Update on D0 Fragmentation Function

Gurtaj Singh

Supervisor: Prof. Lokesh Kumar
In Collaboration with Shyam Kumar and Rongrong Ma

Department of Physics, Panjab University Chandigarh
March 10, 2026

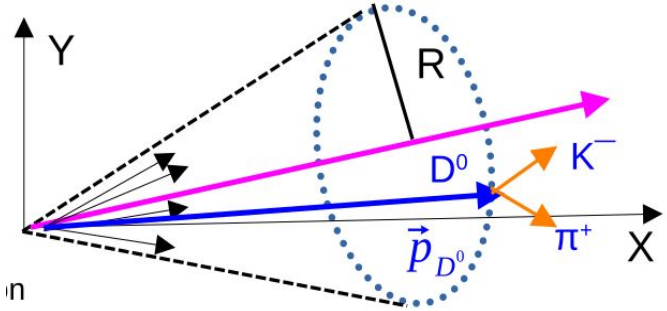
Physics Motivation:

- **Fragmentation Variable(Z):** It describes the hadronization of quarks
- In our case, we are looking for the Fragmentation variable for the D0 meson,

Z = (Jet Momentum.D0 Momentum) / (Jet Momentum.Jet Momentum)

(It should always be less than 1.0)

- This study will help us to understand the D0 hadronization in e+p collisions and possible modifications in e + Au collisions



Jet Radius (R) = 1.0

➔ Data Sample:

- ep 10 X 100 GeV **314290** Events
- Reconstruction Version 25.10.3
- SIDIS ep :
  - PYTHIA 8.306
    - Q2 min = 1 GeV^2

➔ Algorithm for Jet Reading:

1. Loop over RC Charged Jets.

2. Find the jet momentum vector and using this find the jet eta and phi.

   Skip **loop** iteration for jets that have only 1 constituent RC charged particle. Also skip **loop** iteration for jets that have pseudo-rapidity greater than 2.5 or less than -2.

3. Create a **for loop**, which runs each time for a unique RC daughter pion and kaon corresponding to a D0 parent:
   a. Find the daughter pion momentum vector and using this find the corresponding eta and phi values.
   b. Similarly, find the daughter kaon momentum vector and using this find the corresponding eta and phi values.
   c. Construct Lorentz vector corresponding to parent D0 , and using this find the corresponding eta and phi values.

   Skip **for loop** iteration for pion and kaon that have pseudo-rapidity greater than 2.5 or less than -2. Skip **for loop** iteration for pion and kaon that have RC index equal to **REMOVED**.(**REMOVED** is a very large positive integer, that no RC charged particle can have as RC index)

d. Calculate the ∆R for each of the daughter pion, daughter kaon and parent D0;
$\Delta R = \sqrt{[(\Delta\eta)^2 + (\Delta\varphi)^2]}$, where **$\Delta\eta = \eta\_jet - \eta\_particle$** and **$\Delta\varphi = \varphi\_jet - \varphi\_particle$**

e. If ∆R for all of three particles D0 parent, daughter pion and daughter kaon is less than 1 **(as Jet Radius = 1.0)**, then calculate **Fragmentation Variable(Z)**:

Z = (Jet_MomentumVector).(D0_MomentumVector) / |Jet_MomentumVector|^2

and fill the second bin of *hJetStat* histogram.

Set the RC index of daughter pion and kaon equal to **REMOVED** in **rc_index_D0_pii** and **rc_index_D0_kk** array respectively.

f. If ∆R is less than 1 for D0 parent:

- Fill the first bin of *hJetStat* histogram,

- If ∆R is less than 1 for both pion and kaon daughter:
  - Fill the third bin of *hJetstat* histogram.
  - Set the RC index of daughter pion and kaon equal to **REMOVED** in **rc_index_D0_pii** and **rc_index_D0_kk** array respectively.

- If ∆R is less than 1 for pion only or if ∆R is less than 1 for kaon only:
  - Fill the fourth bin of *hJetstat* histogram.
  - Set the RC index of daughter pion and kaon equal to **REMOVED** in **rc_index_D0_pii** and **rc_index_D0_kk** array respectively.

Repeat the step 3 for all the RC pi-k daughter pairs.

Repeat the step 2 and 3 for all the RC charged Jets.

CODE:

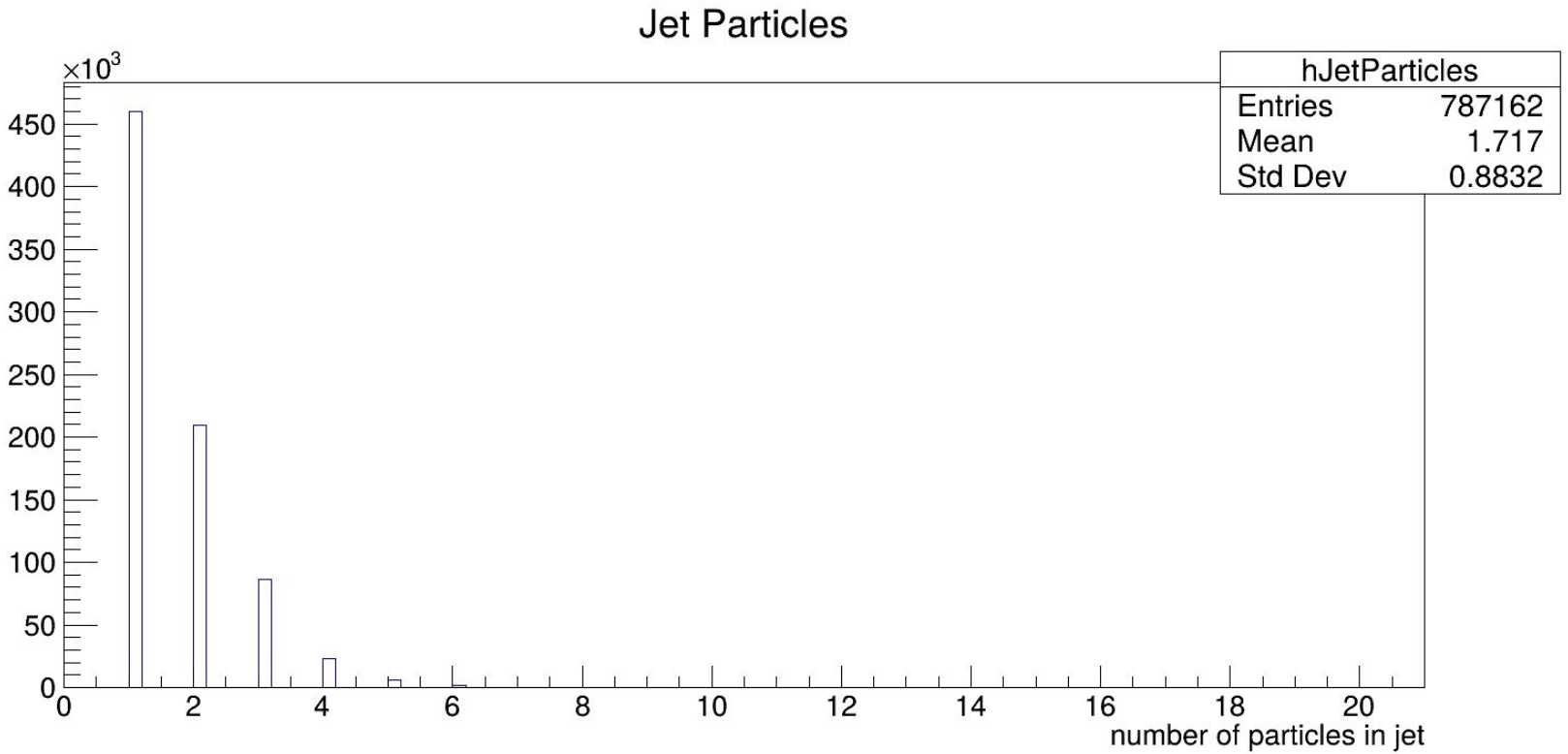
```
//new:
// Reading Jet:
for(unsigned int i=0; i<recoType.GetSize(); i++) // Loop over RC Jets
 {

   int x = partsBegin[i];     //Calculate No. of Jet Particles
   int y = partsEnd[i];
   int NumberOf_JetParticles = y-x ;

  hJetParticles->Fill(NumberOf_JetParticles);
  if(NumberOf_JetParticles < 2 ) continue;    // skip the jets with only 1 constituent

  TVector3 jetMom(recoMomX[i],recoMomY[i],recoMomZ[i]);     // Jet Kinematics
  double eta_Jet = jetMom.Eta();
  double phi_Jet = jetMom.Phi();
  hJetpT->Fill(jetMom.Pt());

 if(eta_Jet >2.5 || eta_Jet < -2) continue;  //eta cut on jet

 for(unsigned int j=0; j<rc_index_D0_pii.size(); j++ ){    // Loop over all RC Unlike Charged pi-k daughters of D0

   if( rc_index_D0_pii[j] == REMOVED || rc_index_D0_kk[j] == REMOVED) continue;      // REMOVED == 96844441

 int index_pii = rc_index_D0_pii[j];
 int index_kk =  rc_index_D0_kk[j];

 TVector3 piiMom(rcMomPx[index_pii],rcMomPy[index_pii],rcMomPz[index_pii]); // pi-k Daughters Kinematics
 TVector3 kkMom(rcMomPx[index_kk],rcMomPy[index_kk],rcMomPz[index_kk]);

 double eta_pii = piiMom.Eta();
 double phi_pii = piiMom.Phi();

 double eta_kk = kkMom.Eta();
 double phi_kk = kkMom.Phi();

```

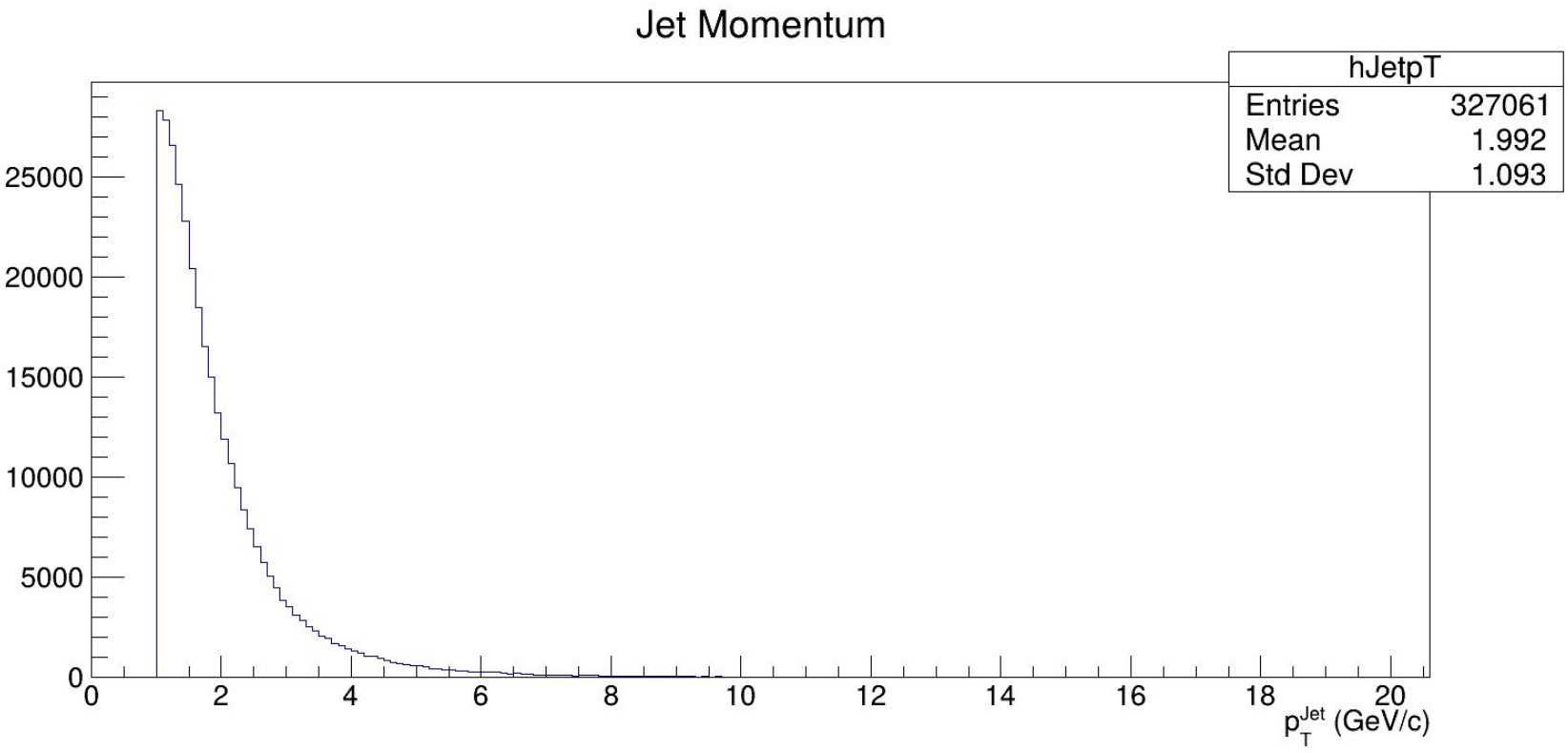
```
669
670 if(eta_pii>2.5 || eta_pii <-2 || eta_kk>2.5 || eta_kk <-2) continue;     // eta cut on pi-k
671
672
673 double deltaEta_pii = eta_Jet - eta_pii;
674 double deltaPhi_pii = phi_Jet - phi_pii;
675 double DeltaR_pii = sqrt(deltaEta_pii * deltaEta_pii + deltaPhi_pii * deltaPhi_pii); // DeltaR for pi
676
677 double deltaEta_kk = eta_Jet - eta_kk;
678 double deltaPhi_kk = phi_Jet - phi_kk;
679 double DeltaR_kk = sqrt(deltaEta_kk * deltaEta_kk + deltaPhi_kk * deltaPhi_kk); // DeltaR for k
680
681 float dcaDaughters, cosTheta, decayLength, V0DcaToVtx;        // parent D0 of pi-k daughters
682 TLorentzVector parent = getPairParent(index_pii,index_kk, vertex_rc, dcaDaughters, cosTheta, decayLength, V0DcaToVtx); // Parent D0 Four Vector
683
684     double eta_parent = parent.Eta();   //D0 Parent Kinematics
685     double phi_parent = parent.Phi();
686
687     double deltaEta_parent = eta_Jet - eta_parent;
688     double deltaPhi_parent = phi_Jet - phi_parent;
689     double DeltaR_parent = sqrt(deltaEta_parent * deltaEta_parent + deltaPhi_parent * deltaPhi_parent);     // DeltaR D0parent&Jet
690
691     TVector3 parentMom = parent.Vect();  //Parent Momentum vector
692     double D0_momMag = parentMom.Mag();
693
694     double _3dAngle_pik = piiMom.Angle(kkMom);     // Calculating 3DAngle
695     double _3dAngle_D0Jet = parentMom.Angle(jetMom);
696
697     double deltaEta_PiK = eta_pii - eta_kk;     double deltaPhi_PiK = phi_pii - phi_kk;    //DeltaR_Pi&K
698     double DeltaR_PiK = sqrt(deltaEta_PiK * deltaEta_PiK + deltaPhi_PiK * deltaPhi_PiK);
699
```

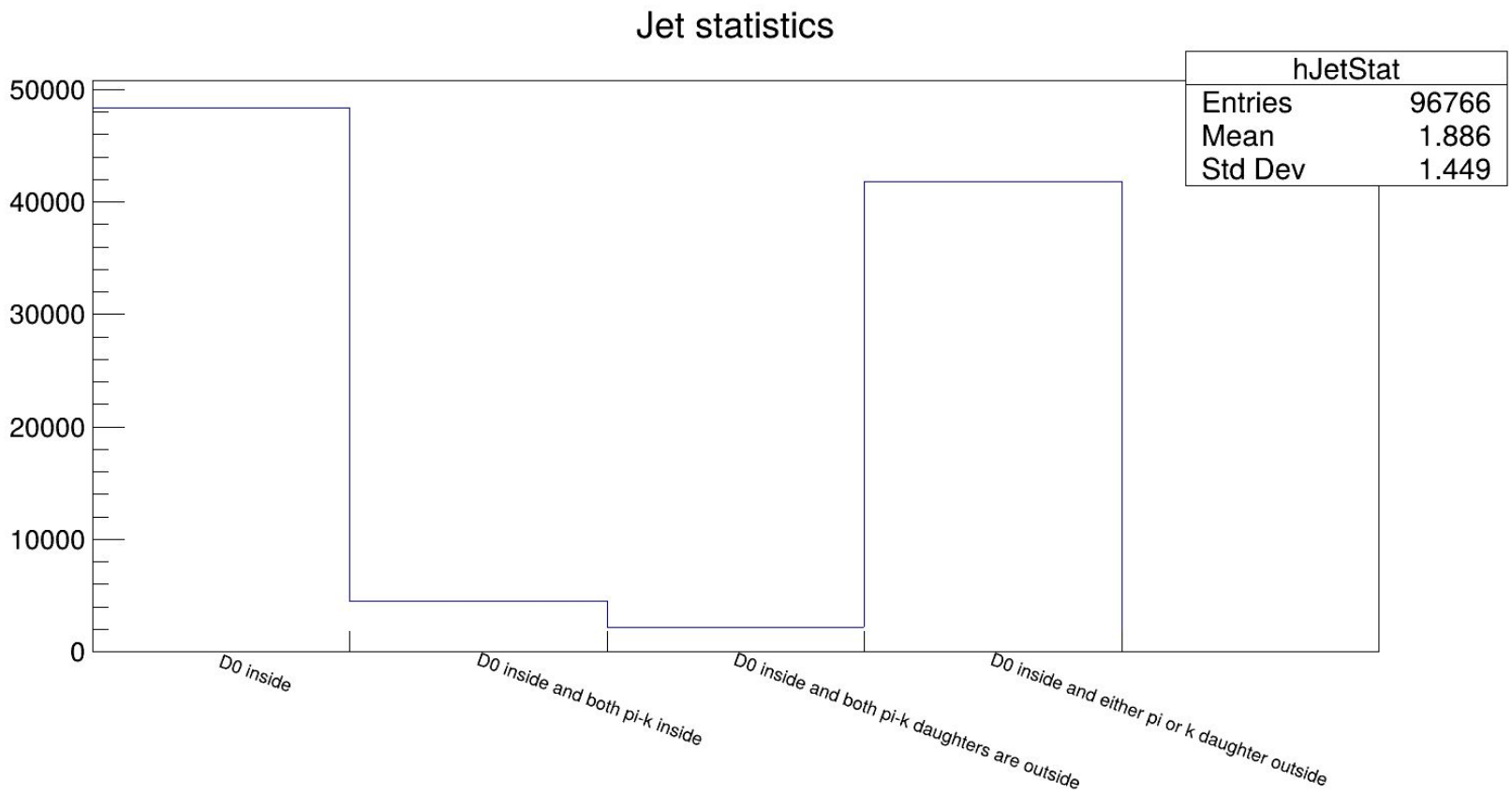
```
699
700
701     if(DeltaR_pii < 1.0 && DeltaR_kk <1.0 && DeltaR_parent <1.0 ){  // Calculate Fragmentaion Variable
702
703     double z = double(jetMom.Dot(parentMom))/double(jetMom.Dot(jetMom)) ;
704
705     hFragmentation->Fill(z);
706     hJetStat->Fill(1.5);
707     rc_index_D0_pii[j] = REMOVED; // Setting the index of used pi-k equal to "REMOVED", in order to avoid multiple counting
708     rc_index_D0_kk[j] = REMOVED;
709
710     hD0MomVsDeltaR_PiK_BothInside->Fill(D0_momMag,DeltaR_PiK);      //Delta_R
711     hD0MomVsDeltaR_D0Jet_BothInside->Fill(D0_momMag,DeltaR_parent);
712
713     hD0MomVs3dAngle_pik_BothInside->Fill(D0_momMag,_3dAngle_pik);    //3DAngle
714     hD0MomVs3dAngle_D0Jet_BothInside->Fill(D0_momMag,_3dAngle_D0Jet);
715     }
716
```

```
      if(DeltaR_parent<1.0){    // If D0 Parent Inside the Jet
      hJetStat->Fill(0.5);

      if(DeltaR_pii>1.0 && DeltaR_kk >1.0){  // D0 parent inside the jet, and both  pi-k doughters are outside the jet
      hJetStat->Fill(2.5);
      rc_index_D0_pii[j] = REMOVED;
      rc_index_D0_kk[j] = REMOVED;

      hD0MomVsDeltaR_PiK_BothOut->Fill(D0_momMag,DeltaR_PiK);       //Delta_R
      hD0MomVsDeltaR_D0Jet_BothOut->Fill(D0_momMag,DeltaR_parent);

      hD0MomVs3dAngle_pik_BothOut->Fill(D0_momMag,_3dAngle_pik);    //3DAngle
      hD0MomVs3dAngle_D0Jet_BothOut->Fill(D0_momMag,_3dAngle_D0Jet);
      }

    if((DeltaR_pii>1.0 && DeltaR_kk <1.0) || (DeltaR_kk>1.0 && DeltaR_pii<1.0)){  // D0 parent inside the jet, but at least one of daughter is
outside
      hJetStat->Fill(3.5);
      rc_index_D0_pii[j] = REMOVED;
      rc_index_D0_kk[j] = REMOVED;

      hD0MomVsDeltaR_PiK_EitherOut->Fill(D0_momMag,DeltaR_PiK);     //Delta_R
      hD0MomVsDeltaR_D0Jet_EitherOut->Fill(D0_momMag,DeltaR_parent);

      hD0MomVs3dAngle_pik_EitherOut->Fill(D0_momMag,_3dAngle_pik);     //3DAngle
      hD0MomVs3dAngle_D0Jet_EitherOut->Fill(D0_momMag,_3dAngle_D0Jet);
      }
      hD0MomVsDeltaR_D0Jet_All->Fill(D0_momMag,DeltaR_parent);  //Delta_R
      hD0MomVs3dAngle_D0Jet_All->Fill(D0_momMag,_3dAngle_D0Jet); //3DAngle
     }
    }
  }
    cout<<nevents<<"event ends"<<endl;
    nevents++;
   }
```

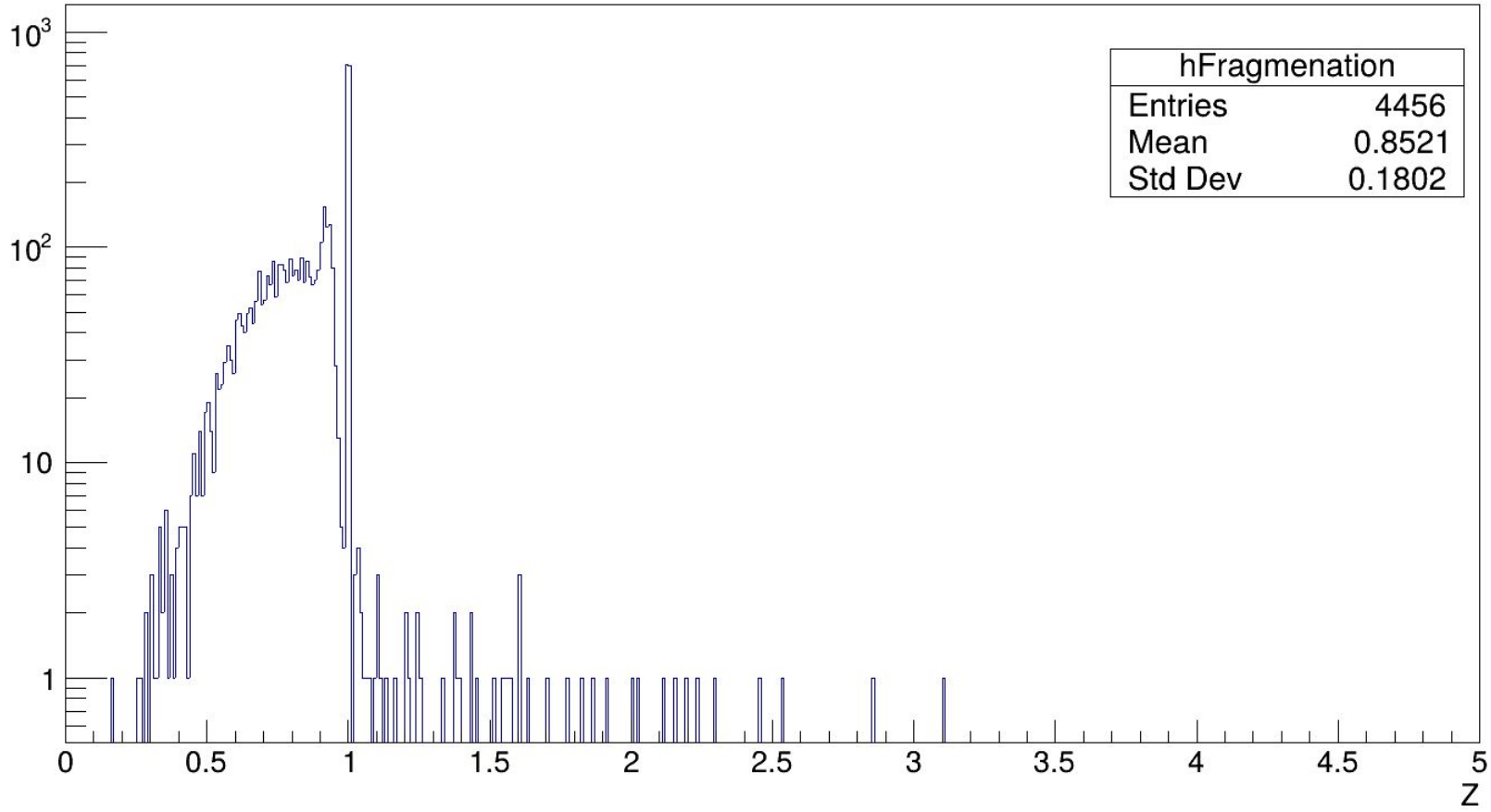
Results:


Jet Particles
hJetParticles
Entries 787162
Mean 1.717
Std Dev 0.8832
×10^3
450
400
350
300
250
200
150
100
50
0
0
2
4
6
8
10
12
14
16
18
20
number of particles in jet

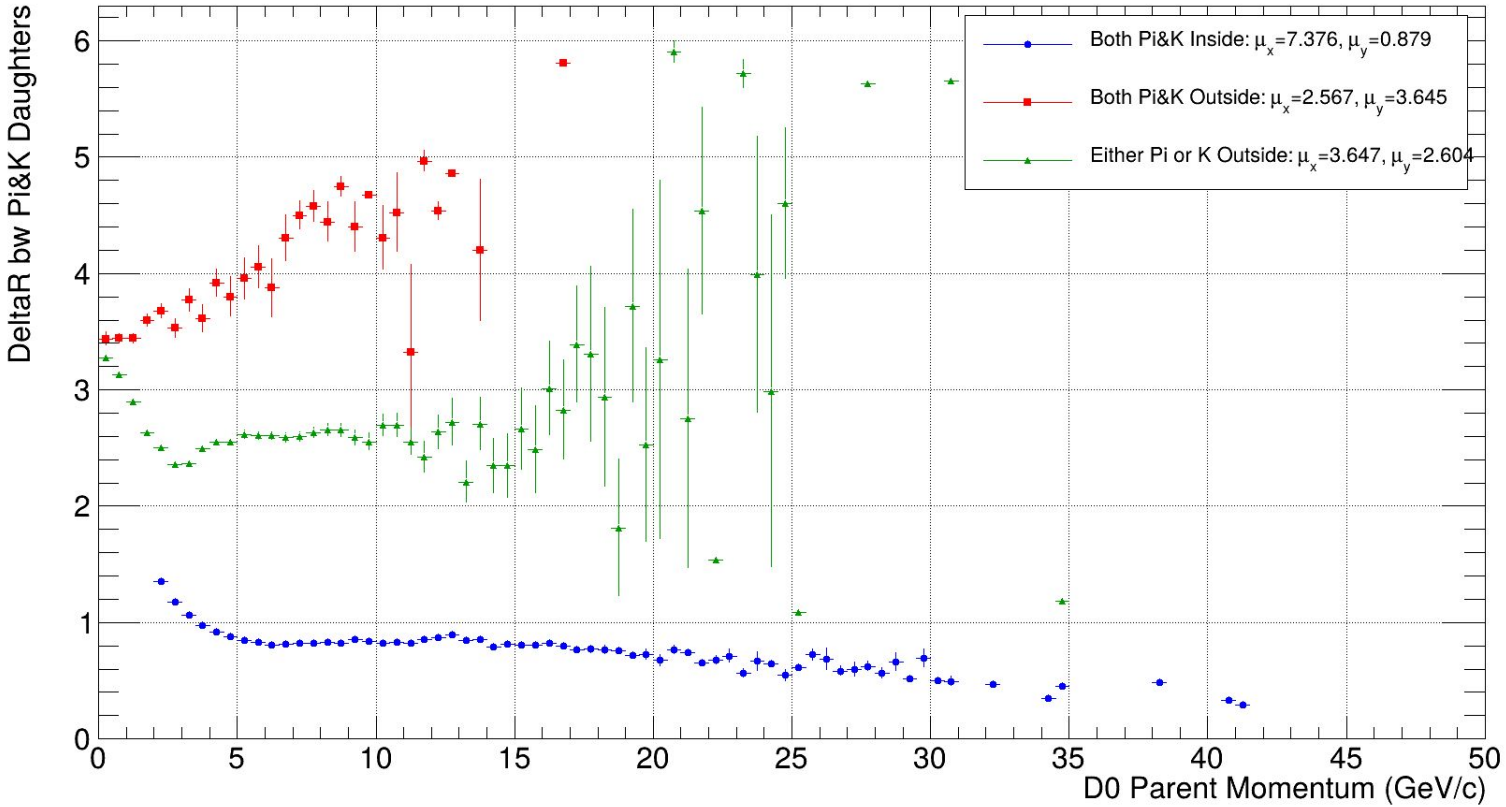

Jet Momentum
hJetpT
Entries 327061
Mean 1.992
Std Dev 1.093
25000
20000
15000
10000
5000
0
0
2
4
6
8
10
12
14
16
18
20
p_T^{Jet} (GeV/c)


Jet statistics
hJetStat
Entries 96766
Mean 1.886
Std Dev 1.449
50000
40000
30000
20000
10000
0
D0 inside
D0 inside and both pi-k inside
D0 inside and both pi-k daughters are outside
D0 inside and either pi or k daughter outside

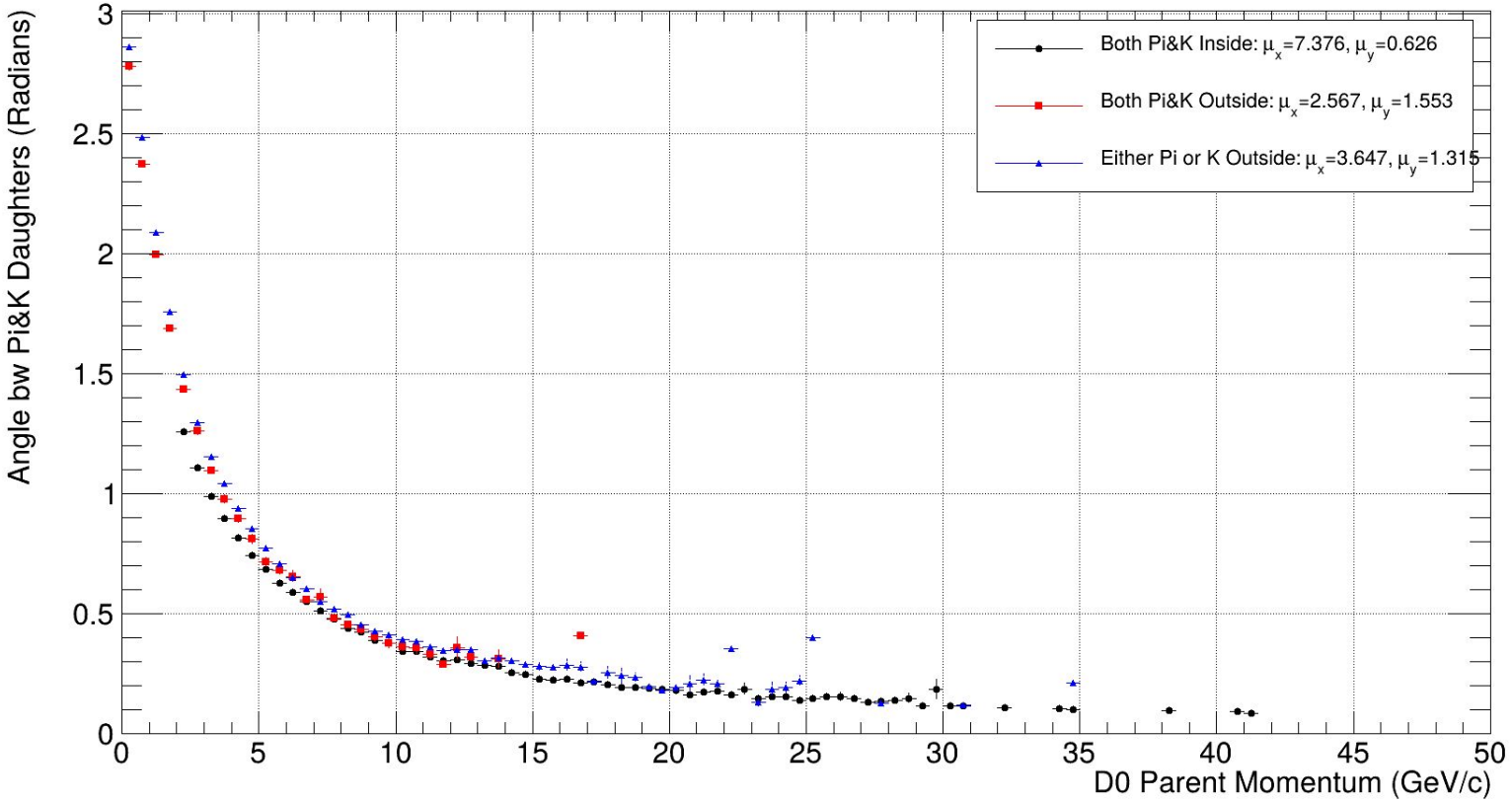
Fragmentation Variable (Z) :

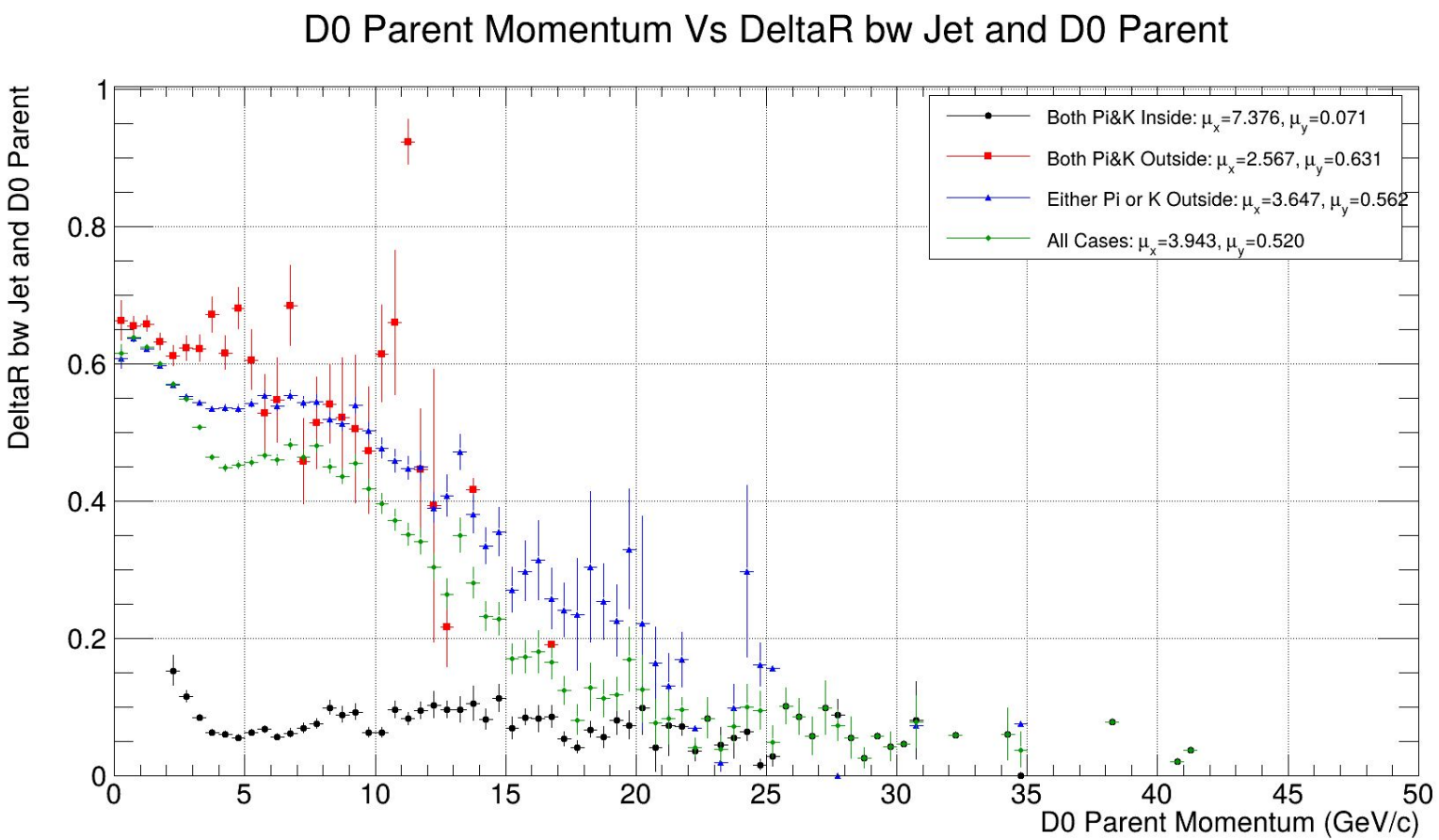


D0 Parent Momentum Vs DeltaR bw Pi&K Daughters

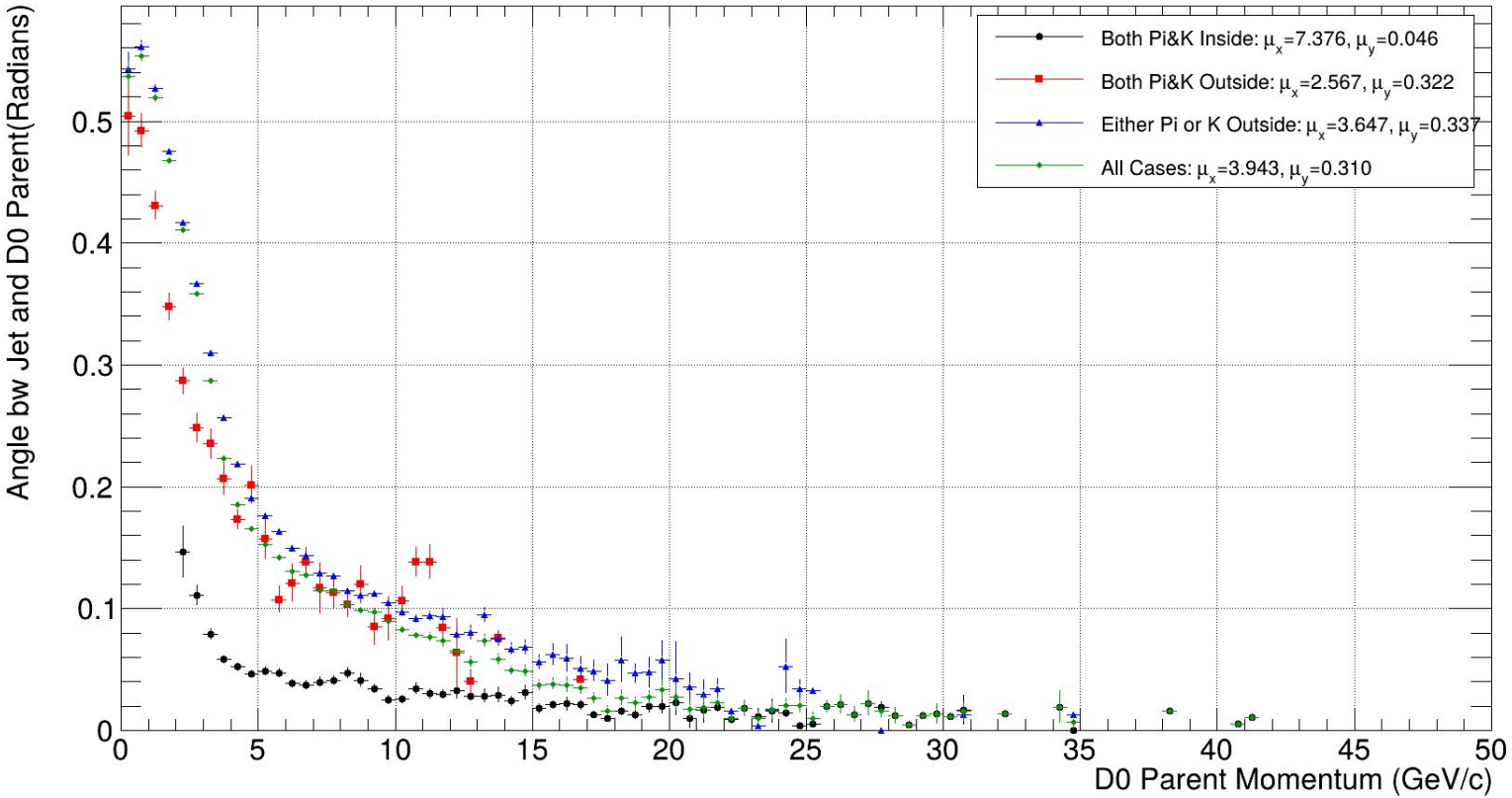


D0 Parent Momentum Vs Angle bw Pi&K Daughters




D0 Parent Momentum Vs DeltaR bw Jet and D0 Parent
DeltaR bw Jet and D0 Parent
D0 Parent Momentum (GeV/c)
Both Pi&K Inside: μx=7.376, μy=0.071
Both Pi&K Outside: μx=2.567, μy=0.631
Either Pi or K Outside: μx=3.647, μy=0.562
All Cases: μx=3.943, μy=0.520
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25
30
35
40
45
50

D0 Parent Momentum Vs Angle bw Jet and D0 Parent



Conclusion:

- For jet radius R=0.8, the fragmentation variable (z) values for all entries are less than 1.0.

- We need to do Jet Reconstruction with some smaller Jet Radius value??