Extra practice with O2-HF

V. Kucera, L. Dello Stritto

HF O2 Hackathon

09/12/2021

Exercise 1

- Add a histogram of the D^0 candidate rapidity Y in the D^0 task:
  - Add the Y column in the candidate table
  - Compute the D^0 rapidity
  - Add a histo to the histogram registry and fill it

- Hints:
  - Y definition in RecoDecay.h: link
  - Same syntax already implemented in the taskMini.cxx for the decayLenght

```
DECLARE_SOA_DYNAMIC_COLUMN(DecayLength, decayLength, //!
                           [](float xVtxP, float yVtxP, float zVtxP, float xVtxS, float yVtxS, float zVtxS) -> float { return RecoDecay::distance(array{xVtxP, yVtxP, zVtxP}, array{xVtxS, yVtxS, zVtxS}); });
```

- Y D^0 computation like the inv mass one:

```
template <typename T>
auto InvMassD0(const T& candidate)
{
  return candidate.m(array{RecoDecay::getMassPDG(kPiPlus), RecoDecay::getMassPDG(kKPlus)});
}
```

Exercise 2

- Change the partition below, employed to loop only on the selected D^0 candidate, into an upfront Filter:

```
Partition<soa::Join<aod::HfCandProng2, aod::HfSelCandidateD0>> selectedD0Candidates = aod::hf_selcandidate_d0::isSelD0 >= flagSelCandD0 || aod::hf_selcandidate_d0::isSelD0bar >= flagSelCandD0bar;
```

- Hints:
  - ALICE O2 documentation for Filtering: link
  - Anton's slides: slides

Exercise 2

- Change the partition below, employed to loop only on the selected D^0 candidate, into an upfront Filter:

```
Partition<soa::Join<aod::HfCandProng2, aod::HfSelCandidateD0>> selectedD0Candidates = aod::hf_selcandidate_d0::isSelD0 >= flagSelCandD0 || aod::hf_selcandidate_d0::isSelD0bar >= flagSelCandD0bar;
```

- Hints:
  - ALICE O2 documentation for Filtering: link
  - Anton's slides: slides

- Solution:

```
o2::framework::expressions::Filter filterSelectCandidates = (aod::hf_selcandidate_d0::isSelD0 >= flagSelCandD0 || aod::hf_selcandidate_d0::isSelD0bar >= flagSelCandD0bar);
void process(soa::Filtered<soa::Join<aod::HfCandProng2, aod::HfSelCandidateD0>>& selectedD0Candidates)
```

Exercise 3

- Compute the number of tracks in the D^0 task and add a histogram.
- Compute the number of D^0 candidate number per collision and add a histogram.

- Hints:
  - You need to add the collisions and the tracks to your process function
  - Retrieve the number of track with the method ***.size()***

Exercise 3

- Compute the number of tracks in the D^0 task and add a histogram.
- Compute the number of D^0 candidate number per collision and add a histogram.

- Hints:
  - You need to add the collisions and the tracks to your process function
  - Retrieve the number of track with the method ***.size()***

- Solution:

```
void process(aod::Collision& collision, soa::Join<aod::Tracks, aod::TracksExtended>& tracks,soa::Join<aod::HfCandProng2, aod::HfSelCandidateD0>& selectedD0Candidates)
{

  registry.fill(HIST("hMultiplicity"), tracks.size());
  registry.fill(HIST("hNCand"), selectedD0Candidates.size());
```

Exercise 4

- Change the constant cut on the cosine of the pointing angle in a p_T dependent cut in the D^0 selector.

```
// cosine of pointing angle
if (candidate.cpa() < cpaMin) {
  return false;
}
```

- Hints
  - You need to define a vector with the desired p_T binning: link
  - You need to define a vector with the p_T dependent cuts : link
  - Syntax to apply the p_T dependent cut discussed in Antonio, Fabio and Vit talk: link

```
Configurable<std::vector<double>> pTBins{"pTBins", std::vector<double>{hf_cuts_d0_topik::pTBins_v}, "pT bin limits"};
Configurable<LabeledArray<double>> cuts{"D0_to_pi_K_cuts", {hf_cuts_d0_topik::cuts[0], hf_cuts_d0_topik::npTBins, hf_cuts_d0_topik::nCutVars, hf_cuts_d0_topik::pTBinLabels,
                                                            hf_cuts_d0_topik::cutVarLabels}, "D0 candidate selection per pT bin"};

// normalised decay length in XY plane
if (candidate.decayLengthXYNormalised() < cuts->get(pTBin, "normalized decay length XY")) {
  return false;
}
```

  - Check the name of the needed variable: link