PID Calibration

Sneha Malde

Scope

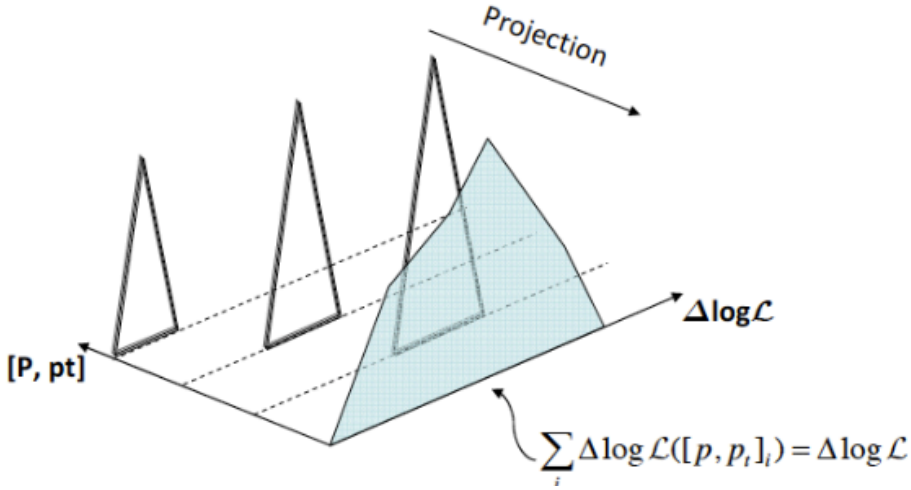
- Charged PID
- How it is done
- What will be available to you
- Outline of systematic uncertainty application
- FAQ
- Your questions.
- Daniel's tutorial on using PIDCalib2
- README has examples

Charged PID

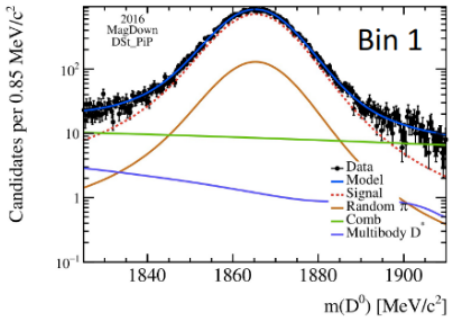
- Simulation PID response doesn't match that of data
- Define a fully data driven method to derive PID efficiencies
- Choose samples of decays where particle type can be determined **without** PID selection (ideally at all, in practice at least no PID on that track)
- Desired properties
  - High purity
  - High yields
  - Broad kinematic coverage
  - A rate that doesn't saturate the bandwidth of the calibration stream

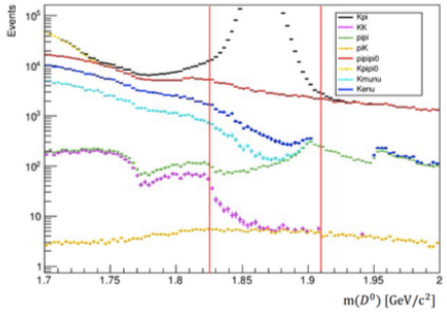
Basic Premise

- The PID response is track based
- Find the variable to which the PID response is sensitive
- Tracks within a sufficiently small bin of these variables will all have the same PID response



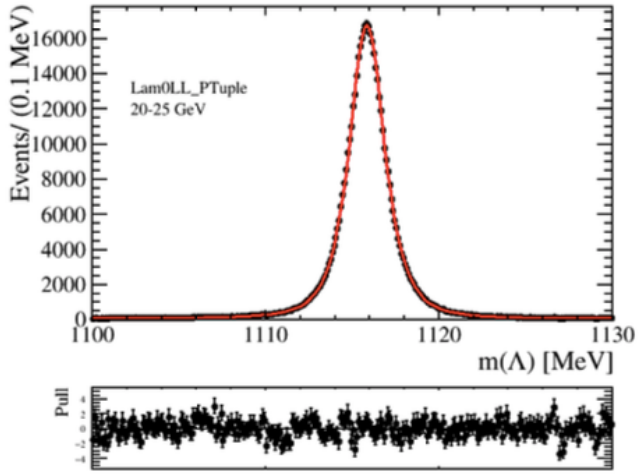
Basic samples (esp for EM)





- For Kaons and Pions: D* → D^0 pi^+, D^0→K- pi+
- For Protons: L0 →ppi-
- For Muons: B→J/psi K, J/psi → mumu
- For electrons: B→J/psi K, J/Psi ->ee

- Other possible samples, but unlikely to be available for EM due to complexities requiring extra data validation.



Calibration method

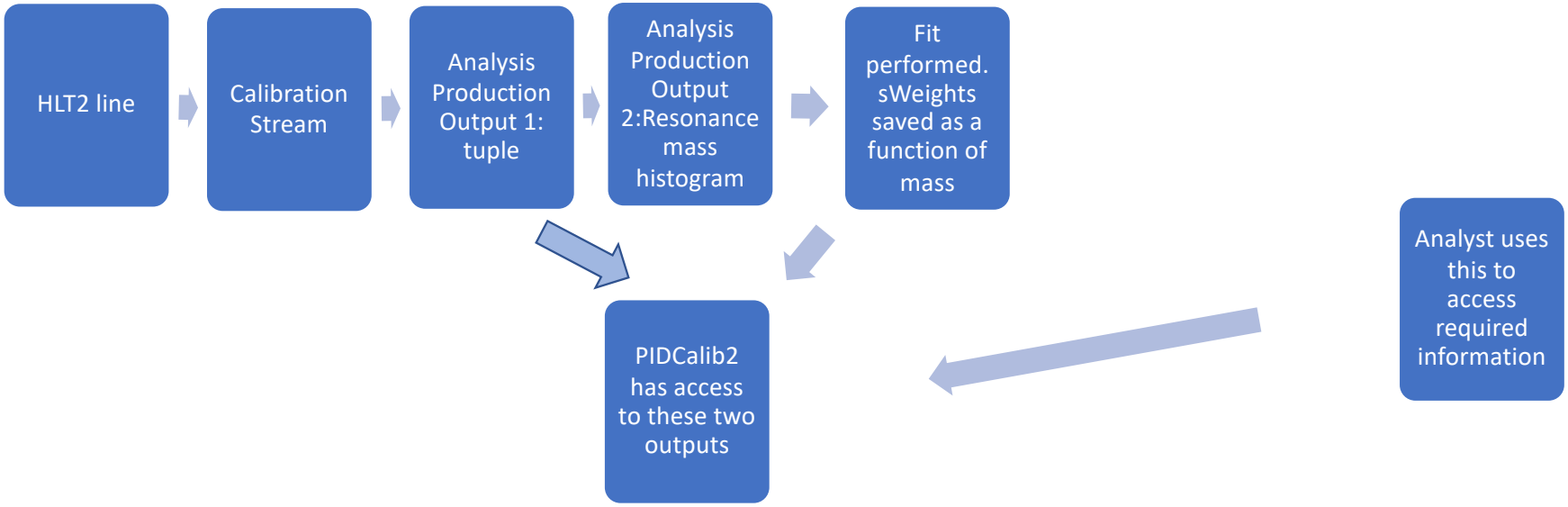
Requirements:

- One cut doesn’t fit all – calibration must work for any PID cut choice
- Relatively fast speed – no analysis wants to spend weeks doing this
- Low learning curve – undesirable for every analysis to reinvent the wheel

Design:

- Extraction of the ”track samples” is a dedicated responsibility
- Tracks are assigned signal weights to remove background from sample
- Analyst simply counts weights before and after cut in order to determine efficiency

Run 3 vision

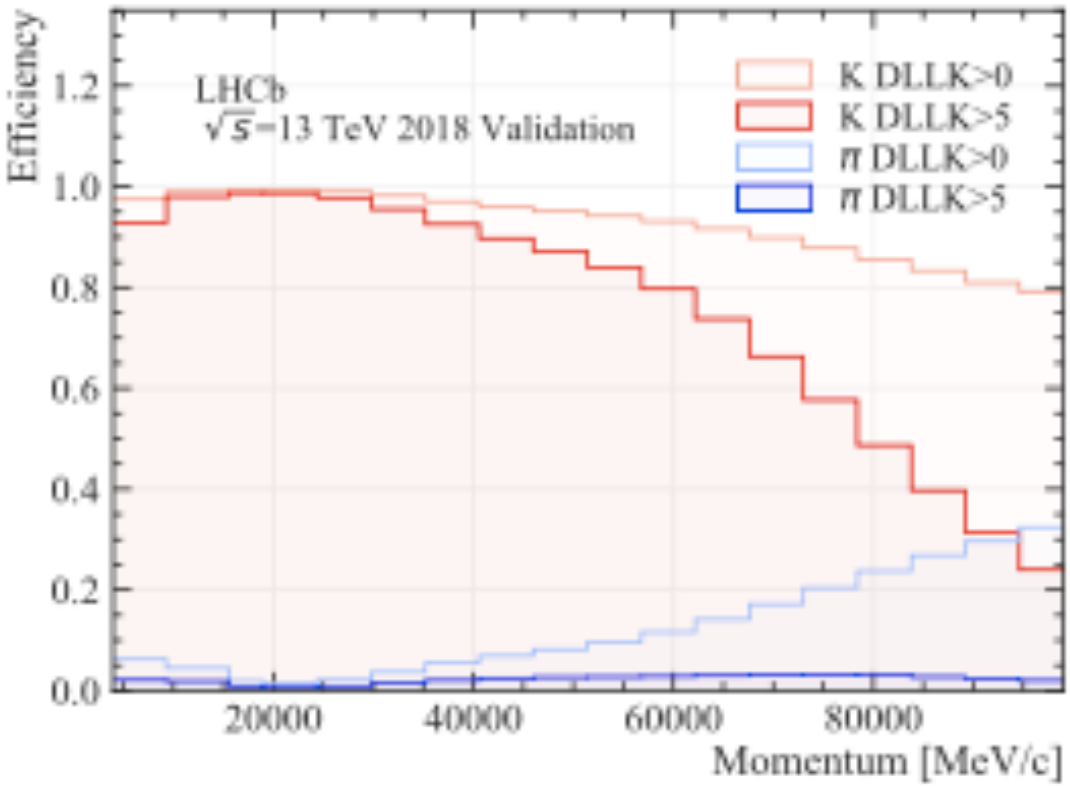

HLT2 line
Calibration Stream
Analysis Production Output 1: tuple
Analysis Production Output 2:Resonance mass histogram
Fit performed. sWeights saved as a function of mass
PIDCalib2 has access to these two outputs
Analyst uses this to access required information

First you need binned efficiencies

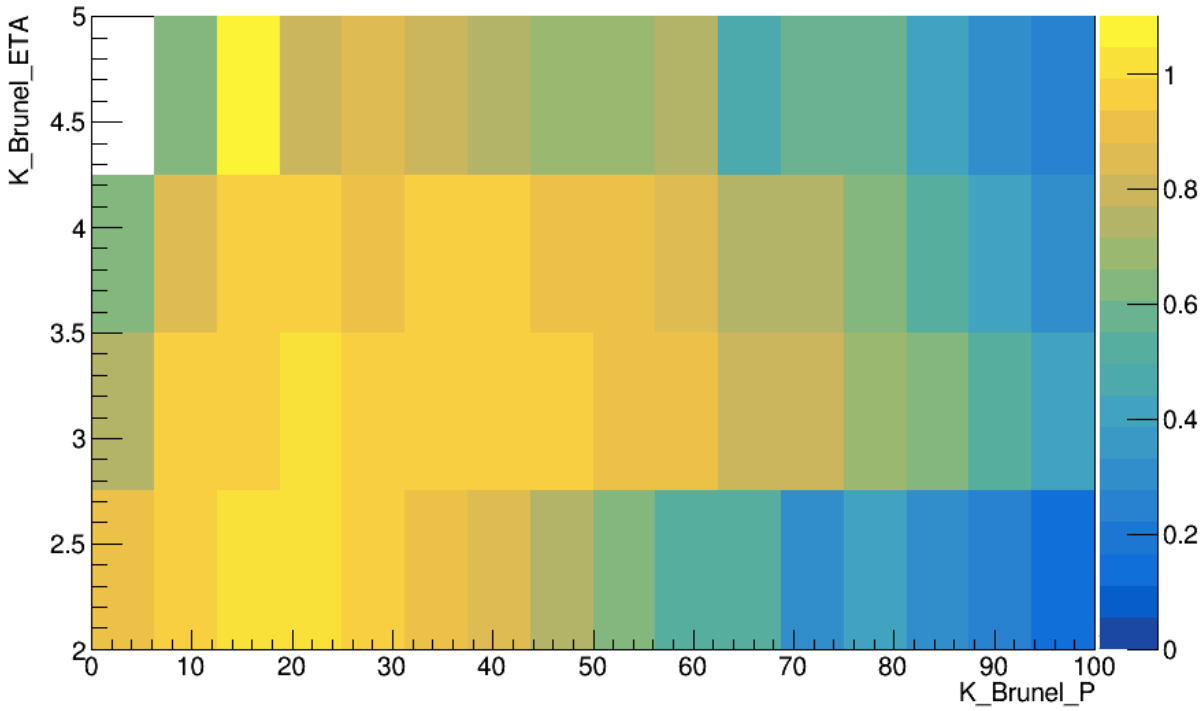
- Create efficiency table:

- You enter the cuts of choice
- You enter the defined bins

- PIDCalib2 looks up the calibration tracks in each bin and counts all sWeights and those that pass the cut. Saves information of efficiency
- Can put these into a root histogram if you wish.



In 2 dimensions



Ntracks or not?

- If your reference sample is MC, then it is likely that your Ntrack distribution doesn’t match that of data.
- In this case it can make sense to only bin in P, eta or Pt. You then assume that the calibration sample has similar multiplicity to your own data, and therefore assume an integration over this variable

What to think about:

- Traditional binning variables: P, Pt or eta, N_tracks (may not need)
- ProbNN variables may have some dependence on other track quantities
- Where to place the bins should depend on the kinematic dependence of the tracks in your analysis.
  - Trade between small bins vs being sensitive to statistical fluctuations due to yields of calibration data.

Next: Calculate the efficiency for your analysis

- Essentially, find the average efficiency over all tracks in your physics analysis

- PIDCalib2 can accept your input tuple, and add to it efficiency information from the histograms you created in Step 1.

Practical Tips (1):

- What if I want to put two PID cuts on a track e.g DLLK>5 and ProbNNp<0.2
  - No problem! The response of the two PID variables is correlated:
  - Do not create separate histograms for the two PID cuts and multiply together
  - Instead ask PIDCalib2 to give you the histogram for the combined cut
  - ε(DLLK >5 && ProbNNp <0.2) != ε(DLLK>5) * ε(ProbNNp<0.2)

Practical Tips (2):

- I have a three track decay, each track has different PID cut applied (e.g Lc ->pKpi), how do I find the PID efficiency of the decay?
- Multiply the track efficiencies for each event first. Then take the average overall events
- $\varepsilon(Lc) = < \varepsilon(P)* \varepsilon(K)* \varepsilon(pi)> != < \varepsilon(p)>*< \varepsilon(K)>*< \varepsilon(pi)>$

How much calibration data? When available?

- You need to match the run range of calibration data to the run range of your analysis.
  - PID performance is fill dependent, might be stable but might not. You can cut the calibration sample by run if required
  - From experience, calibration yields are sufficient for HF analysis
- AP will be run and take some time (presumably your analysis has the same issue)
  - Hope that fits are sufficiently automated that time from end of AP to availability in PIDCalib2 ~24-48 hours

What's the error?

- Calibration statistical uncertainty should hopefully be too small to worry about. It is however calculable from the PIDCalib2 output. The statistical uncertainty associated to your reference sample should be larger
- There is a systematic associated with your choice of binning scheme
  - The assumption that the efficiency is constant across the bin is an assumption
  - Usually this is a significant contribution
- sWeights come with issues. Run2 estimate ~0.2% absolute. Expect similar in Run 3. The binning systematic is still likely to be bigger

PIDCalib2 or PIDGen

Remember, both derive from the same calibration samples, and the same sTables.

Hence both method have the same limitations in this respect

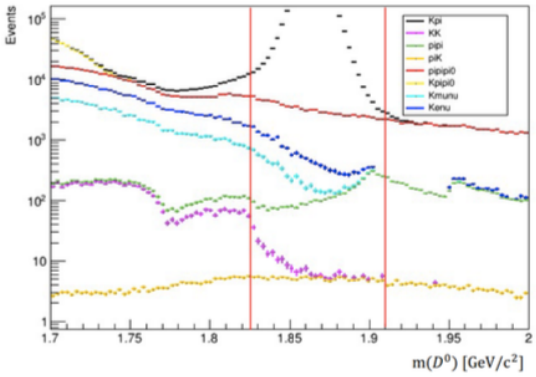
PIDGen – effectively resamples PID distributions from the calibration data and replaces the variable in your tuple dependent on track kinematics. Particularly useful if you want to put the PID distribution into a NN.

PIDCorr – resampling taking into account correlations between PIDvariables for the same track.

Things PIDCalib won't do for you

- Decay in flight
- Provide samples for physics analyses (there are often veto cuts on the samples to keep rates manageable)
- Replace your own thinking especially when it comes to systematic uncertainties
- Currently doesn't provide an automated Fit & Count derived efficiency

Dealing with this is non-trivial




Questions ?