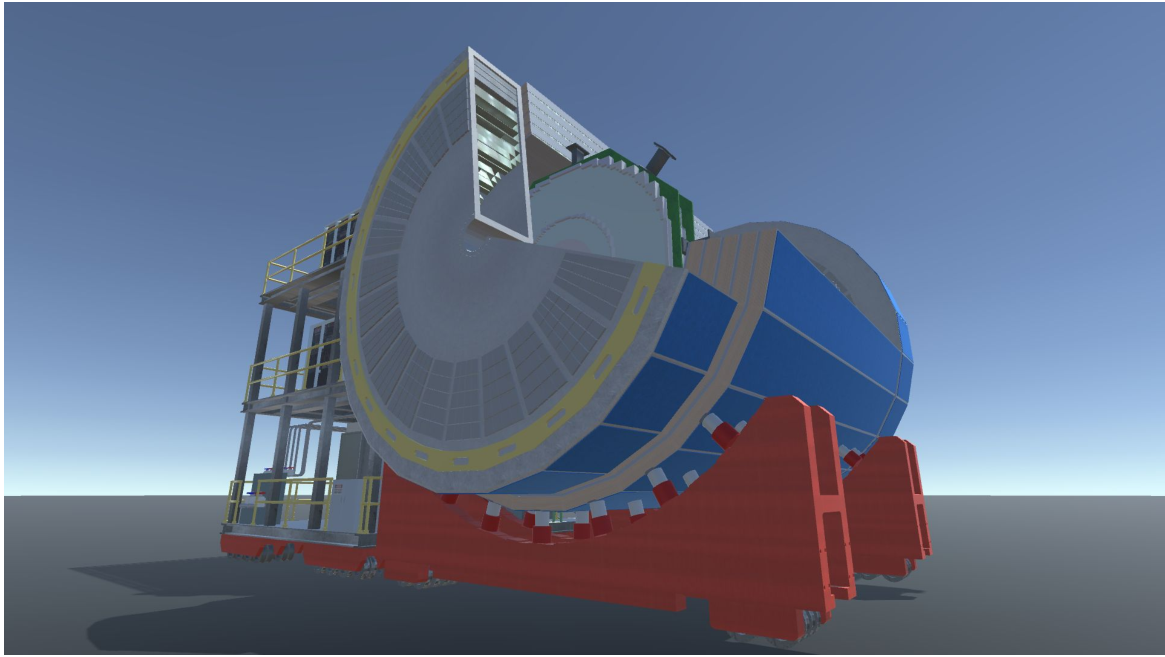
Jet studies with Full ATHENA sims



Miguel Arratia (UCR),
June 22th 2021
(T-162 days)

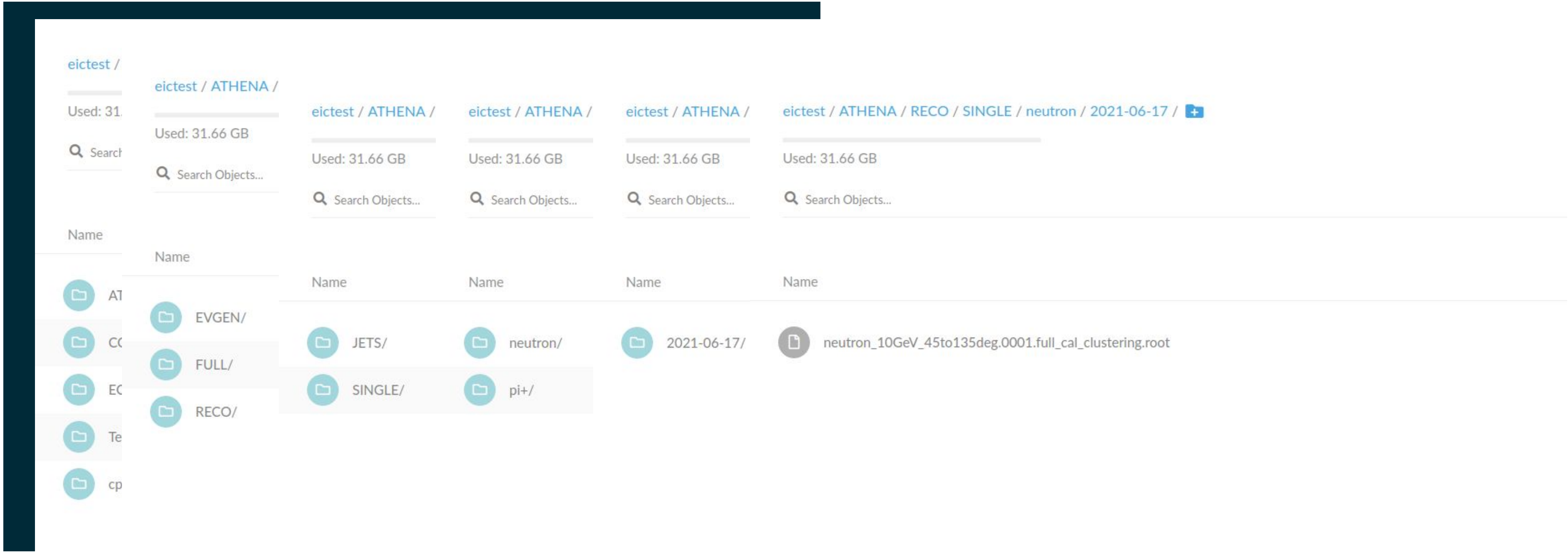
An example of
using DD4hep for Jet Benchmark Studies

Step #1: Get simulation samples

Slides from Wouter

Accessing Large Data Productions: From the Web (username: eicS3read , eicS3read pass)

https://dtn01.sdcc.bnl.gov:9000/minio/login



Slides from Wouter

Accessing Large Data Productions: Command Line

Download the Minio client:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

Register your S3 instance:

```
./mc config host add S3 https://dtn01.sdcc.bnl.gov:9000 $u $p
```

Copy files (recursively):

```
./mc cp -r S3/eictest/ATHENA/RECO/SINGLE/neutron/2021-06-17 .
```

Full docs: http://doc.athena-eic.org/en/latest/howto/s3_file_storage.html

Step #2: Analyze

localhost:8888/notebooks/calostudies/jets.ipynb

jupyter jets Last Checkpoint: 18 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:
```
#%matplotlib notebook
import uproot as ur
import matplotlib.pyplot as plt
import k3d
import numpy as np
import awkward as ak
from pyjet import cluster
from pyjet.testdata import get_event
from pyjet import DTYPE_EP
from pyjet import DTYPE_PTEPM
from pyjet import PseudoJet, JetDefinition, ClusterSequence, ClusterSequenceArea
```

Get data, transform ROOT tree into array

In [2]:
```
file = ur.open('rec_.root')
tree = file['events']
ak_arrays = tree.arrays()
```

In [3]:
```
def get_vector(varname='HcalBarrelHitsReco',energy='energy'):
    E = np.array(ak.to_list(ak_arrays["%s.%s"%(varname,energy)]), dtype="O")
    x = np.array(ak.to_list(ak_arrays["%s.position.x"%varname]), dtype="O")
    y = np.array(ak.to_list(ak_arrays["%s.position.y"%varname]), dtype="O")
    z = np.array(ak.to_list(ak_arrays["%s.position.z"%varname]), dtype="O")
    theta = np.array(ak.to_list(ak_arrays["%s.polar.theta"%varname]), dtype="O")
    phi = np.array(ak.to_list(ak_arrays["%s.polar.phi"%varname]), dtype="O")
    #E = E/1000.0
    return E,x, y, z, theta, phi
```

Get clusters

In [5]:
```
for i in ['HcalHadronEndcapClusters','EcalEndcapClusters']:
    E[i], x[i], y[i],z[i], theta[i], phi[i] = get_vector("%s"%i,energy='energy')
```

Initialize fast jet ¶

In [6]:
```
vectors = get_event()
sequence = cluster(vectors, R=1.0, p=-1)
jets = sequence.inclusive_jets()  # list of PseudoJets
```

Loop over events, fill clusters into constituent arrays; run jet clustering

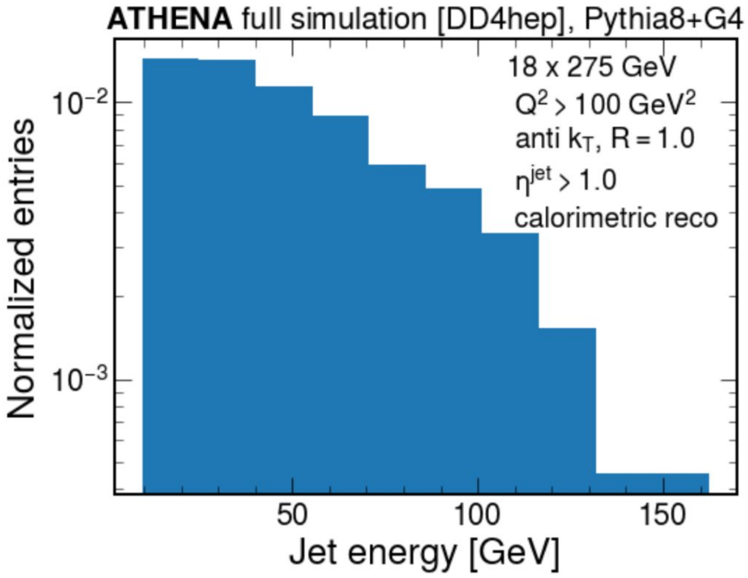
In [8]:
```
jet_E = np.array([])
jet_eta = np.array([])

#loop over events
for ievt in range(1000):
    constituents = np.array([], dtype=DTYPE_PTEPM)#DTYPE_EP)
    #looping over HCAl clusters
    for i in range(len(E['HcalHadronEndcapClusters'][ievt])):
        part_energy = E['HcalHadronEndcapClusters'][ievt][i]/1000.0
        if(part_energy<0.5):
            continue
        part_phi = phi['HcalHadronEndcapClusters'][ievt][i]
        part_theta = theta['HcalHadronEndcapClusters'][ievt][i]
        part_eta = -np.log(np.tan(part_theta/2.0))
        part_pt  = part_energy*np.sin(part_theta)
        print('Energy=%2.2f GeV, phi =%2.2f rad, theta= %2.2f rad, eta=%2.2f, pT = %2.2f GeV'%(part_energy,part_pl
        cluster = np.array([(part_pt, part_eta, part_phi, 0.0)], dtype=DTYPE_PTEPM)
        constituents = np.append(constituents, cluster)
    #looping over ECal clusters
    for i in range(len(E['EcalEndcapClusters'][ievt])):
        part_energy = E['EcalEndcapClusters'][ievt][i]/1000.0
        if(part_energy<0.100):
            continue
        part_phi = phi['EcalEndcapClusters'][ievt][i]
        part_theta = theta['EcalEndcapClusters'][ievt][i]
        part_eta = -np.log(np.tan(part_theta/2.0))
```

```
    part_eta = -np.log(np.tan(part_theta/2.0))
    part_pt  = part_energy*np.sin(part_theta)
    print('Energy=%2.2f GeV, phi =%2.2f rad, theta= %2.2f rad, eta=%2.2f, pT = %2.2f GeV'%(part_energy,part_phi
    cluster = np.array([(part_pt, part_eta, part_phi, 0.0)], dtype=DTYPE_PTEPM)
    constituents = np.append(constituents, cluster)


### Jet reconstruction
jet_def = JetDefinition(algo = 'genkt', R = 1.0,p=-1.0)
cs = ClusterSequence(constituents, jet_def)
jets = cs.inclusive_jets()
#print(jets.pt)
for jet in jets:
    if(jet.pt<5): continue
    print(jet.pt*np.cosh(jet.eta))
    jet_E = np.append(jet.pt*np.cosh(jet.eta),jet_E)
    jet_eta = np.append(jet.eta,jet_eta)
```

Plot jet energy spectrum

```
In [10]: fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot()
ax.hist(jet_E,density=True)

plt.text(0.80, 0.77,'18 x 275 GeV \n $Q^{2}>100~GeV^{2}$ \n anti $k_{T}$, $R=1.0$ \n $\eta^{jet}>1.0$ \n calorimetr
                 horizontalalignment='center',multialignment='left',
                 verticalalignment='center',transform = ax.transAxes, fontsize=20)
plt.title(r"$\bf{ATHENA}$"+' full simulation [DD4hep], Pythia8+G4',fontsize=22)
plt.ylabel('Normalized entries')
plt.xlabel('Jet energy [GeV]')
plt.yscale('log')
plt.show()
##
```

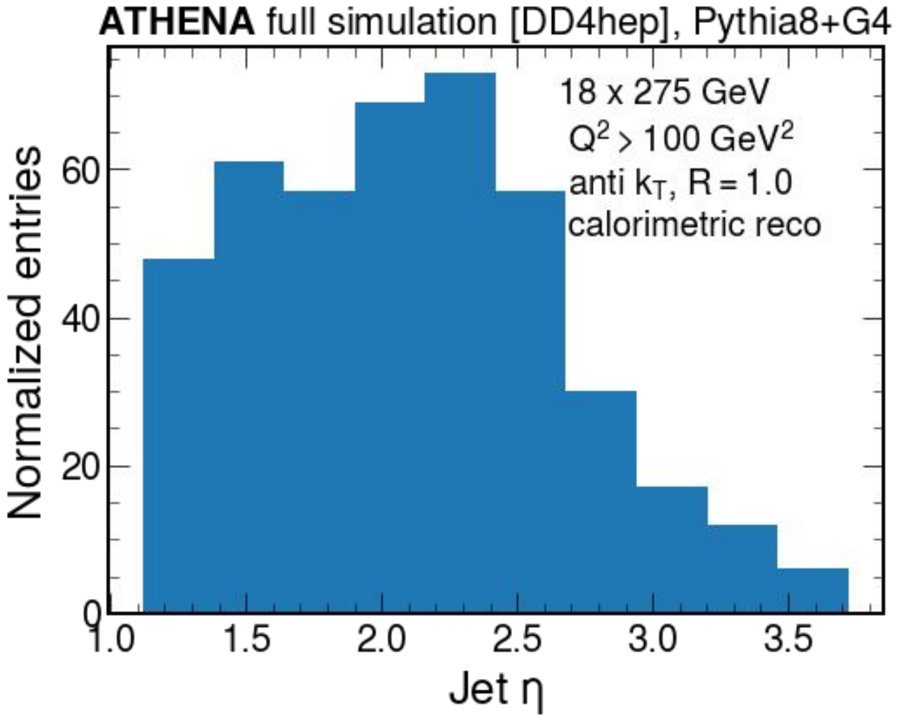
```
findfont: Font family ['cursive'] not found. Falling back to DejaVu Sans.
```

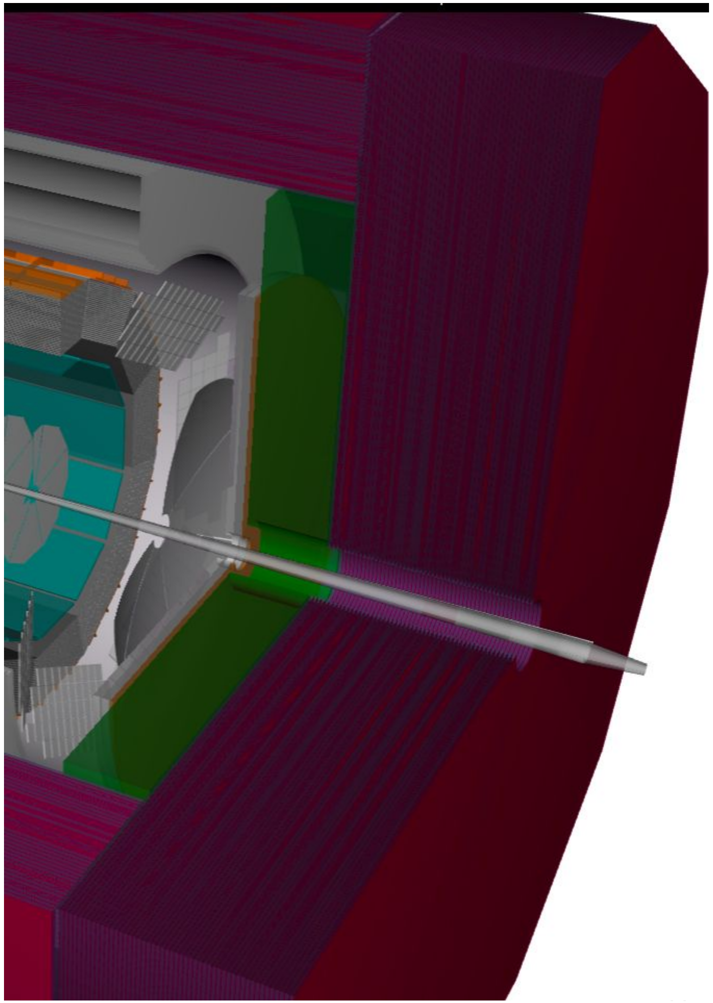


Truth information also in the samples.
Truth-level jets, resolution plots coming soon.
Will be included in next tutorial

Jet reconstruction

With forward calorimeter system





Give it a try:

https://github.com/miguelignacio/calostudies/blob/main/jets.ipynb

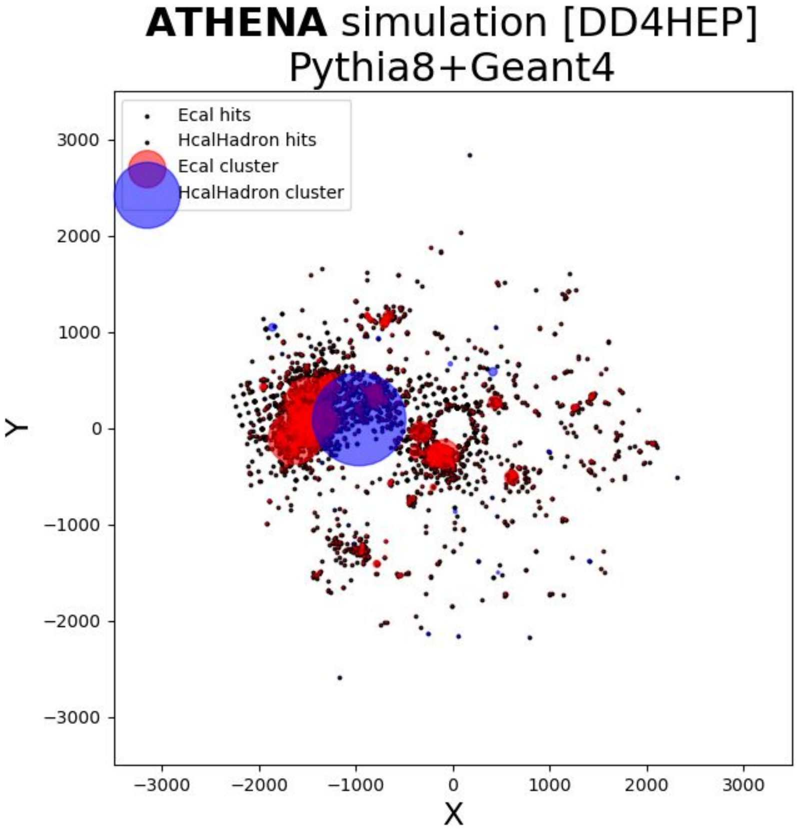
Coming soon: examples of hadronic reconstruction, calo-based electron ID, including isolation

```
def isolation(cone_theta,cone_phi, cluster_container, E_threshold=0.1):
    nclusters= len(cluster_container['E'])
    #if(cone_theta<0.05):
    #    return -999
    cone_eta = -np.log(np.tan(cone_theta/2.0))
    cone_iso = 0.0
    for i in range(nclusters):
        clus_E = cluster_container['E'][i]/1000.0
        if(clus_E<E_threshold):
            continue
        clus_E = cluster_container['E'][i]/1000.0
        clus_phi = cluster_container['phi'][i]
        clus_theta = cluster_container['theta'][i]
        clus_eta = -np.log(np.tan(clus_theta/2.0))
        clus_pt  = clus_E*np.sin(clus_theta)
        dr = np.sqrt((clus_phi - cone_phi)**2 + (clus_eta-cone_eta)**2)
        if(dr<0.4):
            cone_iso += clus_E

    print('Cone phi %2.2f, cone eta %2.2f, cone theta %2.2f, cone E= %2.2f GeV'%(
    return cone_iso

def find_electron(cluster_container,hcal_container, E_threshold=5.0):
    #Returns cluster with the highest pT in the event
    #print(cluster_container.keys())
    ptmax = 0.0
    index_max = -999
    nclusters= len(cluster_container['E'])

    for i in range(nclusters):
```

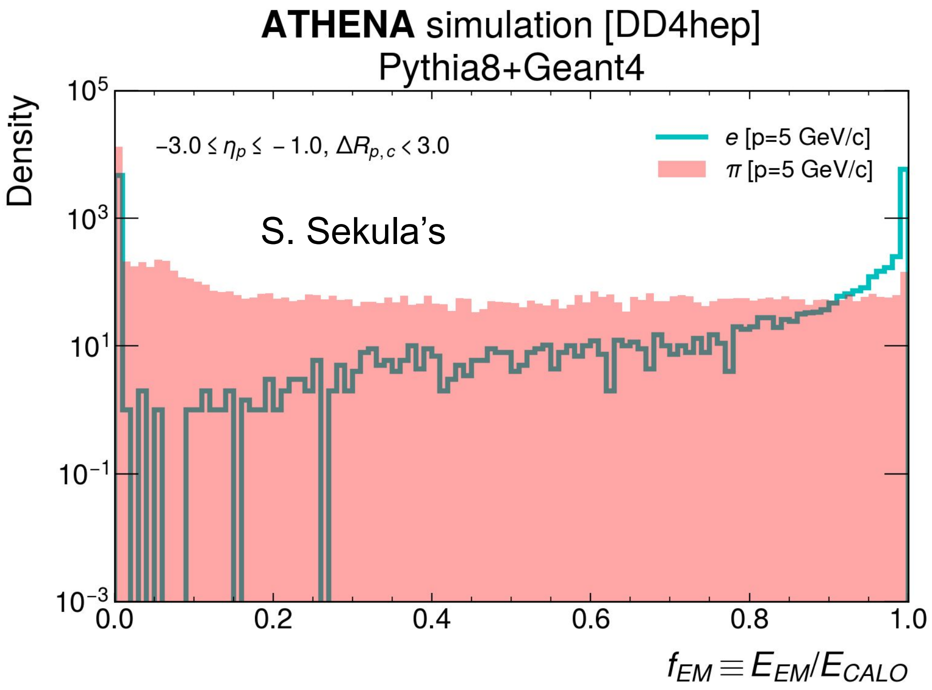
```
def get_Empz(cluster_container):
    nclusters= len(cluster_container['E'])
    Empz = 0.0
    #print('nclusters ', nclusters)
    for i in range(nclusters):
        #print(cluster_container['E'][i])
        Empz += cluster_container['E'][i]*(1-np.cos(cluster_container['theta'][i]))
    #print(Empz)
    return Empz/1000.0

def get_total_pxpy(cluster_container,E_threshold=0.1):
    nclusters= len(cluster_container['E'])
    sum_px = 0.0
```

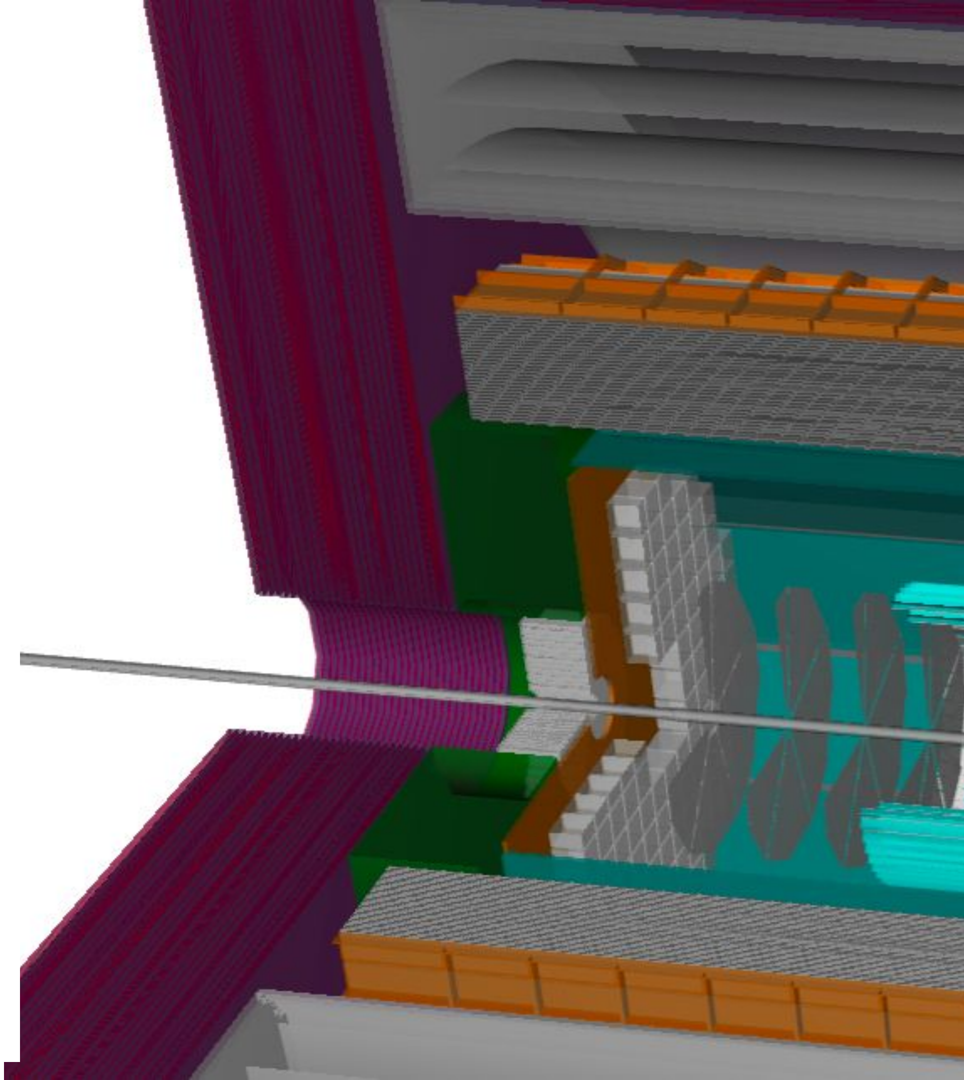
Step #3: Potential analyzes that you can contribute to

Jet performance at high rapidity, study impact of energy leakage to beam-pipe

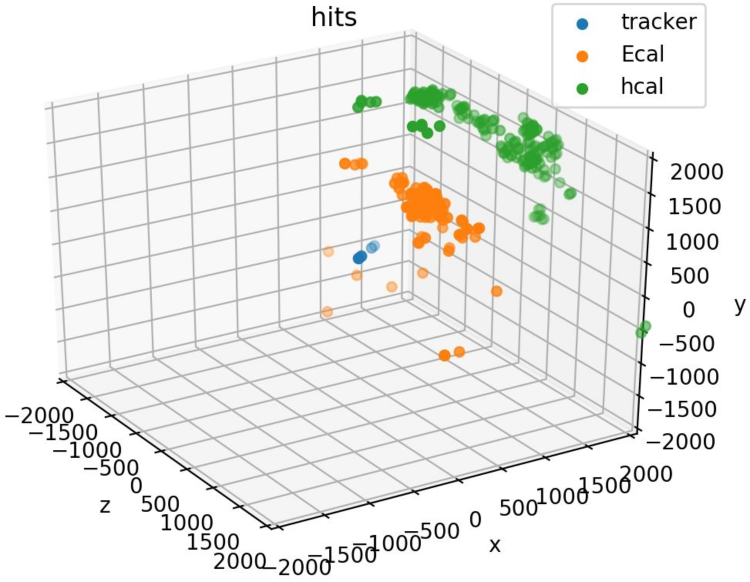


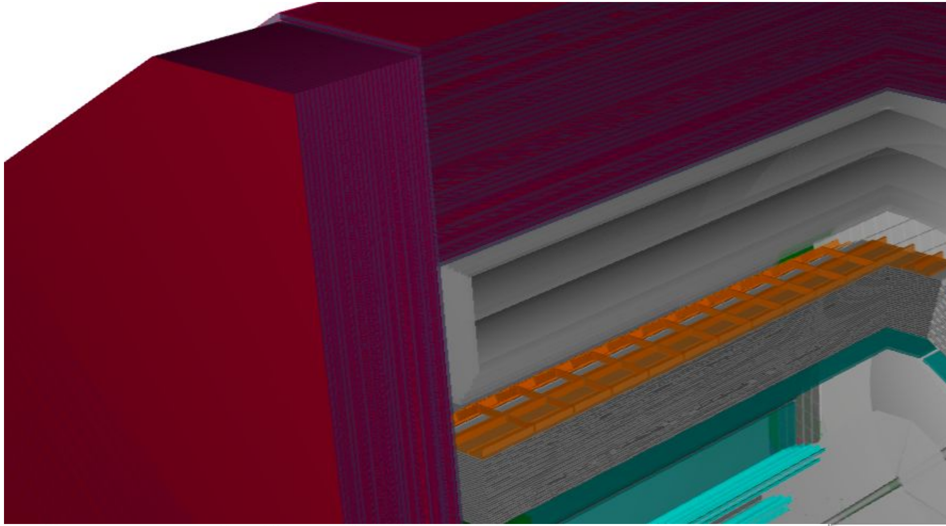
Electron-pion separation With HCAL

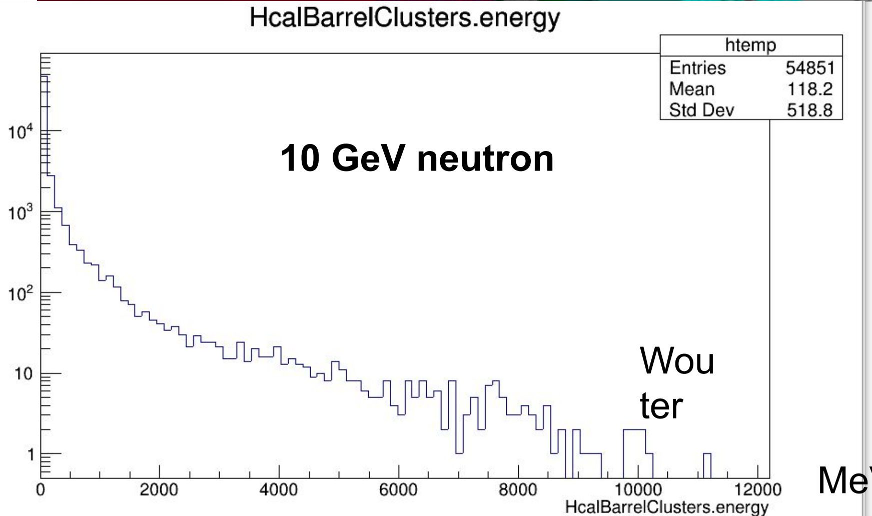




Impact of magnet on barrel HCAL, jet/hadronic reco performance







MeV