Reconstructed Root Tree

Shyam Kumar
Shyam.kumar@ba.infn.it

ePIC software tutorial

March 7-8, 2023

Please contact if there is any further doubt or questions

Reading an Output Tree

root [2] events->Show(EventNo)

// using TTreeReaderArray in root

```
TFile* file = new TFile("tracking_test_gun.edm4eic.root");
TTreeReader myReader("events", file);
// MC Information
TTreeReaderArray<Float_t> charge(myReader, "MCParticles.charge");
TTreeReaderArray<Float_t> px_mc(myReader, "MCParticles.momentum.x");
TTreeReaderArray<Float_t> py_mc(myReader, "MCParticles.momentum.y");
TTreeReaderArray<Float_t> pz_mc(myReader, "MCParticles.momentum.z");
// Reco Information
TTreeReaderArray<Float_t> px_rec(myReader,"ReconstructedChargedParticles.momentum.x");
TTreeReaderArray<Float_t> py_rec(myReader,"ReconstructedChargedParticles.momentum.y");
TTreeReaderArray<Float_t> pz_rec(myReader,"ReconstructedChargedParticles.momentum.z");
```

```
root -l tracking_test_gun.edm4eic.root
root [1] .ls
TFile**         tracking_test_gun.edm4eic.root
 TFile*         tracking_test_gun.edm4eic.root
  KEY: TTree    events;1        Events tree
  KEY: TTree    metadata;1      Metadata tree
  KEY: TTree    run_metadata;1  Run metadata tree
  KEY: TTree    evt_metadata;1  Event metadata tree
  KEY: TTree    col_metadata;1  Collection metadata tree
root [2] events->Print()
```

```
*Br   13 :MCParticles.momentum.x : Float_t x[MCParticles_]                 *
*Entries :  3000000 : Total  Size=   60555359 bytes  File Size  =   51929166 *
*Baskets :      140 : Basket Size=    2293248 bytes  Compression=   1.17     *
*.............................................................................*
*Br   14 :MCParticles.momentum.y : Float_t y[MCParticles_]                 *
*Entries :  3000000 : Total  Size=   60555359 bytes  File Size  =   51929516 *
*Baskets :      140 : Basket Size=    2293248 bytes  Compression=   1.17     *
*.............................................................................*
*Br   15 :MCParticles.momentum.z : Float_t z[MCParticles_]                 *
*Entries :  3000000 : Total  Size=   60555359 bytes  File Size  =   51899040 *
*Baskets :      140 : Basket Size=    2293248 bytes  Compression=   1.17     *
*                                                                           *
```

You have to correctly specify data type (int, float, double), How do we know?

One way (using Next)

```
 while (myReader.Next())
 {
// True Particle
 for (int imcPart = 0; imcPart < charge.GetSize(); ++imcPart){
   double  pmc = sqrt(px_mc[imcPart]*px_mc[imcPart]
+py_mc[imcPart]*py_mc[imcPart]+pz_mc[imcPart]*pz_mc[imcPart]);
   }
// Reco Particle
 for (int irecPart = 0; irecPart <  px_rec.GetSize(); ++irecPart){
   double  prec = sqrt(px_rec[irecPart]*px_rec[irecPart]
+py_rec[irecPart]*py_rec[irecPart]+pz_rec[irecPart]*pz_rec[irecPart]);
   }
}
```

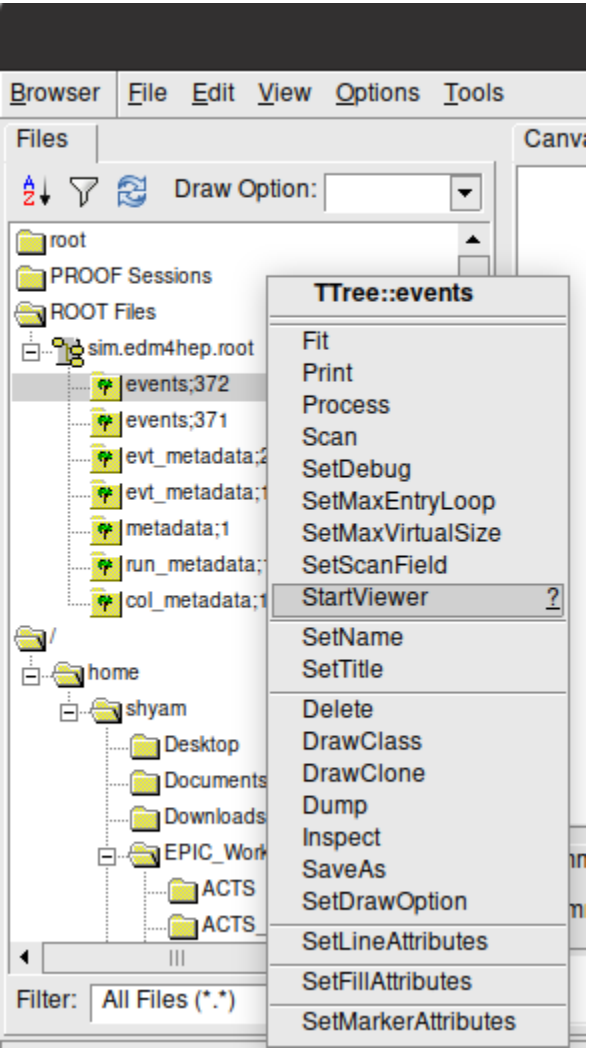
Other way (For loop)

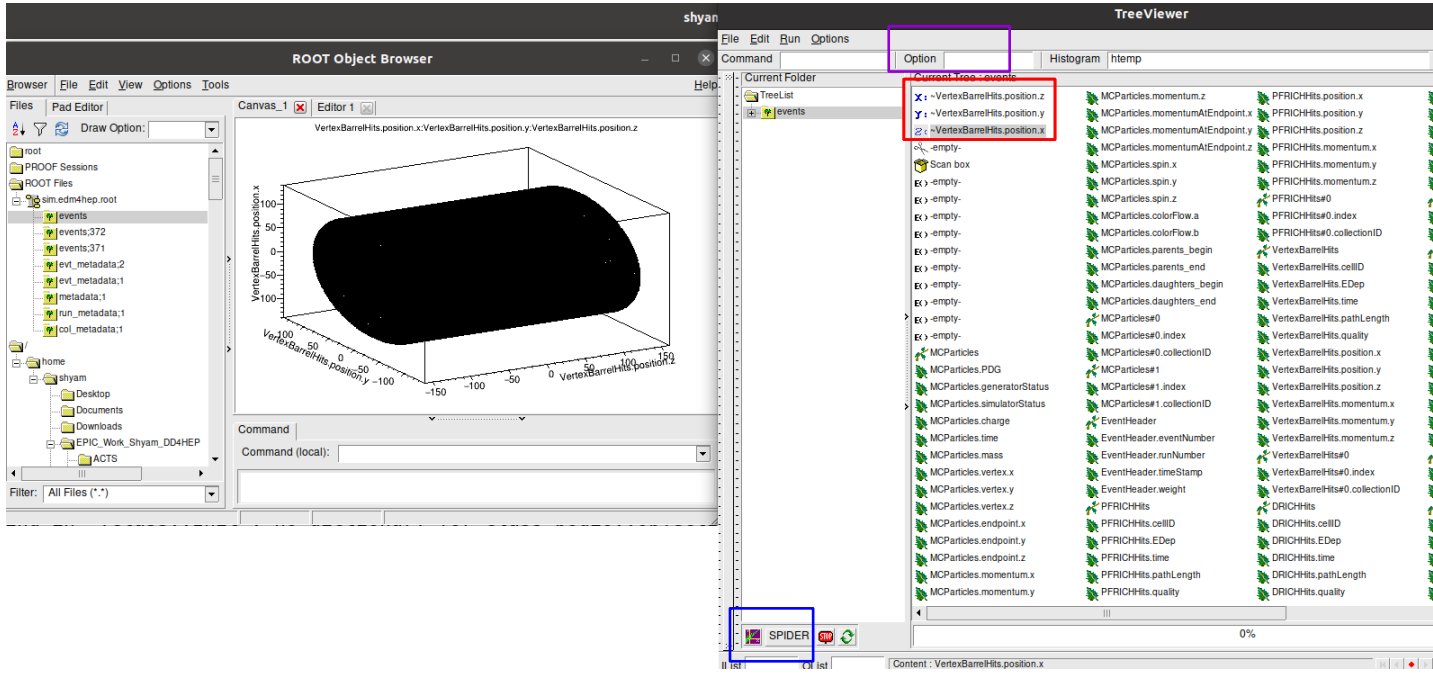
```
 Int nEvents = myReader.GetEntries();
  for (int iEvent =0; iEvent<nEvents; ++iEvent){
  {
   myReader.SetEntry(iEvent);
// True Particle
 for (int imcPart = 0; imcPart < charge.GetSize(); ++imcPart){
   double  pmc = sqrt(px_mc[imcPart]*px_mc[imcPart]
+py_mc[imcPart]*py_mc[imcPart]+pz_mc[imcPart]*pz_mc[imcPart]);
   }
// Reco Particle
 for (int irecPart = 0; irecPart <  px_rec.GetSize(); ++irecPart){
   double  prec = sqrt(px_rec[irecPart]*px_rec[irecPart]
+py_rec[irecPart]*py_rec[irecPart]+pz_rec[irecPart]*pz_rec[irecPart]);
   }
}
```

Drawing Tree using Browser

Select and drag variables

Write histogram draw options e.g. colz





Let's see behind the scene

Hit to draw

shyam@shyam:~/EPIC_Work_Shyam_DD4HEP/EPIC_LeverArm$ cat $HOME/.root_hist

```
TFile *_file0 = TFile::Open("sim.edm4hep.root")
new TBrowser
tv__tree = (TTree *)0x55e752794f60;
```

```
tv__tree->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.z","","", 3000000, 0);
```

Long64_t Draw(const char* varexp, const **TCut**& selection, Option_t* option = "", Long64_t nentries = kMaxEntries, Long64_t firstentry = 0) // General command

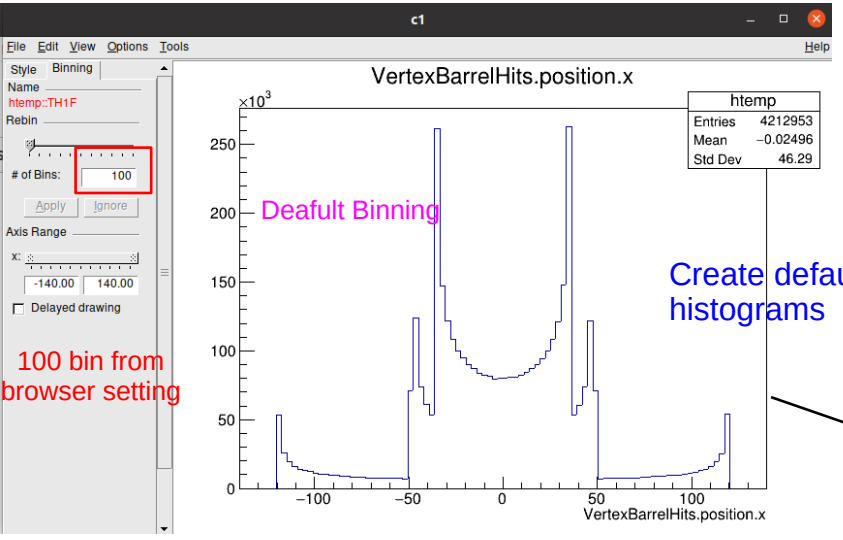
Drawing Tree using Commands

Deafult Binning

root [3] events->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.z") // 3D MC Points
root [2] events->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y") // 2D MC Points
root [3] events->Draw("VertexBarrelHits.position.x") // 1D points

locate system.rootrc

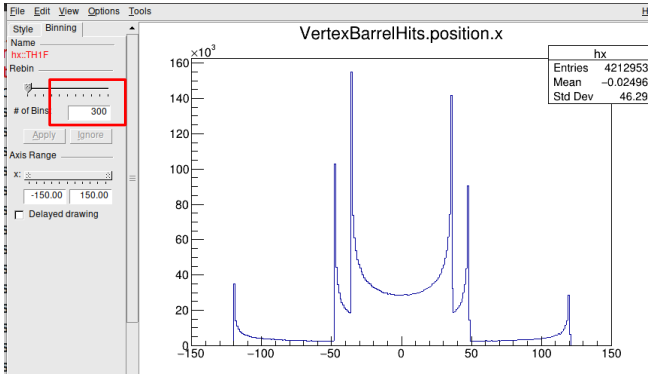
```
# Default histogram binnings for TTree::Draw().
Hist.Binning.1D.x:          100

Hist.Binning.2D.x:           40
Hist.Binning.2D.y:           40
Hist.Binning.2D.Prof:       100

Hist.Binning.3D.x:           20
Hist.Binning.3D.y:           20
Hist.Binning.3D.z:           20
```



100 bin from browser setting

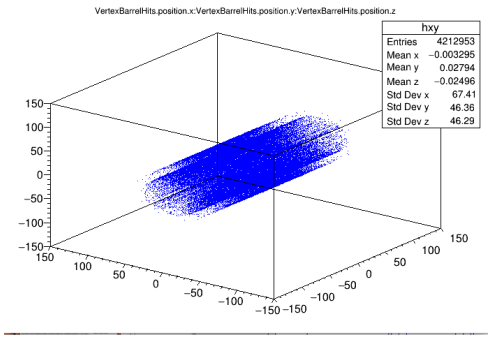
Create default histograms

Different Binning



Different Binning

root [3] events->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.z>>hxy(300,-150.,150.,300,-150.,150.,300,-150.,150.","","")
root [5] hxy->SetMarkerColor(kBlue)
root [6] hxy->Draw()
root [2] events->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y>>hxy(300,-150.,150.,300,-150.,150.","","**colz**") // 2D MC Points
root [3] events->Draw("VertexBarrelHits.position.x>>hx(300,-150.,150.") // 1D points



Check a particular event: 544, same applies for 1D and 2D commands above

root [7] events->Draw("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.z>>hxy(300,-150.,150.,300,-150.,150.,300,-150.,150.","","",1,544)

Scanning Tree using Commands

See full event information:

root [2] events->Show(EventNo)

Sanning event by event:
Row (event number),
Intsance (hit number), etc.

```
root [14] events->Scan("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.quality")
************************************************************
*    Row   * Instance * VertexBar * VertexBar * VertexBar *
************************************************************
*      0 *        0 *          *          *          *
*      1 *        0 *          *          *          *
*      2 *        0 * 15.060219 * 32.700527 *          0 *
*      2 *        1 * 20.149187 * 43.568118 *          0 *
*      3 *        0 *          *          *          *
*      4 *        0 * -24.21204 * -26.64168 *          0 *
*      4 *        1 * -32.32071 * -35.48781 *          0 *
*      5 *        0 * -17.24302 * -31.60338 *          0 *
*      5 *        1 * -23.01074 * -42.12712 *          0 *
*      5 *        2 * -57.83017 * -105.1556 *          0 *
*      6 *        0 *          *          *          *
*      7 *        0 *          *          *          *
```

Scanning is very helpful in debugging !!

Debugging a Particular event

```
root [12] events->Scan("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.quality","","",1,544)
************************************************************
*    Row   * Instance * VertexBar * VertexBar * VertexBar *
************************************************************
*      544 *        0 * 25.375404 * 25.536283 *          0 *
*      544 *        1 * 33.869543 * 34.012707 *          0 *
*      544 *        2 * 85.215595 * 84.490031 *          0 *
*      544 *        3 * 25.380613 * 25.540895 * 1.073e+09 *
*      544 *        4 * 27.946634 * 22.694060 * 1.073e+09 *
*      544 *        5 * 25.403257 * 25.508431 * 1.073e+09 *
*      544 *        6 * 27.828384 * 22.838148 * 1.073e+09 *
*      544 *        7 * 25.596884 * 25.314804 * 1.073e+09 *
*      544 *        8 * 27.590694 * 23.127773 * 1.073e+09 *
*      544 *        9 * 25.371865 * 25.539822 * 1.073e+09 *
*      544 *       10 * 27.505789 * 23.231231 * 1.073e+09 *
*      544 *       11 * 25.734042 * 25.177646 * 1.073e+09 *
*      544 *       12 * 26.935729 * 23.887613 * 1.073e+09 *
*      544 *       13 * 25.369291 * 25.542396 * 1.073e+09 *
*      544 *       14 * 26.382858 * 24.497613 * 1.073e+09 *
*      544 *       15 * 25.555143 * 25.356544 * 1.073e+09 *
*      544 *       16 * 27.109404 * 23.695993 * 1.073e+09 *
*      544 *       17 * 26.261448 * 24.631568 * 1.073e+09 *
*      544 *       18 * 27.048914 * 23.762733 * 1.073e+09 *
*      544 *       19 * 26.497876 * 24.340928 * 1.073e+09 *
*      544 *       20 * 25.362540 * 25.520863 * 1.073e+09 *
*      544 *       21 * 25.039325 * 25.857991 * 1.073e+09 *
************************************************************
(long long) 22
root [13]
```

Debugging a Particular event (Primary hits)

```
root [13] events-
>Scan("VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.quality","VertexBarrelHits.position.quality==0","")
************************************************************
*    Row   * Instance * VertexBar * VertexBar * VertexBar *
************************************************************
*      544 *        0 * 25.375404 * 25.536283 *          0 *
*      544 *        1 * 33.869543 * 34.012707 *          0 *
*      544 *        2 * 85.215595 * 84.490031 *          0 *
************************************************************
==> 3 selected entries
(long long) 3
root [14]
```

Reading an Output Tree

// using TTree::Draw() in root

Code Based on logic explained can be used but slow w.r.t. TTreeReaderArray but doesn't care about int, float, double and also framework independent

```
void Read_Tree(){

    TFile *f = TFile::Open("sim.edm4hep.root");
    TTree *t = (TTree*) f->Get("events");
    Int_t nEvents = t->GetEntries();
    for (Int_t iEvent =0; iEvent<nEvents; ++iEvent){
    // Read Tracks array this way
    Int_t  nTracks = t->Draw("MCParticles.PDG:MCParticles.generatorStatus:MCParticles.simulatorStatus:MCParticles.charge","","goff",1,iEvent);
    cout<<"NTracks ="<<nTracks<<endl;

    Double_t *pdg = t->GetVal(0); Double_t *genstatus = t->GetVal(1); Double_t *simstatus = t->GetVal(2); Double_t *charge = t->GetVal(3);

    for (int iTrack =0; iTrack <nTracks; ++iTrack){
    cout<<"Track Pdg: "<<pdg[iTrack]<<" Charge: "<<charge[iTrack]<<endl;
    }

  // VTX Layer Hits
    Int_t  nhitsVTX = t-
>Draw("VertexBarrelHits.quality:VertexBarrelHits.position.x:VertexBarrelHits.position.y:VertexBarrelHits.position.z","VertexBarrelHits.quality==0","goff",1,iEvent);
    Double_t *quality_Vtx = t->GetVal(0); Double_t *xhit_Vtx = t->GetVal(1); Double_t *yhit_Vtx = t->GetVal(2); Double_t *zhit_Vtx = t->GetVal(3);

    for (int ihit =0; ihit <nhitsVTX; ++ihit){
    cout<<"VTX: X: "<<xhit_Vtx[ihit]<<" Y: "<<yhit_Vtx[ihit]<<" Z: "<<zhit_Vtx[ihit]<<" Quality: "<<quality_Vtx[ihit]<<endl;
    }

// Barrel Layer Hits
    Int_t  nhitsBarr = t->Draw("SiBarrelHits.quality:SiBarrelHits.position.x:SiBarrelHits.position.y:SiBarrelHits.position.z","SiBarrelHits.quality==0","goff",1,iEvent);

    Double_t *quality_Barr = t->GetVal(0);  Double_t *xhit_Barr = t->GetVal(1); Double_t *yhit_Barr = t->GetVal(2); Double_t *zhit_Barr = t->GetVal(3);

    for (int ihit =0; ihit <nhitsBarr; ++ihit){
    cout<<" Barrel: X: "<<xhit_Barr[ihit]<<" Y: "<<yhit_Barr[ihit]<<" Z: "<<zhit_Barr[ihit]<<" Quality: "<<quality_Barr[ihit]<<endl;
    }
    } // Events Loop
}
```

Flag to select primary hits

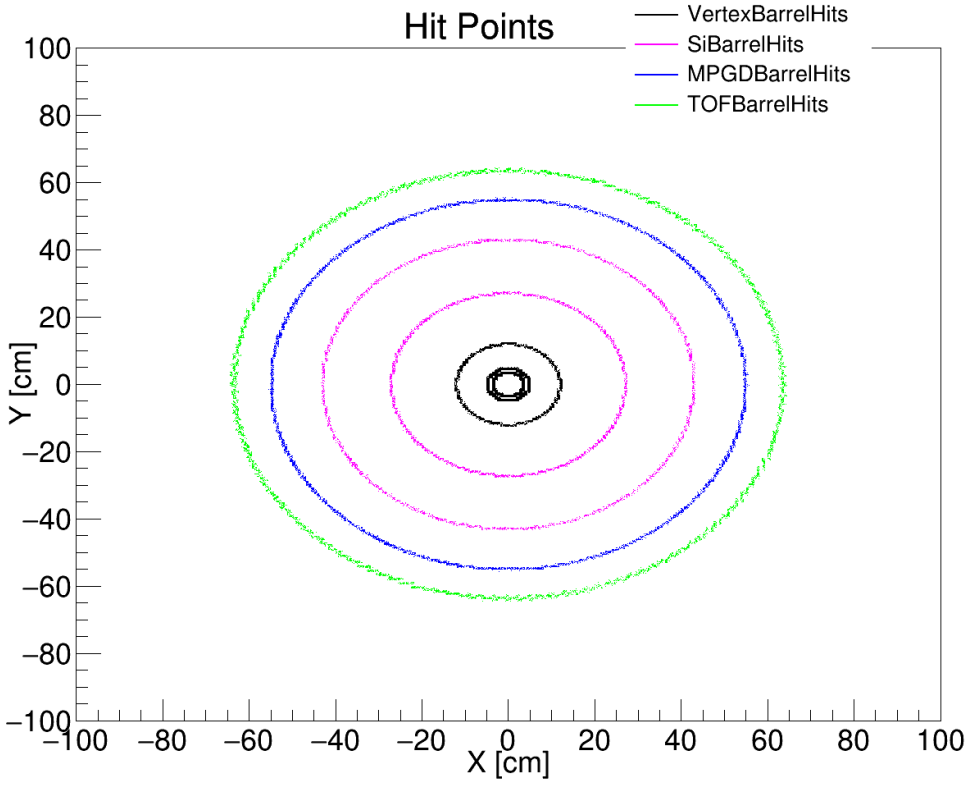
I also attached the other working code based on this tutorial and can be used quickly

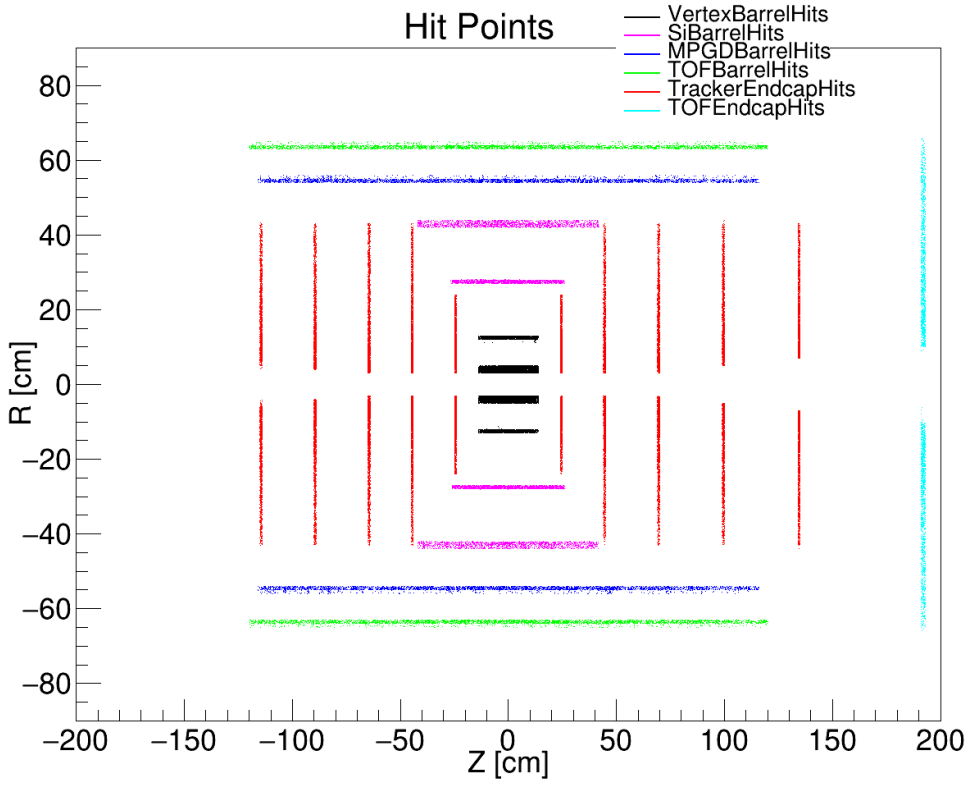
Input simulation output root file

Output of Example Code

root -l draw_hits.C

Output of the code (Hit Map (X,Y) and (R,Z)



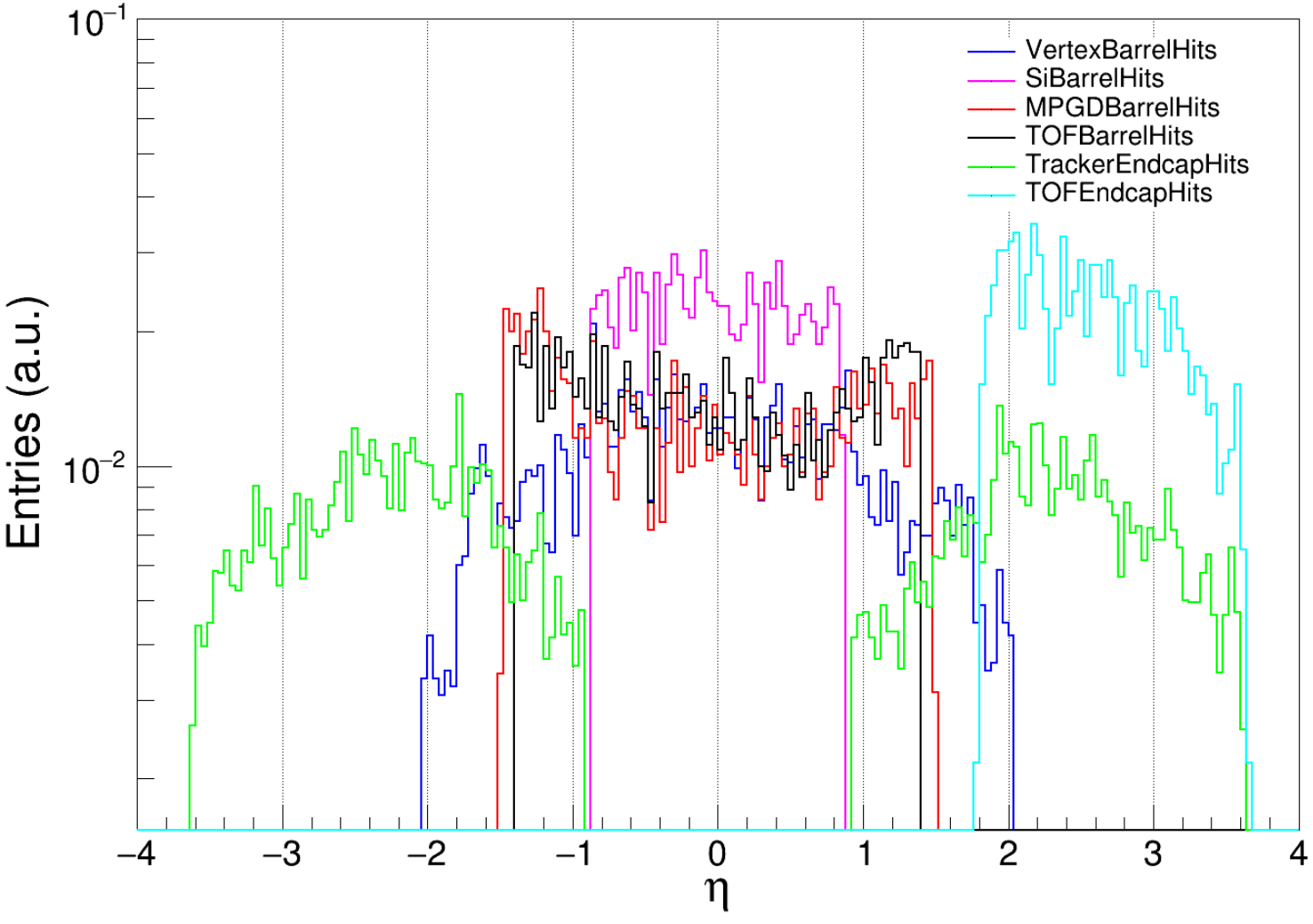


Input simulation output root file

Output of Example Code

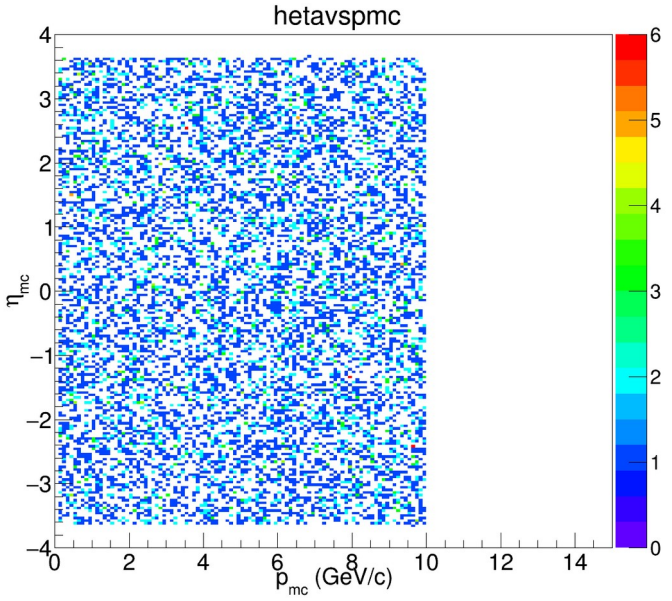
root -l Plot_eta.C

Output of the code (Eta coverage of each detectors)



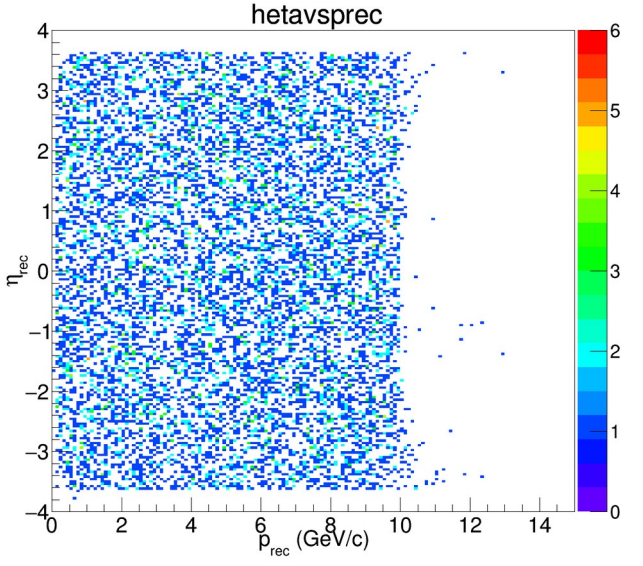
Input: EIC Recon: Output file

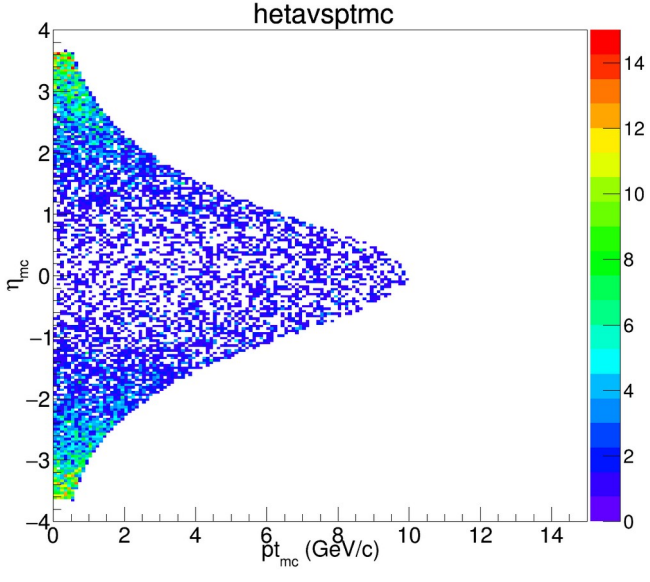
Output of Example Code

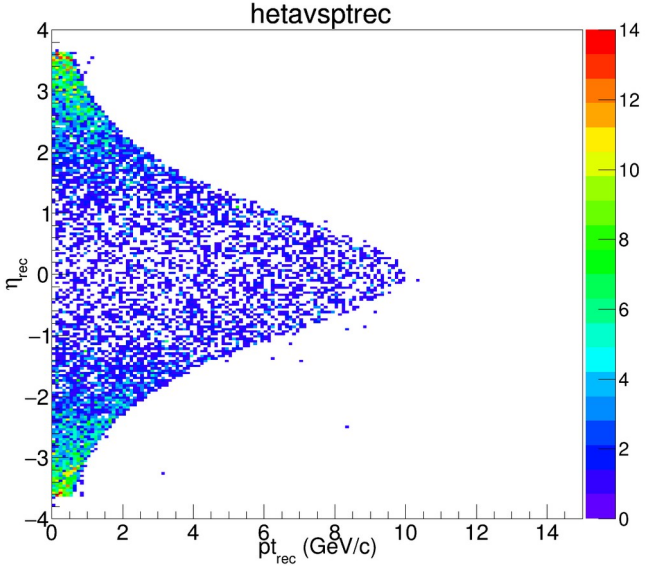


10k pi+ uniform in η [-3.5,3.5] and momentum

root -l draw_Performance.C





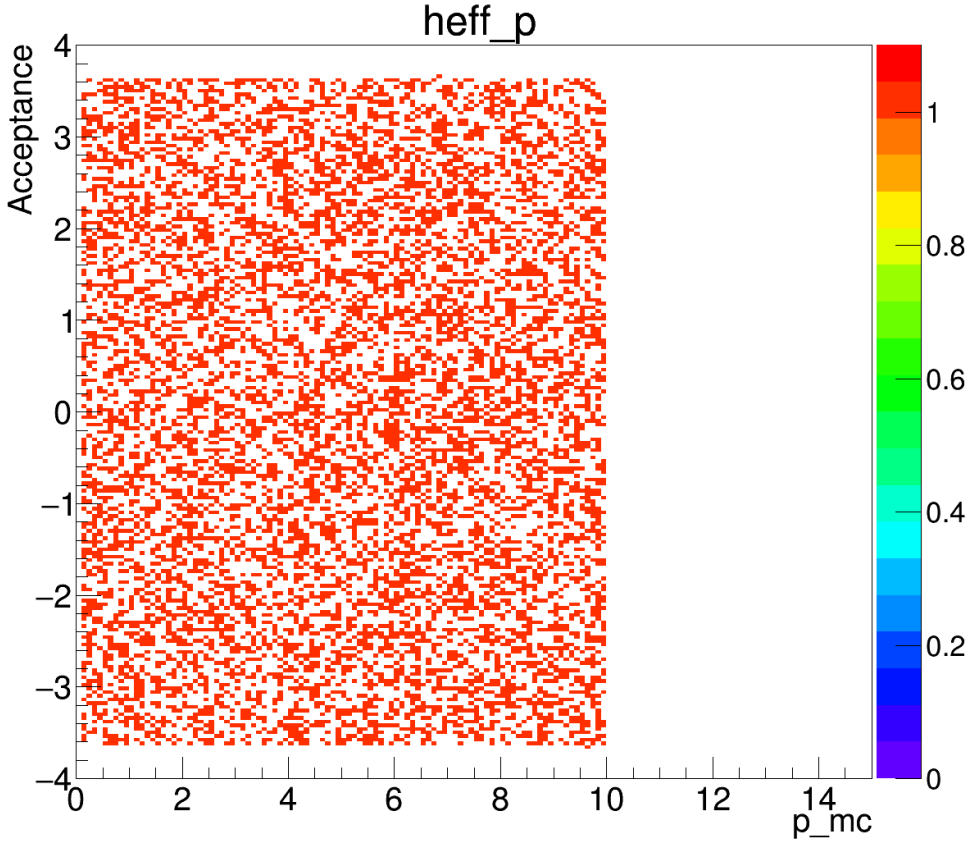


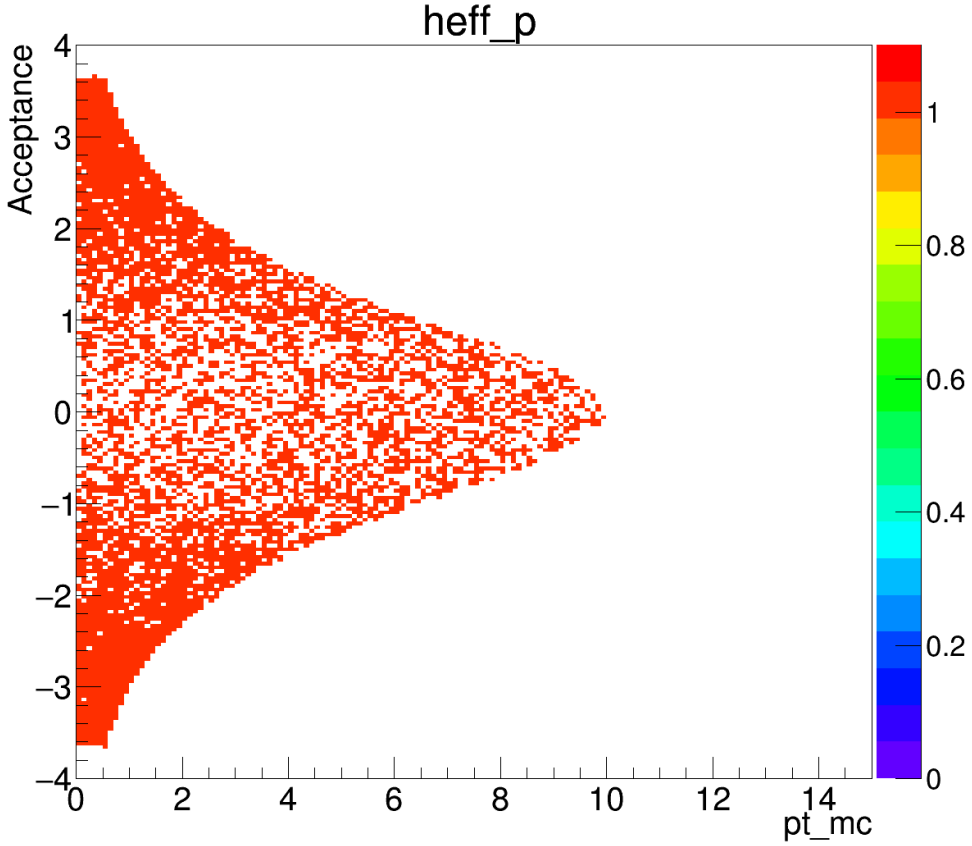
Input: EIC Recon: Output file

Output of Example Code

root -l draw_Performance.C

Single particle (pi+) per event





Note: I need to change logic for larger multiplicity