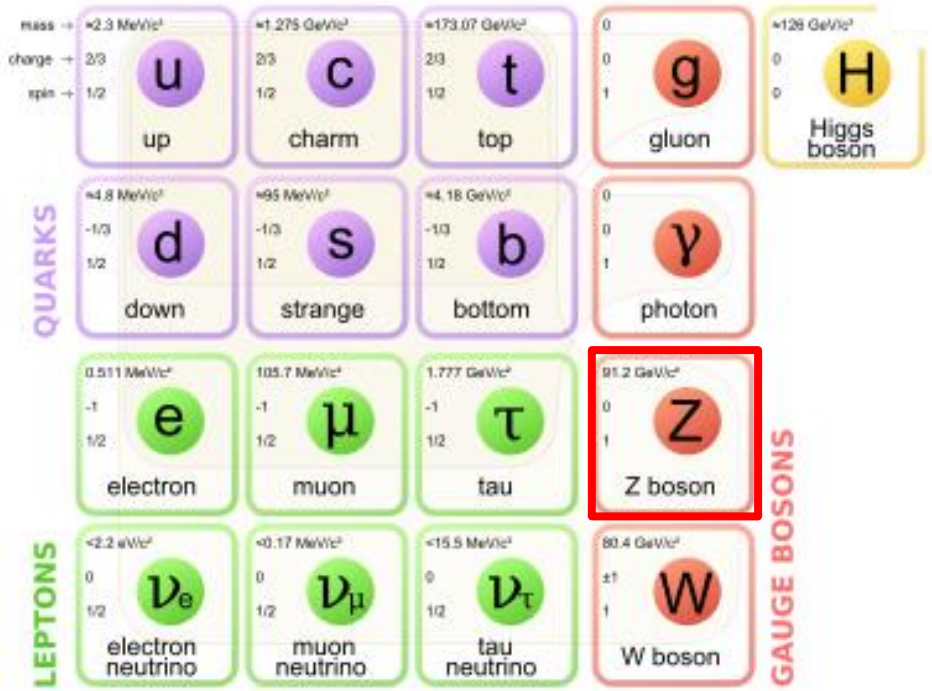
# Root Tutorial: Plotting the Z mass

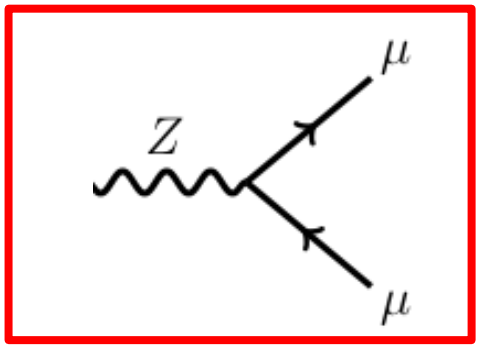
Nikolina Ilic

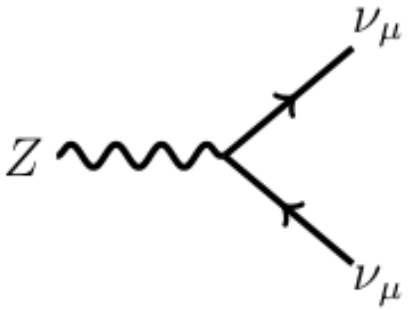
Stanford University

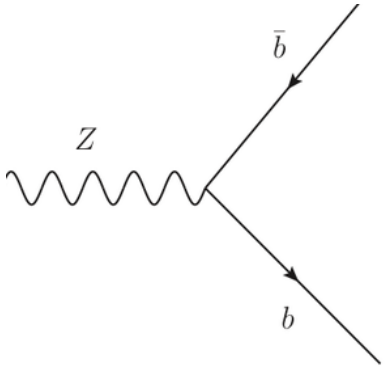
# Introduction

- Z and W are weak bosons (have integer spin – 1 in this case) that mediate the weak force, Z has charge 0, and mass 91.19 GeV
- Z can decay to 2 fermions (particles with half integer spin - quarks and leptons in this case) of the same type but **opposite charge**

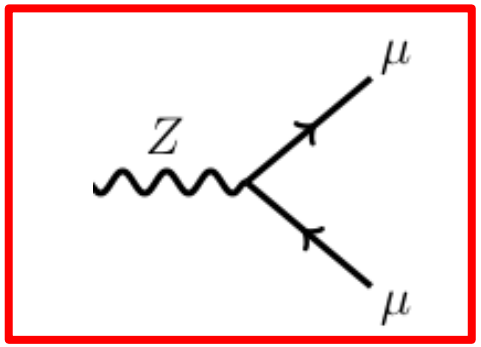


# Introduction




In our detectors we see the decay products of the Z boson -> the two muons ($\mu$)

In this tutorial we will

1. Identify the muons and construct the Z boson from them
2. Plot the mass of the Z boson
3. Plot the mass two muons in an event that are not decay products of the Z (the background)

The code you will have to fill into your macro is outlined in red boxes

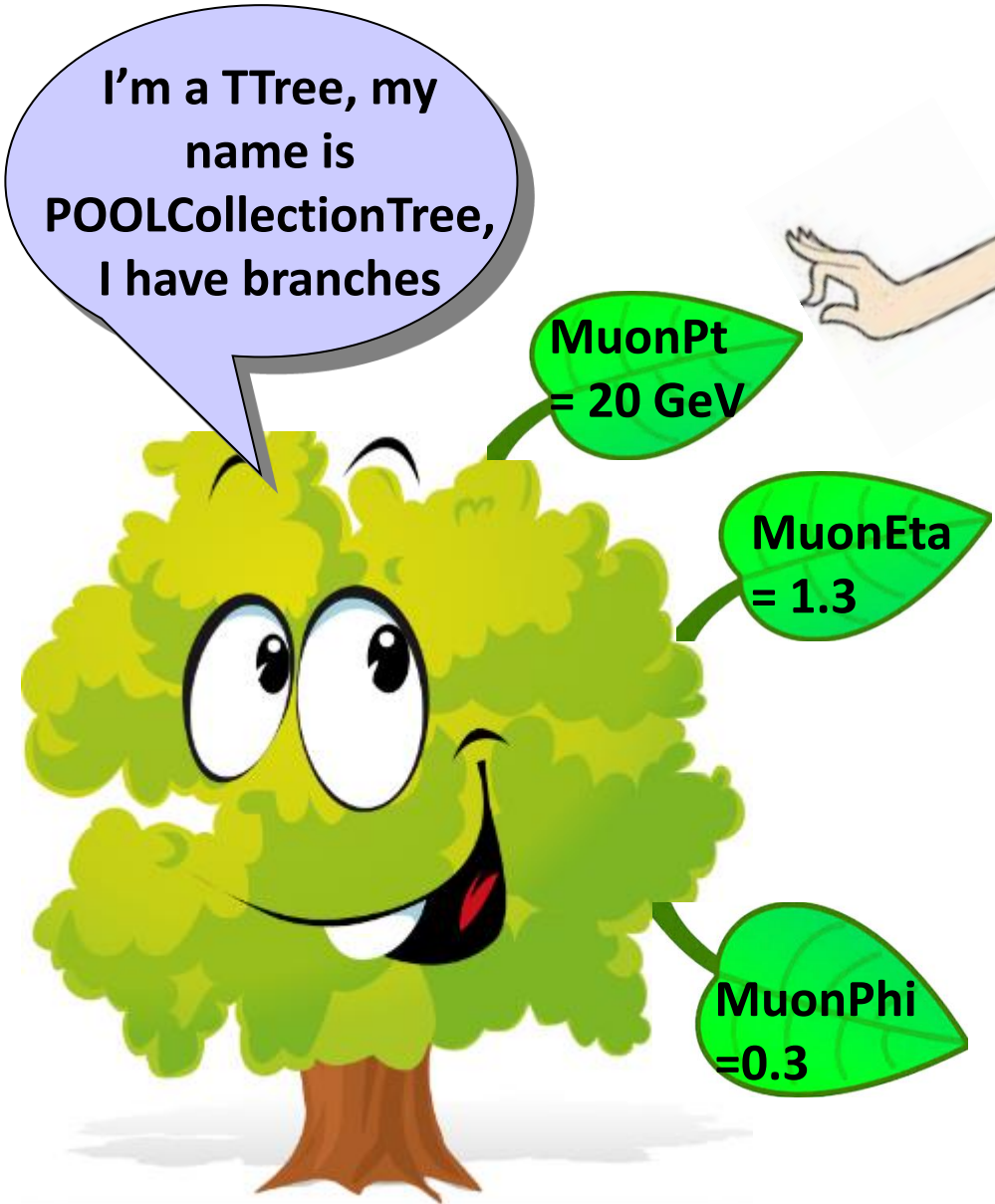
There are Questions along the tutorial labelled “Q”, answer in class or on paper

For more information take a look at the root class reference links on each slide

https://root.cern.ch/doc/master/classTBrowser.html

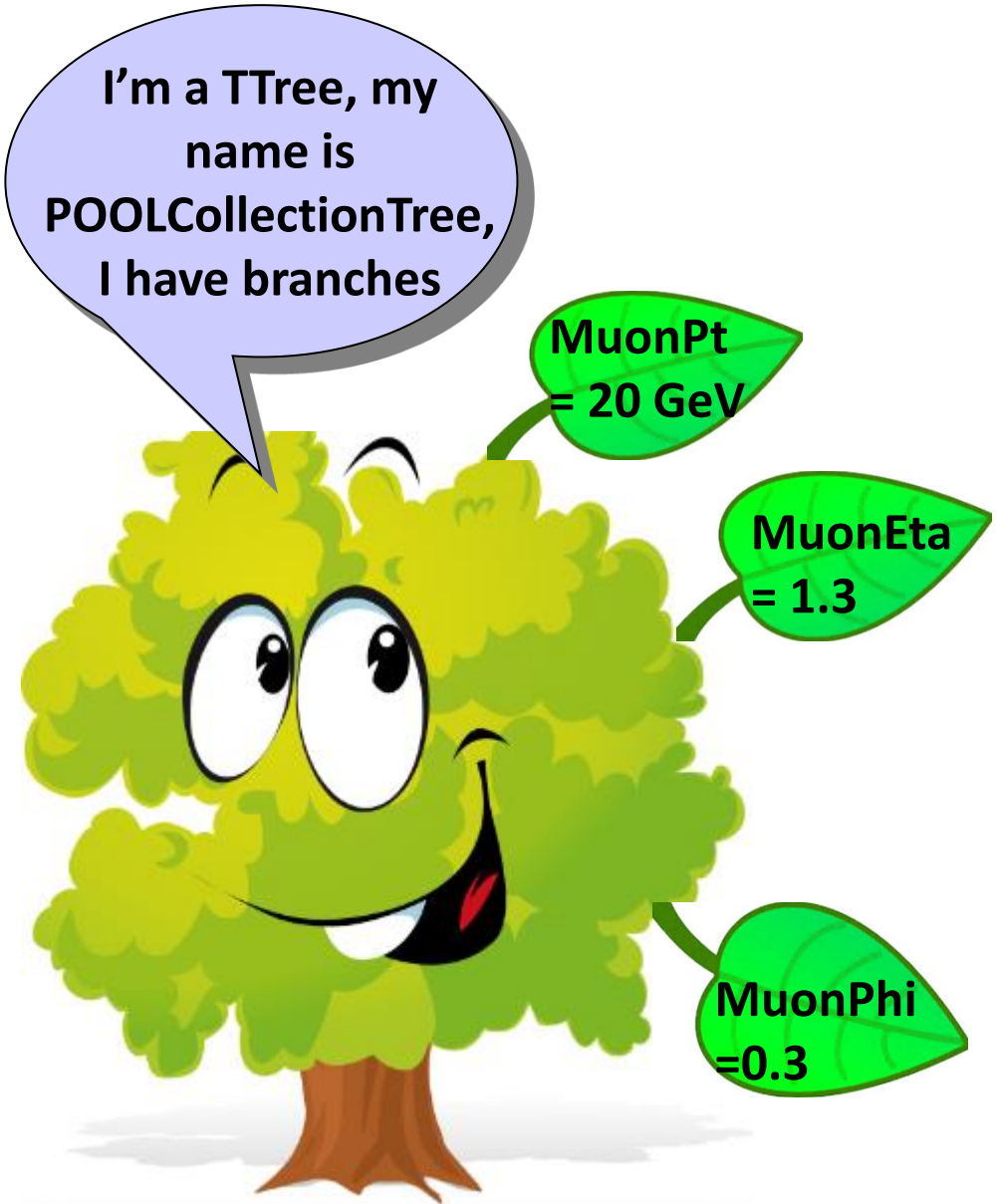
# What we will do in Root

**Event 1 from LHC Collision**

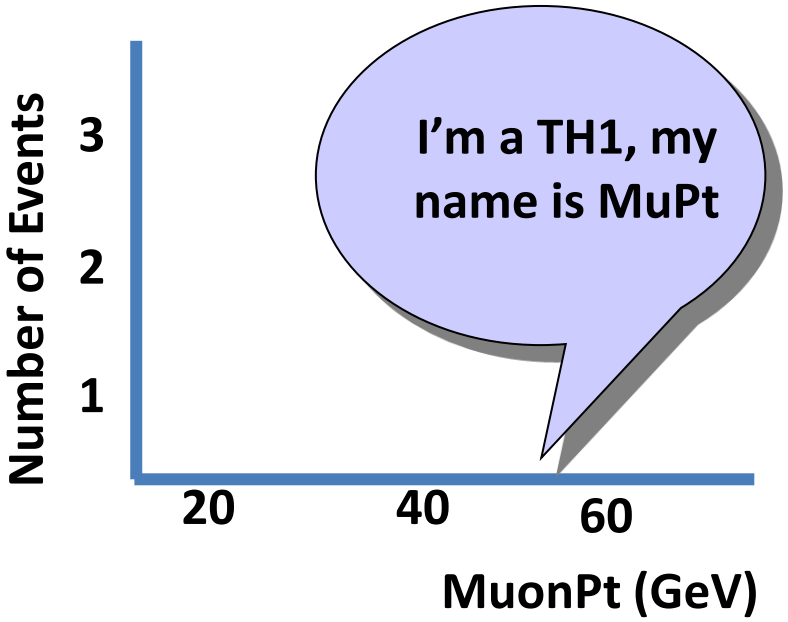


# What we will do in Root

**Event 1 from LHC Collision**

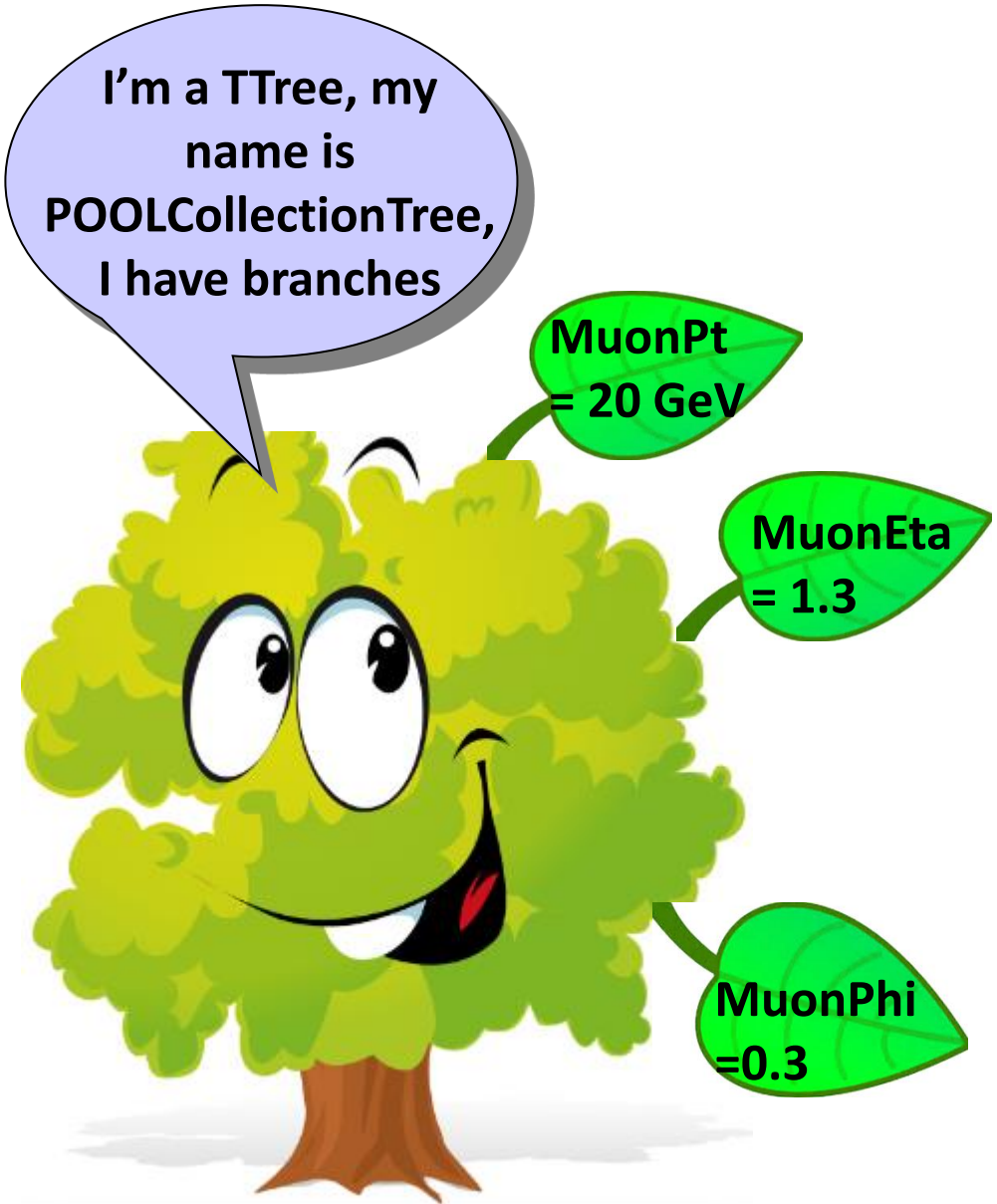



**Make a Histogram (TH1) with 3 bins**

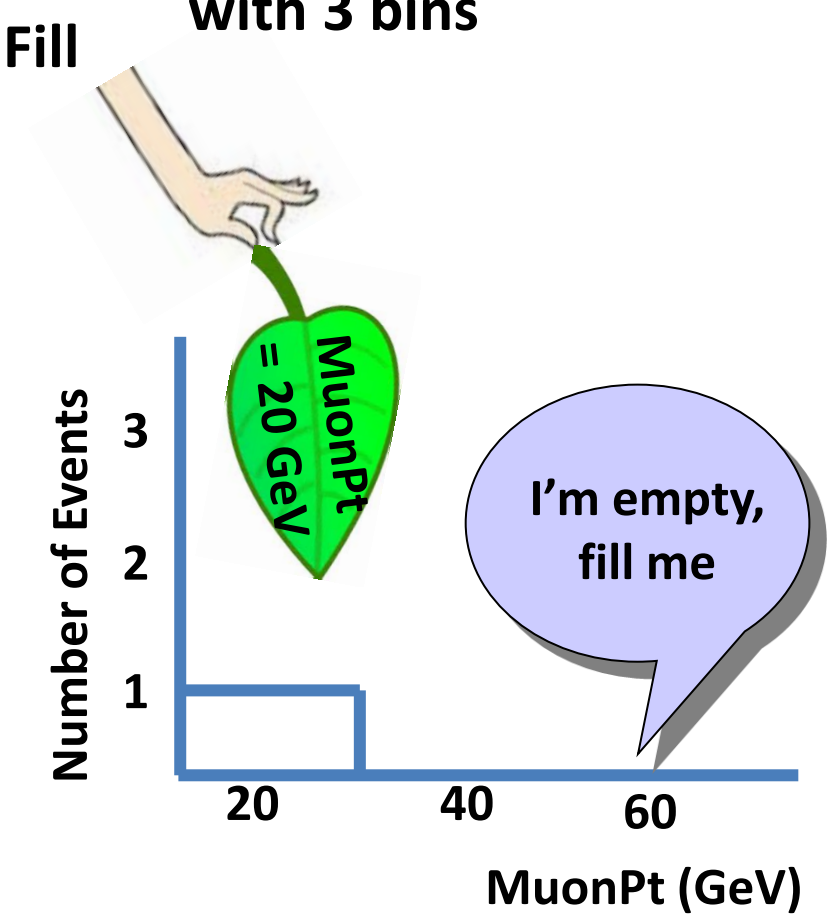


# What we will do in Root

**Event 1 from LHC Collision**

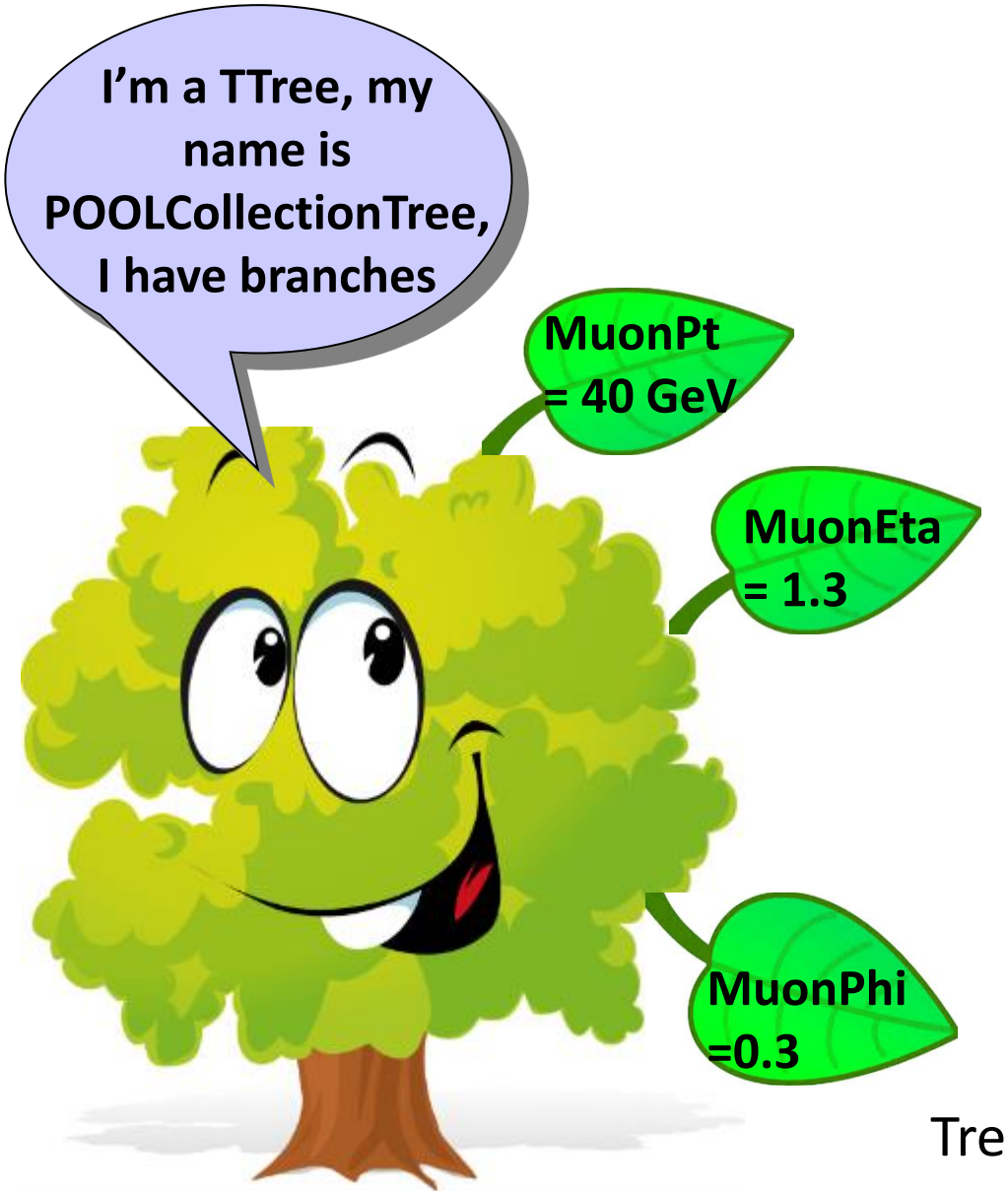



**Make a Histogram (TH1) with 3 bins**



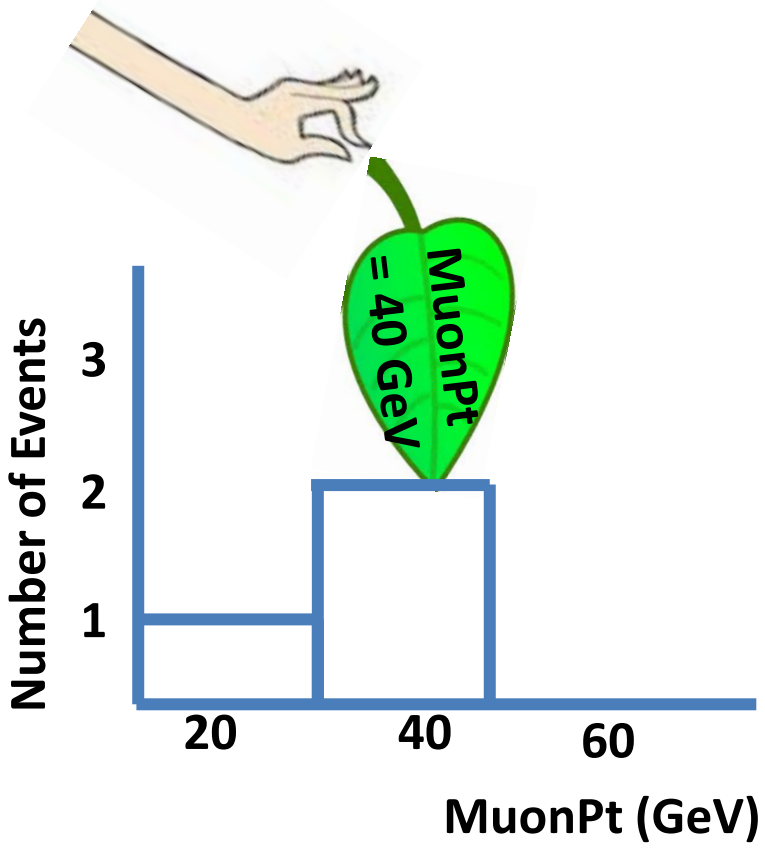
# What we will do in Root

**Event 2, 3 from LHC Collision:** go through all events to fill histograms




**Make a Histogram (TH1) with 3 bins**

**Fill**




Tree +branches sit in a single .root file

# Download and Inspect .root file

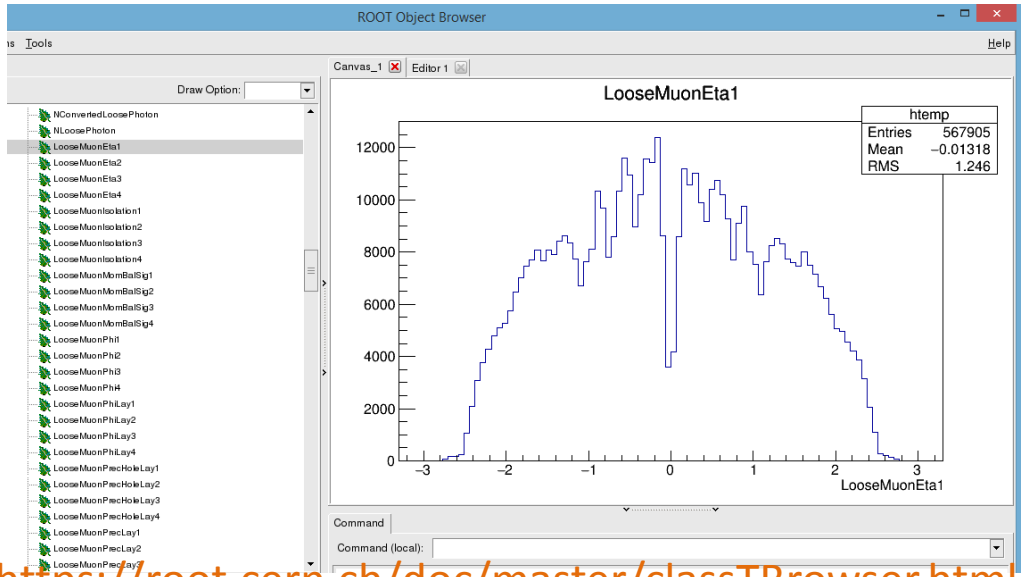
- Let's download the muons.root file, in your terminal type:

```
mkdir ZPeak
cd ZPeak
cp /afs/cern.ch/user/n/nilic/public/APS_Zpeak/APS/Zpeak_2012/muons.root .
```

- Let's inspect the file in a Tbrowser, in your terminal type:

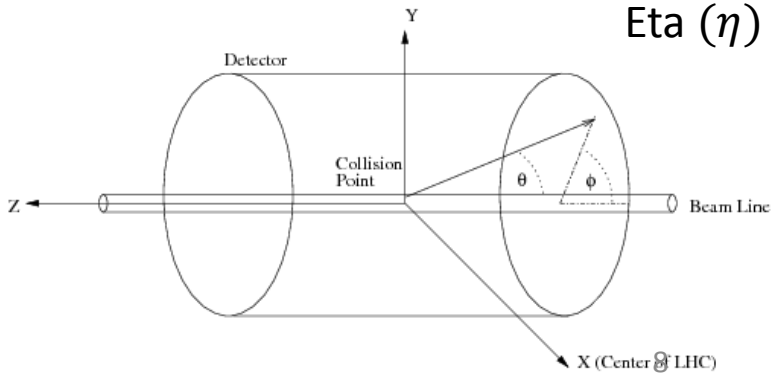
```
root -l muons.root
Attaching file muons.root as _file0...
TBrowser b
```

- Find and look these leaves located in the POOLCollectionTree: LooseMuonEta1, LooseMuonPhi1, LooseMuonPt1




https://root.cern.ch/doc/master/classTBrowser.html

Pt is the momentum of the muons. The location of the muons is their eta, phi, coordinates



# Download and Inspect our Root Macro

- Let's download and open our root macro

```
cp /afs/cern.ch/user/n/nilic/public/APS_Zpeak/APS/Zpeak_2012/findZ.C .
emacs findZ.C
```

- The top of the file includes the header files we will use that define the root objects we will work with

```
#include "TFile.h"
#include "TTree.h"
#include "TCanvas.h"
#include "TH1F.h"
#include <iostream>

using namespace std;
```

https://root.cern.ch/doc/v606/classTFile.html
https://root.cern.ch/doc/master/classTTree.html
https://root.cern.ch/doc/master/classTCanvas.html
https://root.cern.ch/doc/master/classTH1F.html

allows us to use cout, cin, string vector etc

- The main function we will fill and run is readEvents()

# Fill readEvents(): Task 1

Load our file and read the tree's entries

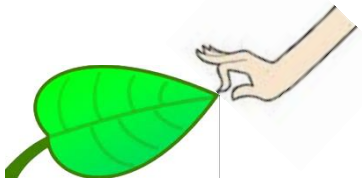
```
void readEvents(){

  //TASK 1*****************************************************************
  //Load the file
  TFile * f = new TFile("muons.root");

  //Get the tree(called POOLCollectionTree) from the file
  TTree *tree = (TTree *) f->Get("POOLCollectionTree");

  //Get entries of the file
  int nEntries = tree->GetEntries();

  //Here, YOU print the number of entries the file found
  cout << "There are " << nEntries << " entries in your ntuple" << endl;
```



https://root.cern.ch/doc/master/classTTree.html

# Let's Fill readEvents(): Task 2

Create local variables for the tree's branches.

- Create variable the variable of the type Uint_t for NLooseMuons (number of muons in the event)
- Create variables of the type Float_t for LooseMuonsEta1, LooseMuonsPhi1, LooseMuonsPt1, LooseMuonsEta2, LooseMuonsPhi2, LooseMuonsPt2

```
//TASK 2*********************************************
// create local variables for the tree's branches
// We need Float_t type for LooseMuonsPhi1, LooseMuonsPt1, L
UInt_t NLooseMuons;
Float_t LooseMuonsEta1;
Float_t LooseMuonsPhi1;
Float_t LooseMuonsPt1;

Float_t LooseMuonsEta2;
Float_t LooseMuonsPhi2;

//Here YOU define a Float_t for LooseMuonPt2
Float_t LooseMuonsPt2;
```

# Let's Fill readEvents(): Task 3

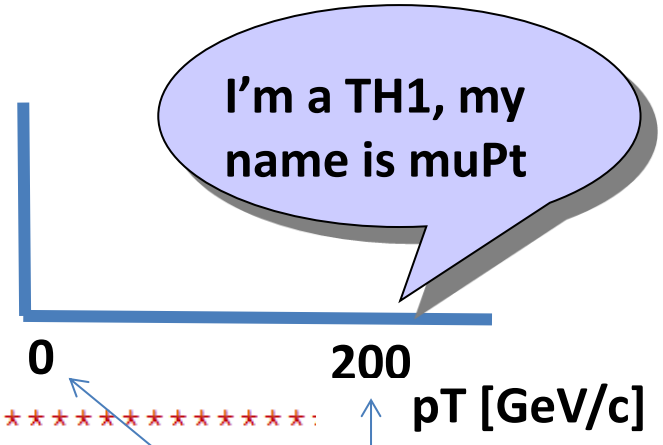
Set the tree's branches to the local variables you defined (LooseMuonsEta1, LooseMuonsPhi1, LooseMuonsPt1, LooseMuonsEta2, LooseMuonsPhi2, LooseMuonsPt2)

```
//Task 3*************************************************
// set the tree's braches to the local variables
tree->SetBranchAddress("NLooseMuon", &NLooseMuons);
tree->SetBranchAddress("LooseMuonEta1", &LooseMuonsEta1);
tree->SetBranchAddress("LooseMuonPhi1", &LooseMuonsPhi1);
tree->SetBranchAddress("LooseMuonPt1", &LooseMuonsPt1);
tree->SetBranchAddress("LooseMuonEta2", &LooseMuonsEta2);
tree->SetBranchAddress("LooseMuonPhi2", &LooseMuonsPhi2);

//Here YOU set a tree's branch to LooseMuonPt2
tree->SetBranchAddress("LooseMuonPt2", &LooseMuonsPt2);
```

# Let's Fill readEvents(): Task 4

- Declare histograms for the momentum, eta, phi and energy of the muons, as well as Zmass of Z events and backgrounds events




```
//Task 4*****************************************************
//declare some histograms
//declare a histogram called muPt, with 50 bins, and range from 0 to 200
TH1F *muPt = new TH1F("muPt", ";p_{T} [GeV/c];Events", 50, 0, 200);

//declare a histogram called muEta, with 50 bins and range -3 to 3
TH1F *muEta = new TH1F("muEta", ";#eta;Events", 50, -3, 3);

//declare a histogram called muPhi with 50 bins and range from -4 to 4
TH1F *muPhi = new TH1F("muPhi", ";#phi;Events", 50, -4, 4);

//declare a histogram called muE with 50 bins and range from 0 to 200
TH1F *muE = new TH1F("muE", ";Energy;Events", 50, 0, 200);

//declare a histogram called Zmass with 50 bins and range from 0 to 200
TH1F *Zmass = new TH1F("Zmass", ";Z Mass;Events", 50, 0, 200);

// Here, YOU declare a histogram called ZmassBkg with 50 bins and range
TH1F *ZmassBkg = new TH1F("ZmassBkg", ";Z Mass;Events", 50, 0, 200);
```

# Inside Event Loop

- Now lets go inside our loop over all events (entries) and fill our histograms for each event
- All of the code within the next slides will be within the EVENT LOOP. This includes TASK A, B, C

```
// loop over each entry (event) in the tree
for( int entry=0; entry<10000; entry++ ){




}//THIS IS THE END OF OUR ENTRIES LOOP
```

# Inside Event Loop: Task A

- Inside the event loop, print out the number of entries (events) every 10 000, and check the event is read properly

```
// loop over each entry (event) in the tree
for( int entry=0; entry<10000; entry++ ){
  //Task A**************************************
  if( entry%10000 == 0 ) cout << "Entry:" << entry << endl;

  // check that the event is read properly
  int entryCheck = tree->GetEntry( entry );
  if( entryCheck <= 0 ){  continue; }
```

# Inside Event Loop: Task B

- Make selections to try separate Z bosons from background
  - Select at least 2 leptons, require that their Pt is > 20 GeV
  - Make a LorentzVector for the muons

```
//Task B***********************************************************
// only look at events containing at least 2 leptons
if(NLooseMuons < 2) continue;

// require the leptons to be greater than 20 GeV
if(abs(LooseMuonsPt1) *0.001 < 20 ) continue;

//here YOU require LooseMuonsPt2 > 20 GeV
if(abs(LooseMuonsPt2) *0.001 < 20 ) continue;

// make a LorentzVector from the muon
TLorentzVector Muons1;
Muons1.SetPtEtaPhiM(fabs(LooseMuonsPt1), LooseMuonsEta1, LooseMuonsPhi1, 0);

//here YOU define a LorentzVector for Muons2
TLorentzVector Muons2;
Muons2.SetPtEtaPhiM(fabs(LooseMuonsPt2), LooseMuonsEta2, LooseMuonsPhi2, 0);

// print out the details of a muon every so often
```

Q1: Why do we multiply the Muon Pt by 0.001?

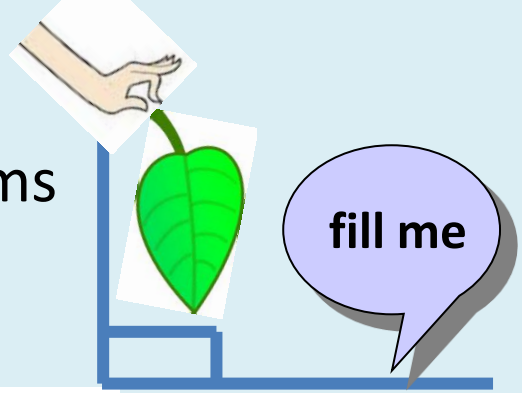
Q2: If we want the muon Pt to be greater than 100, why do we use "<"

Q3: What does the SetPtEtaPhiM function of the TLorentzVector object do?

https://root.cern.ch/doc/master/classTLorentzVector.html

# Inside Event Loop: Task C

- Fill the muPt, muEta, muPhi and muE histograms
- Define the Z boson vector
- Fill the Zmass and ZmassBkg histograms




MuonPt (GeV)

```
  //Task C*****************************************************
  // fill our histograms
  muPt->Fill(Muons1.Pt()*0.001);
  muEta->Fill(Muons1.Eta());
  muPhi->Fill(Muons1.Phi());

  //Here, YOU fill the muE histogram
  muE->Fill(Muons1.E()*0.001);

  //Define the Z boson vector as a sum of the 2 muon vectors
  TLorentzVector Z = Muons1 + Muons2;

  //Fill the Zmass and ZmassBkg histograms
  if((LooseMuonsPt1 *LooseMuonsPt2) < 0 ){
    Zmass->Fill(Z.M()*0.001);
  }
  else{
   ZmassBkg->Fill(Z.M()*0.001);
  }

}//THIS IS THE END OF OUR ENTRIES LOOP
```

- This is the last thing we do inside the Event loop

Q4: Why we fill the Zmass histogram if LooseMuonsP1*LooseMuonsPt2 < 0?

# Let's Fill readEvents(): Task 5

- Now we go back outside the event loop
- Draw the Zmass histogram
- Make a TFile called histograms.root
- Write the muPt, muEta, muPhi, muE, Zmass, ZmassBkg histograms to the file

```
}//THIS IS THE END OF OUR ENTRIES LOOP


//Task 5*****************************************************
// Here, YOU draw the Zmass distribution
Zmass->Draw();

// make a ROOT output file to store your histograms
TFile *outFile = new TFile("histograms.root", "recreate");
muPt->Write();
muEta->Write();
muPhi->Write();
muE->Write();
Zmass->Write();

//Here, YOU write the ZmassBkg histogram
ZmassBkg->Write();

}// END OF readEventsLoop
```

# Now Let's Run our Macro

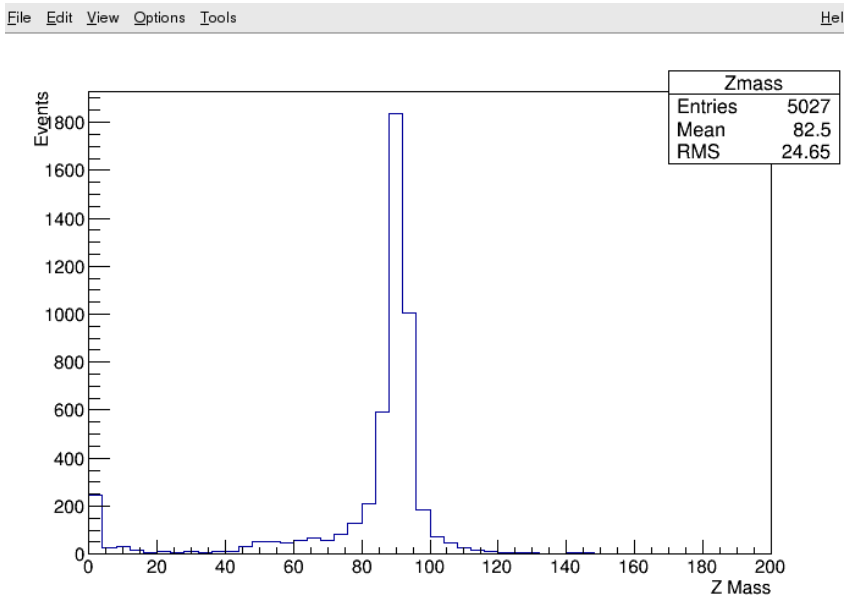
- Run your macro, click SAVE in gedit and close it then in terminal type:

```
root -l
root [0] .L findZ.C
root [0] readEvents()
```

- If your code is working you should see the entries:

```
There are 567905 entries in your
ntuple
Entry:0
Entry:10000
Entry:20000
```

- You should see your Zmass histogram pop up
- If you exit root (press .q), you should see your histograms.root file created



# Now Let's Run our Macro

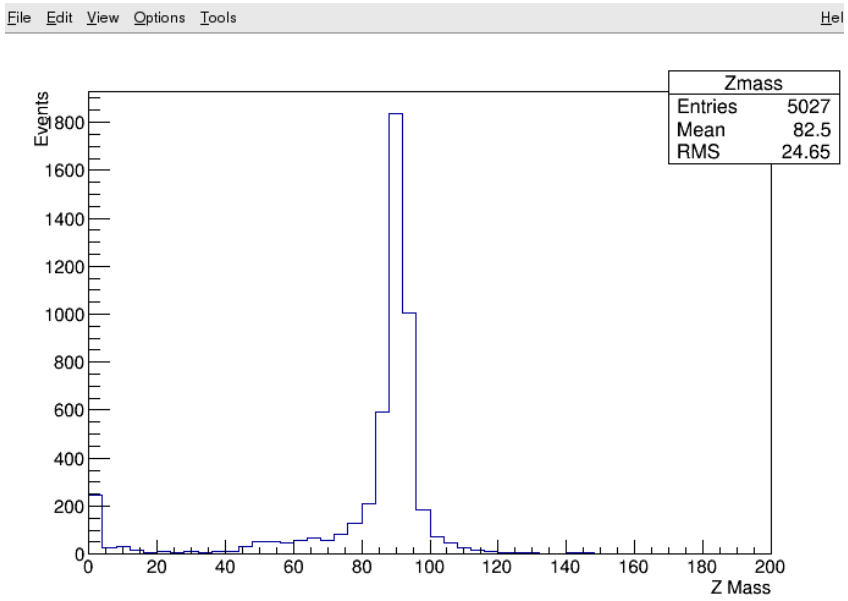
- If that gave you errors, you made a typo in findZ.C, so to save time let's use our already filled in macro findZ_full.C

```
cp  /afs/cern.ch/user/n/nilic/public/APS_Zpeak/APS/Zpeak_2012/findZ_full.C .
root -l
root [0] .L findZ_full.C
root [0] readEvents()
```

- If your code is working you should see the entries:

```
There are 567905 entries in your
ntuple
Entry:0
Entry:10000
Entry:20000
```

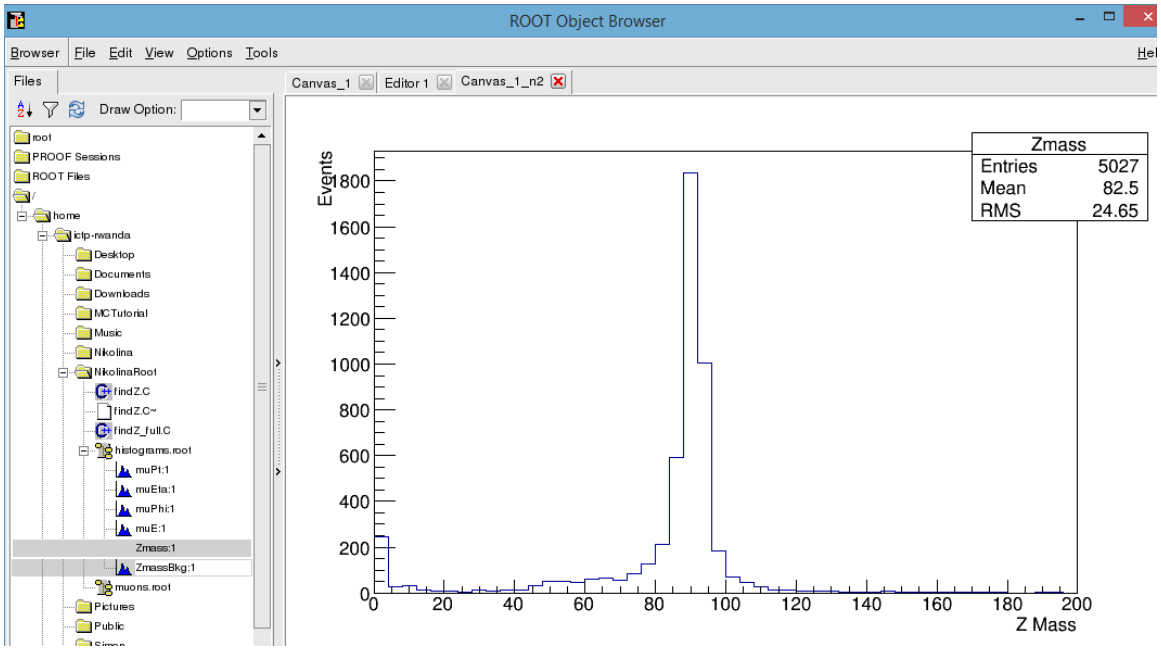
- You should see your Zmass histogram pop up
- If you exit root (press .q), you should see your histograms.root file created



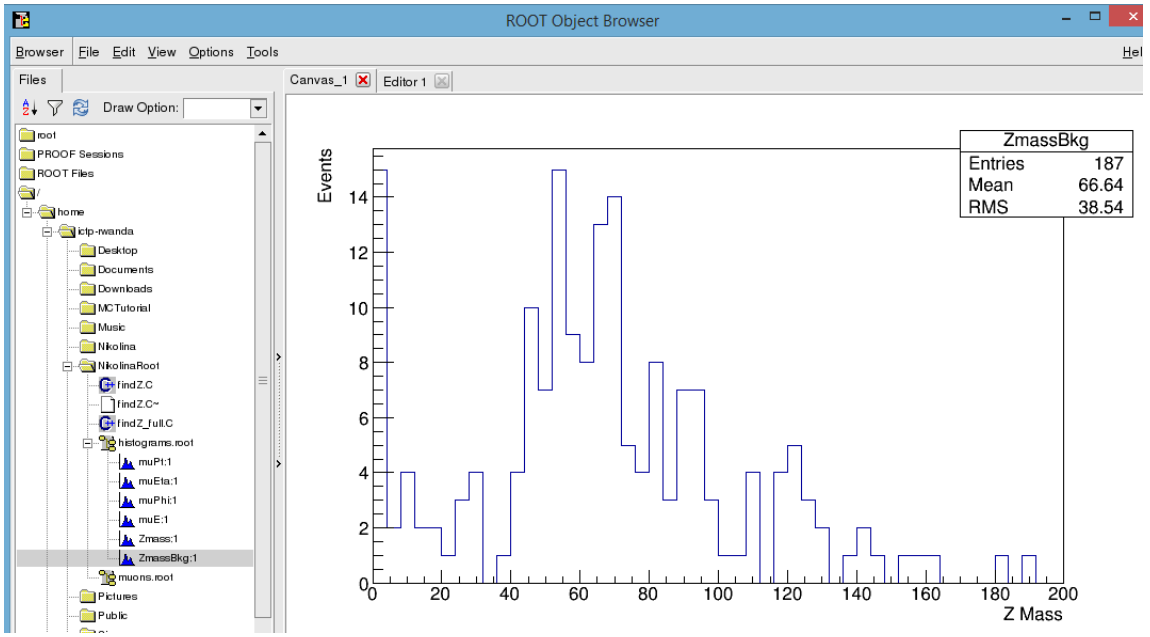
# Let’s inspect the file We Made

- Open a Tbrowser and look at your histograms.root file

```
root -l histograms.root
TBrowser b
```




- Q5: What is the difference between the Z mass from the muons that came from a Z boson and ones that did not?



# If you have finished early

- Change your code to run over more events (max number of events is nEntries), make different cuts
- Explore ATLAS open data
  - http://atlas-opendata.web.cern.ch/atlas-opendata/visualisations/analyser-js.php

# Homework

- Download the electrons.root file and find the Z peak in which the Z decays to electrons!

# BACKUP

# vim text editor

- To open file type
  - vi filename
- To exit vim and save your file, press Esc to get into “command mode” , then press Shift zz
- To exit without saving press Esc to get into command mode, then Shift : q! ,type quite

# Emacs text editor

- To open file name
  - emacs filename
- To exit emacs press Ctrl x, when asked if you want to save, type yes if you want to save (no if you don’t)
  - When asked to about modified buffer exit anyways, type yes