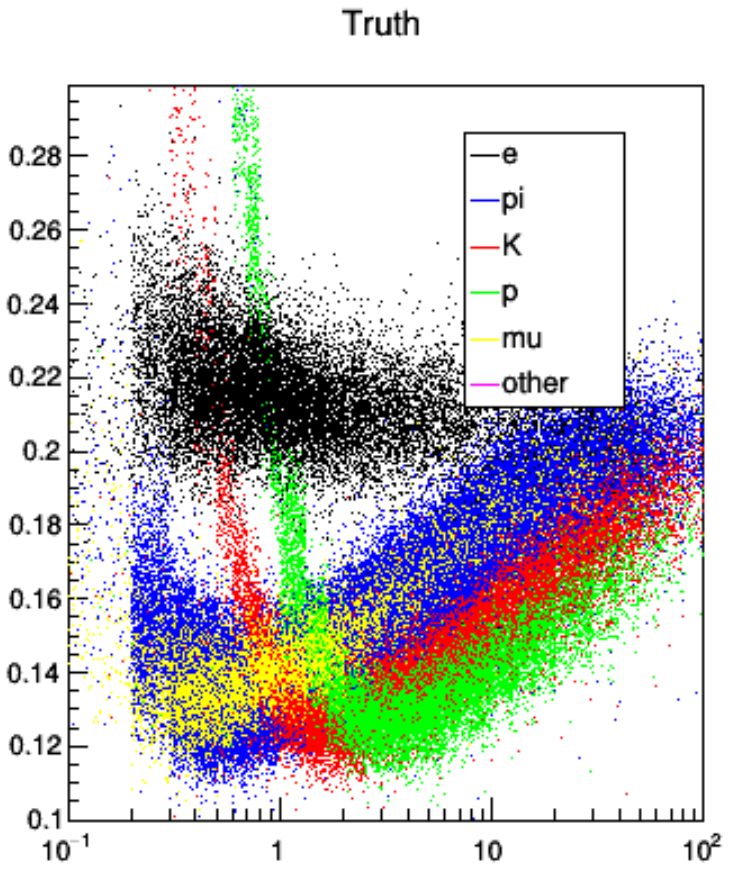
dE/dx (bb, 500GeV, large model)



- All tracks, including primary vertexes

- I am quite sure that the angular correction is done
  - It is written in the thesis text
  - If I apply it once more, the separation becomes quite much worst
  - But it is optimally applied? How has it been optimized? Ar these questions to raise in the ILD meeting or better by private discussions first?
- The new samples include already some Particle Indentification (see PID Handler class in LCIO)
- The two only PID that identify/separate pion/kaon proton are:
  - dEdxPID
  - LikelihoodPID (which seems to have input from the previous)

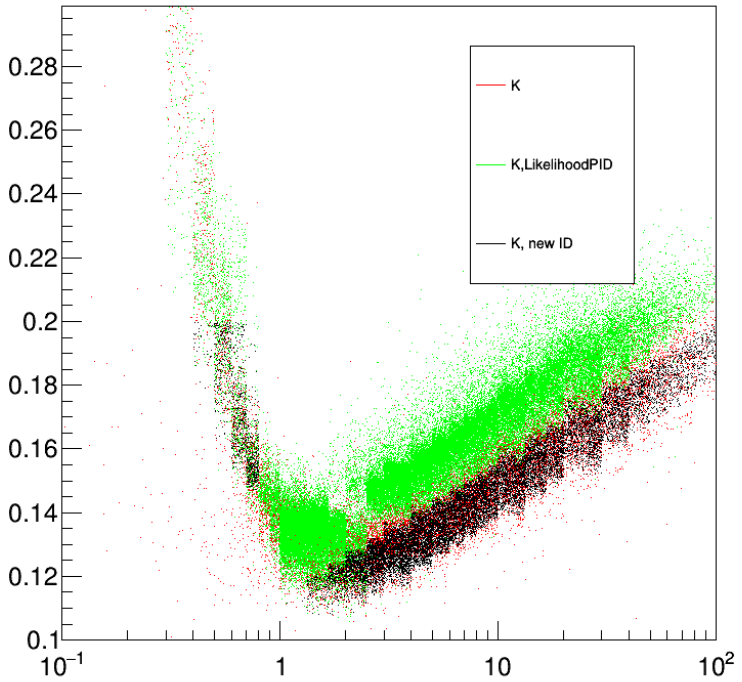
- How to use the PIDHandler? See example in src/ParticleTagger.cc

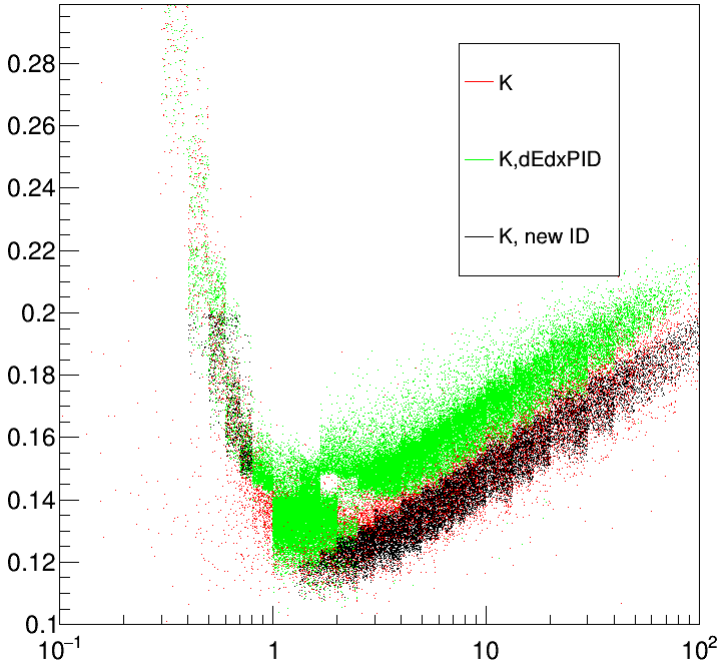
```
PIDHandler pidh(pfoCol); //pfoCol is the name of the PandoraPFOs collection
for(int i=0; i<pfoCol→getNumberOfElements(); i++) {
  ReconstructedParticle * recoparticle = static_cast< ReconstructedParticle * > (pfoCol->getElementAt(i));
  int pid = pidh.getAlgorithmID(“LikelihoodPID”);
  int pdg = pidh.getParticleID(recoparticle, pid).getPDG();//this is the PDG obtained by LikelihoodPID
}
```

- https://github.com/QQbarAnalysis/ParticleTagger Branch QQbarAnalysisBranch2018

dE/dx (bb, 500GeV, large model)

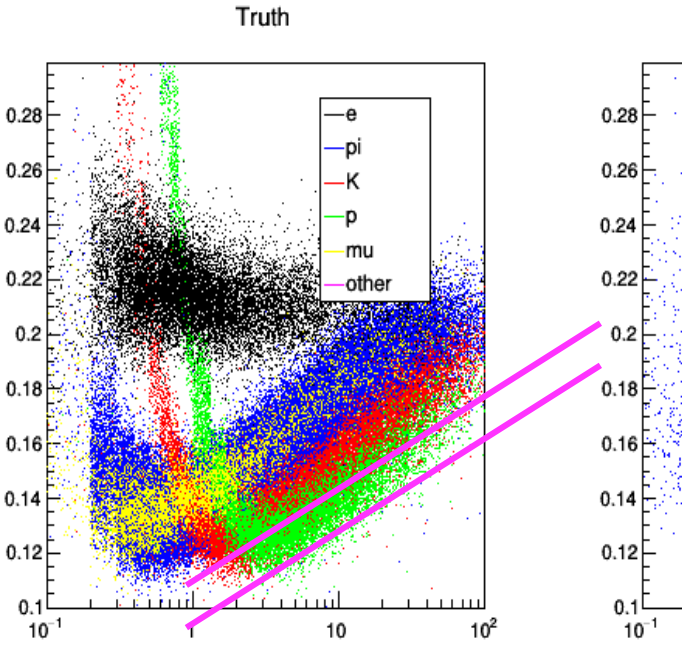
- How the two DST PID algorithms perform (only for kaons)? **The red K are the truth. To be compared with the green.**
- Ignore the black points for the moment !!





dE/dx (bb, 500GeV, large model)

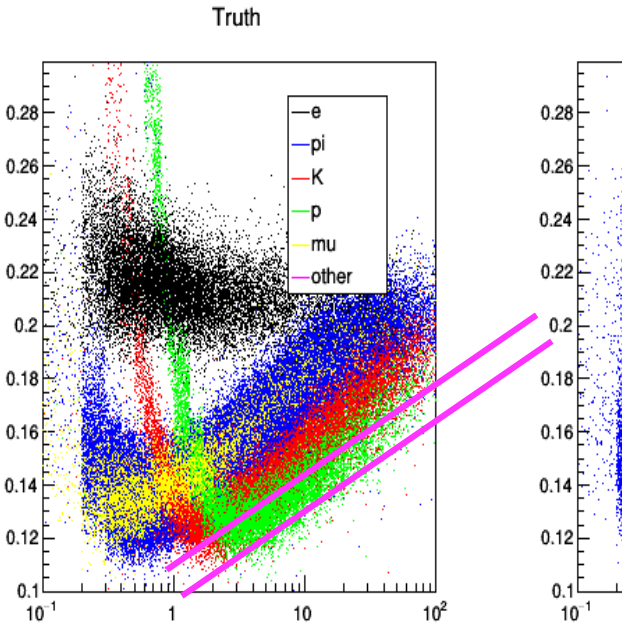
- LikelihoodPID and dEdxPID algorithms perform very badly for kaons.
- It seems obvious that there is a problem in the parametrization of the dEdx curve.
- I have redone Sviatoslav's analysis for dEdx in the ParticleTagger algorithm.



- Basic idea: parametrize the area where the kaon fall and use it for particle separation.
- A **new PIDalgorithm** is created and saved in the PIDHandler of the **PandoraPFOs**
  - **KaonTagger**
  - Without likelihood or probability value.
- It works very nice for " high" p. For low momentum, the likelihood is already good.

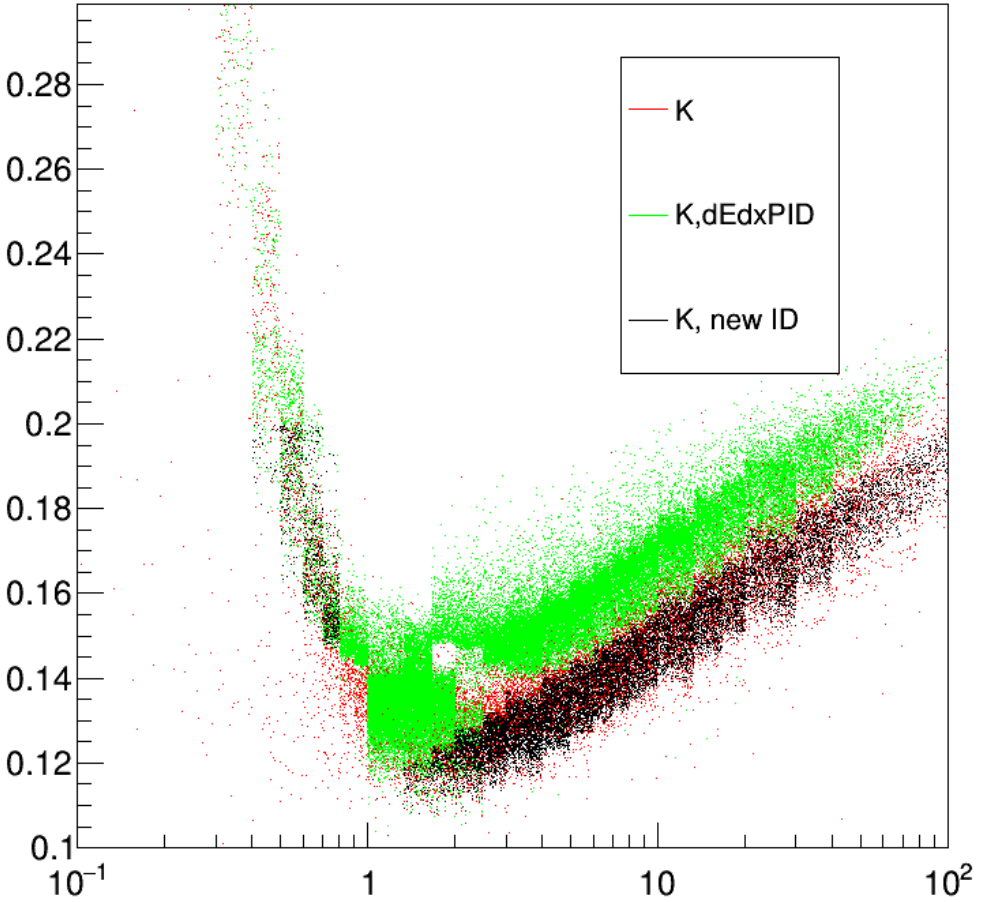
Using the new and the official PIDs

- **First, I reject particles that:**
  - Have low momentum (<2 GeV) and LikelihoodPID says that it is a proton
  - Have low momentum (<0.8 GeV) and LikelihoodPID says that it is a pion
  - Has dEdx>0.20 & LikelihoodPID says that it is a pion or an electron
- **If the paticle is not rejected, I identify it as a kaon if:**
  - Has lowmomentum (<2 GeV) and LikelihoodPID says that it is a kaon
  - Has large momentum and it is in the area covered by the magenta lines.

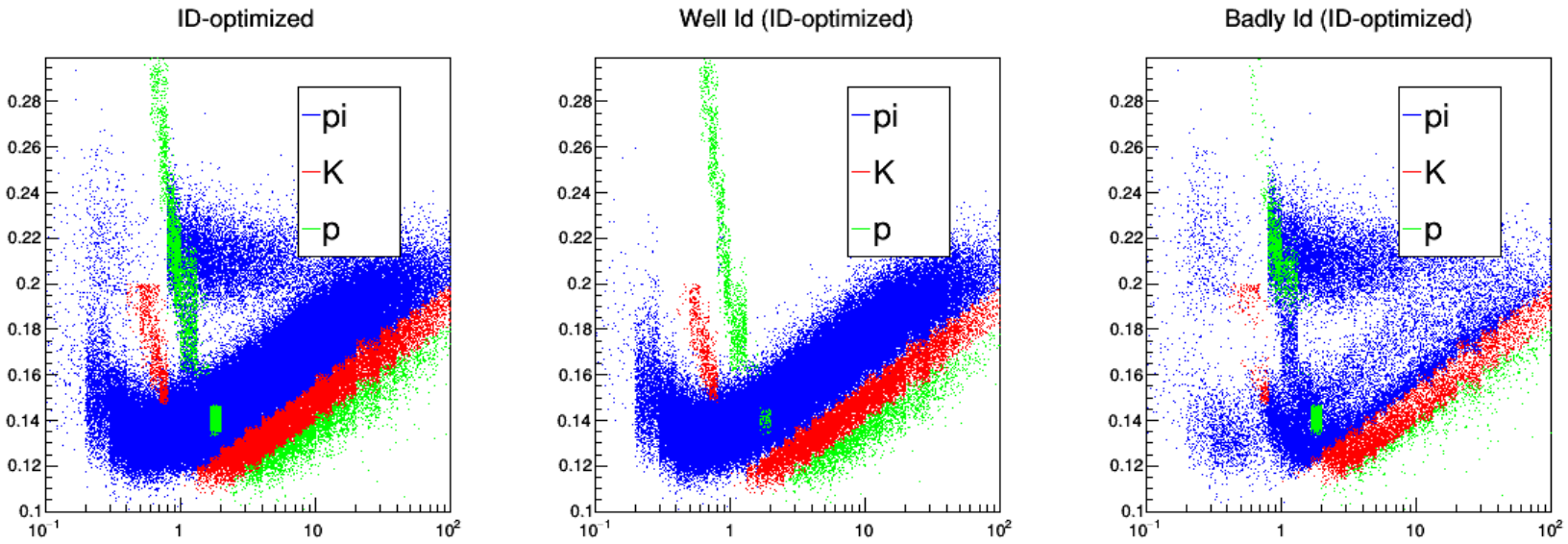


Using the new and the official PIDs

- **Now we can look to the black points, and compare them with the red points.**



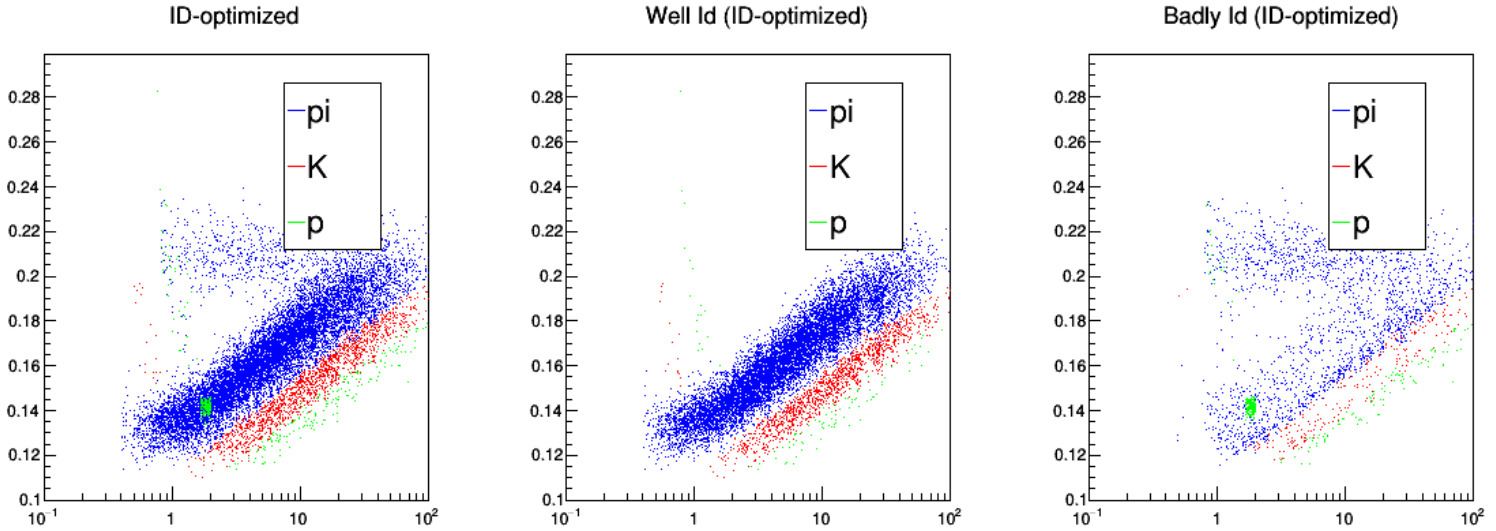
Using the new and the official PIDs



- This plots are not good to measure purities and efficiencies… Let's calculate them

```
  ################## new ID ##################
Simulated kaons: 24071, pions: 201012, protons: 13111, electrons: 20019, muons: 6191, other: 20019
Reco, kaons: 16338, pions: 197604, protons: 10003
Well Reco, kaons: 10743, pions: 169682, protons: 5313
Contamination in kaons id, pions:: 1137, protons: 4206, electrons: 159, muons: 39
  ------------------------------------------------------------
KAON ID: eff =0.678742  purity=0.657547
  ------------------------------------------------------------
```

- Only secondary vertexes, with |cos(theta)|<0.95.
-




```
  ----------------------------------------------------------------------

  ################ new ID ################
Simulated kaons: 2236, pions: 11734, protons: 299, electrons: 867, muons: 405, other: 867
Reco, kaons: 1633, pions: 13315, protons: 414
Well Reco, kaons: 1380, pions: 11345, protons: 105
Contamination in kaons id, pions:: 95, protons: 140, electrons: 14, muons: 4
  ----------------------------------------------------------------------
KAON ID: eff =0.730322  purity=0.84507
  ----------------------------------------------------------------------
```

- Note: adding the low momentum kaons increase the efficiency by ~2% with no loss of purity.

How to use the ParticleTagger

- Add it to the processor chain, at the beginning or at least before QQbarProcessor.
- Example of steering file is in:
  https://github.com/QQbarAnalysis/ParticleTagger/tree/QQbarAnalysisBranch2018/scripts
- It saves the new PIDalgorithm in the PIDHandler of the PandoraPFOs → this has to be used in your analysis!
  - See introductory slides and ParticleTagger processor for tips.
- It also produces a root file. Example of analysis of this root file is in
  https://github.com/QQbarAnalysis/ParticleTagger/tree/QQbarAnalysisBranch2018/analysis
-

```
<processor name="MyParticleTagger" type="ParticleTagger">
  <parameter name="ROOTFileName" type="string" > pid_l5.root </parameter>
  <parameter name="NewPID" type="string" > KaonTagger </parameter>
  <parameter name="PFOCollection" type="string" > PandoraPFOs </parameter>
  <parameter name="TrackRelCollection" type="string" > MarlinTrkTracksMCTruthLink </parameter>
  <parameter name="slope_kaon_selection" type="float" > 0.019 </parameter>
  <parameter name="upper_limit_kaon_selection" type="float" > 0.1115 </parameter>
  <parameter name="lower_limit_kaon_selection" type="float" > 0.096 </parameter>
</processor>
```