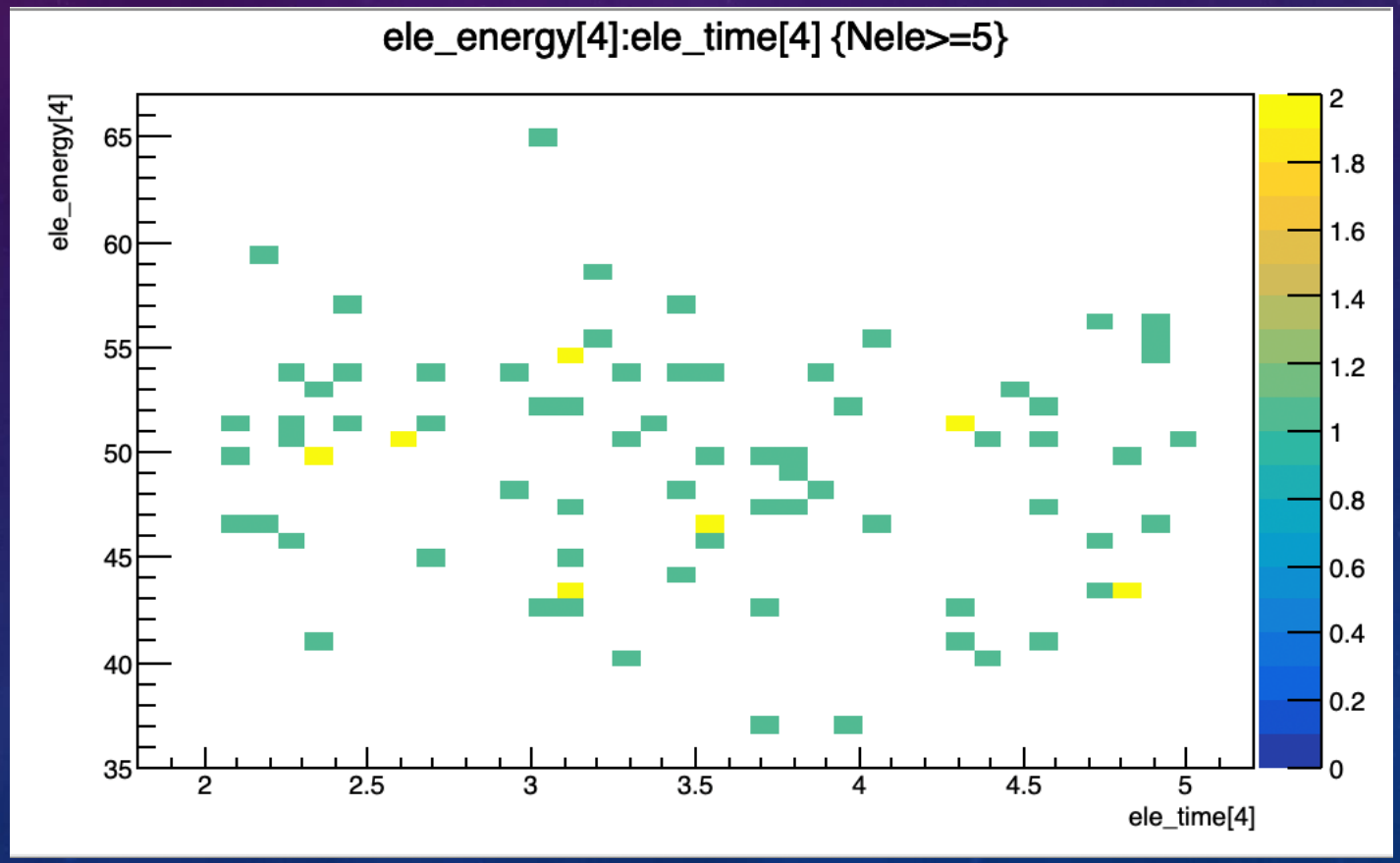
```
root [5] tv__tree->Draw("ele_energy[4]:ele_time[4]","Nele>=5","colz", 100, 0);
```



PLOTTING TTREE VARIABLES WITH TREEVIEWER

- You can do it for 2D plots (and save the result to a histogram)

```
tv__tree->Draw("ele_energy[4]:ele_time[4]>>histo2D","Nele>=5","colz", 100, 0);
```



TGRAPH

- TGraph is the class we use to plot a list of (x,y) pairs.
- There are many plot options
- When we have dx and/or dy errorbars, we use TGraphErrors()
- If the errors are asymmetric, we use TGraphAsymmErrors()