Selector features and upgrades

F. Colamaria, V. Kucera, A. O. da Silva

O2 HF Hackathon - 09/12/2021

REMINDER: GENERAL HF CODE STRUCTURE

Input: O^2 tables (tracks, collisions, …)

This is just the core, to highlight the flow of candidate processing: several related/helper codes not included here

Track skimming

- **HFTrackIndexSkimsCreator**
- Applies event selection, identifies possible multiplets of HF hadron daughters, via track filtering + kinematic cuts + minimal set of cuts for quick candidate preselection
- **Produces**: index tables for N-prong daughter tracks

Candidate processing

- **HFCandidateCreator2Prong**, **HFCandidateCreator3Prong**
- Full vertexing and building of candidate with info on all selection variables
- **Produces**: tables of candidate info (kinematics, cut variable values, …)

Candidate selectors

- Performs (channel-specific) candidate selection based on user-defined cut values → Topological selection + PID selection
- **Produces**: (joinable) table with outcome of the selection → flag(s)

HF-user tasks

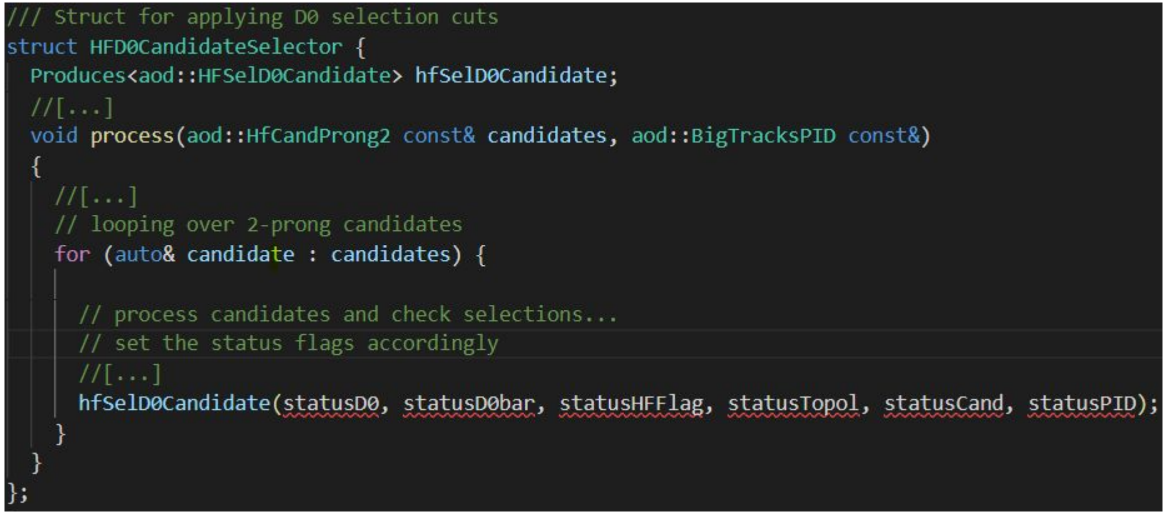
- **Examples**: taskD0, taskDPlus, taskLc
- Written by users for specific analysis operations (M_{inv} distributions, efficiencies, …)
- **Produce**: user-defined outputs, e.g. histograms

HF SELECTOR - CODE STRUCTURE

- Devoted to perform selection of candidates built by creator, transmitting the information to the analysis task
- One selector code for each analysis channel (linked to the corresponding candidate creator)
    - Specific variables, selection criteria and and way of applying the candidate selection
    - Anyway, the general structure is similar for all the cases
- In the following, examples will be done for D0→Kpi reconstruction

D0 selector structure

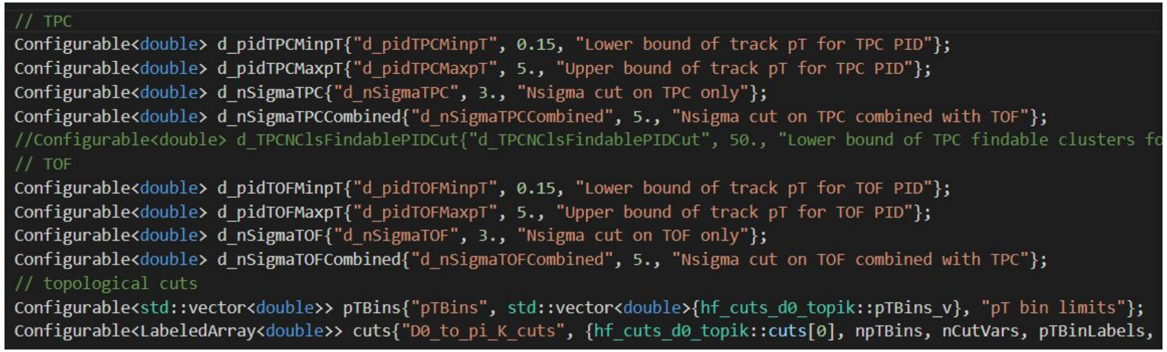
- Produces the hfSelD0Candidate table, storing the values of the flags for the outcome of the selection (several steps in this case)
- Loops over the candidate table (produced by candidate creator)
    - For each candidate, a row is added to the table with the corresponding flag values



```
/// Struct for applying D0 selection cuts
struct HFD0CandidateSelector {
  Produces<aod::HFSelD0Candidate> hfSelD0Candidate;
  //[...]
  void process(aod::HfCandProng2 const& candidates, aod::BigTracksPID const&)
  {
    //[...]
    // looping over 2-prong candidates
    for (auto& candidate : candidates) {

      // process candidates and check selections...
      // set the status flags accordingly
      //[...]
      hfSelD0Candidate(statusD0, statusD0bar, statusHFFlag, statusTopol, statusCand, statusPID);
    }
  }
};
```

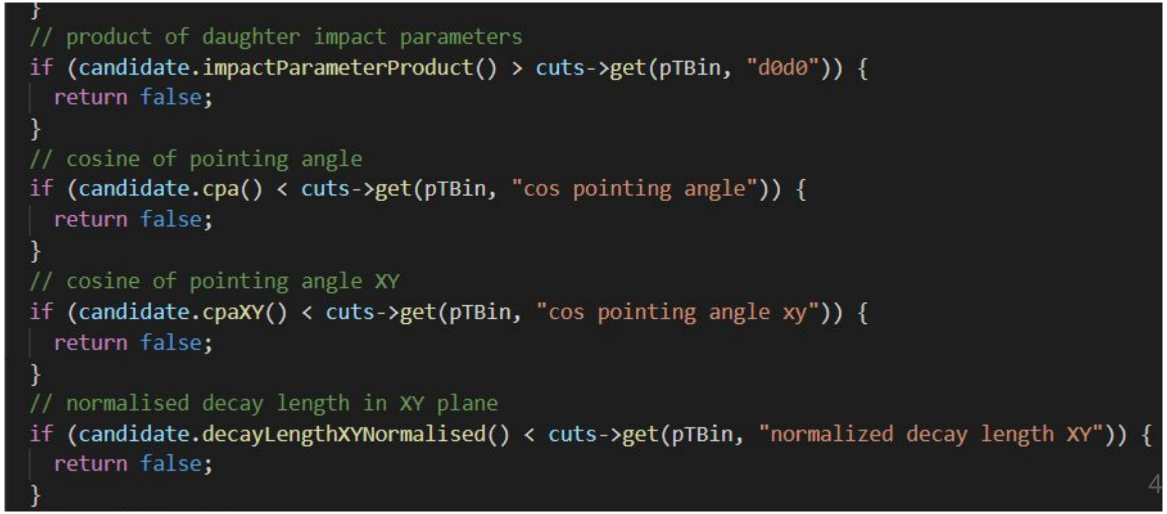
HF SELECTOR - CODE STRUCTURE

- Inside the candidate loop, the candidate selection is performed, applying:
  - Topological selection on the candidate
  - PID selection on the daughter tracks
- The configuration of the selection (topological cut thresholds, pT ranges and nSigma for PID, etc.) is passed to the selector via Configurables (see later)



```
// TPC
Configurable<double> d_pidTPCMinpT{"d_pidTPCMinpT", 0.15, "Lower bound of track pT for TPC PID"};
Configurable<double> d_pidTPCMaxpT{"d_pidTPCMaxpT", 5., "Upper bound of track pT for TPC PID"};
Configurable<double> d_nSigmaTPC{"d_nSigmaTPC", 3., "Nsigma cut on TPC only"};
Configurable<double> d_nSigmaTPCCombined{"d_nSigmaTPCCombined", 5., "Nsigma cut on TPC combined with TOF"};
//Configurable<double> d_TPCNClsFindablePIDCut{"d_TPCNClsFindablePIDCut", 50., "Lower bound of TPC findable clusters fo
// TOF
Configurable<double> d_pidTOFMinpT{"d_pidTOFMinpT", 0.15, "Lower bound of track pT for TOF PID"};
Configurable<double> d_pidTOFMaxpT{"d_pidTOFMaxpT", 5., "Upper bound of track pT for TOF PID"};
Configurable<double> d_nSigmaTOF{"d_nSigmaTOF", 3., "Nsigma cut on TOF only"};
Configurable<double> d_nSigmaTOFCombined{"d_nSigmaTOFCombined", 5., "Nsigma cut on TOF combined with TPC"};
// topological cuts
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], npTBins, nCutVars, pTBinLabels,
```

Snippet of the code inside the candidate loop (checking the value of few variables) → In this specific case, the return value is set to the flag statusTopol



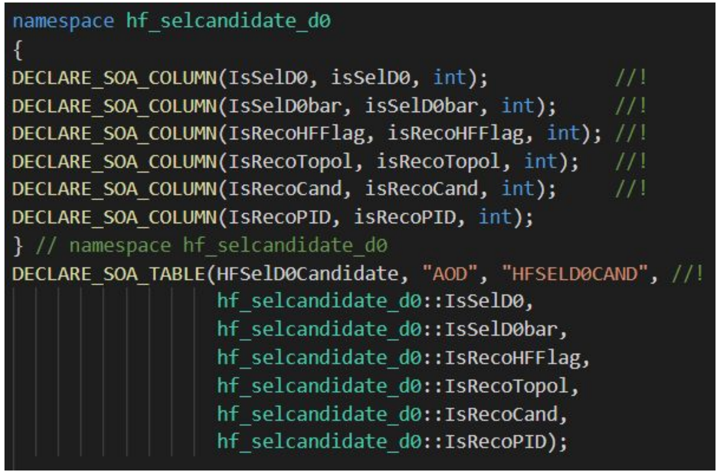
```
}
// product of daughter impact parameters
if (candidate.impactParameterProduct() > cuts->get(pTBin, "d0d0")) {
  return false;
}
// cosine of pointing angle
if (candidate.cpa() < cuts->get(pTBin, "cos pointing angle")) {
  return false;
}
// cosine of pointing angle XY
if (candidate.cpaXY() < cuts->get(pTBin, "cos pointing angle xy")) {
  return false;
}
// normalised decay length in XY plane
if (candidate.decayLengthXYNormalised() < cuts->get(pTBin, "normalized decay length XY")) {
  return false;
}
```

SELECTION-RELATED TABLES

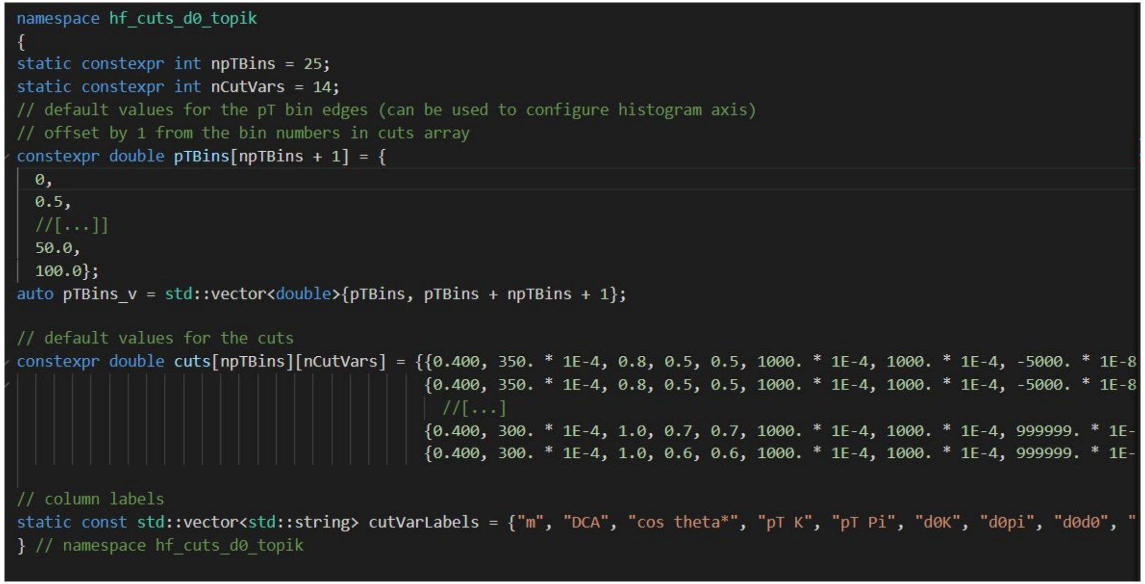
Definition of the table produced by the selector is in PWGHF/DataModel/HFCandidateSelectionTables.h

- Six flags: two ‘final’ selection flags for D0 and D0bar and further step-by-step flags

The tables containing the cut default values (and the topological variable names) is in PWGHF/Core/HFSelectorCuts.h (snippet below)



```
namespace hf_selcandidate_d0
{
DECLARE_SOA_COLUMN(IsSelD0, isSelD0, int);           //!
DECLARE_SOA_COLUMN(IsSelD0bar, isSelD0bar, int);     //!
DECLARE_SOA_COLUMN(IsRecoHFFlag, isRecoHFFlag, int); //!
DECLARE_SOA_COLUMN(IsRecoTopol, isRecoTopol, int);   //!
DECLARE_SOA_COLUMN(IsRecoCand, isRecoCand, int);     //!
DECLARE_SOA_COLUMN(IsRecoPID, isRecoPID, int);
} // namespace hf_selcandidate_d0
DECLARE_SOA_TABLE(HFSelD0Candidate, "AOD", "HFSELD0CAND", //!
                  hf_selcandidate_d0::IsSelD0,
                  hf_selcandidate_d0::IsSelD0bar,
                  hf_selcandidate_d0::IsRecoHFFlag,
                  hf_selcandidate_d0::IsRecoTopol,
                  hf_selcandidate_d0::IsRecoCand,
                  hf_selcandidate_d0::IsRecoPID);
```



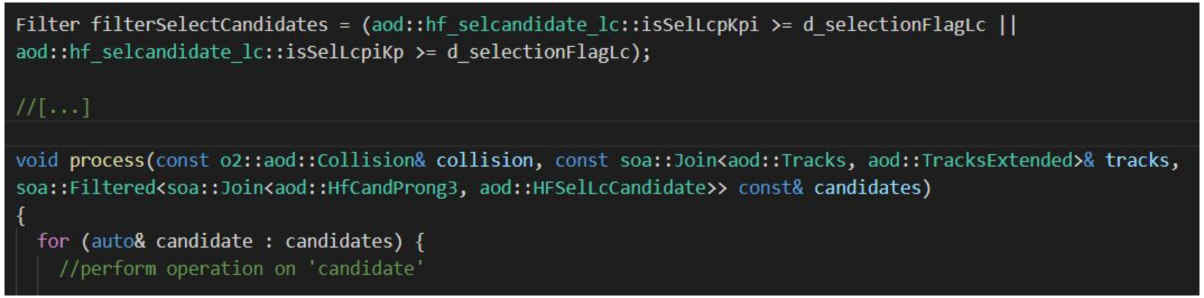
```
namespace hf_cuts_d0_topik
{
static constexpr int npTBins = 25;
static constexpr int nCutVars = 14;
// default values for the pT bin edges (can be used to configure histogram axis)
// offset by 1 from the bin numbers in cuts array
constexpr double pTBins[npTBins + 1] = {
  0,
  0.5,
  //[...]]
  50.0,
  100.0};
auto pTBins_v = std::vector<double>{pTBins, pTBins + npTBins + 1};

// default values for the cuts
constexpr double cuts[npTBins][nCutVars] = {{0.400, 350. * 1E-4, 0.8, 0.5, 0.5, 1000. * 1E-4, 1000. * 1E-4, -5000. * 1E-8
                                            {0.400, 350. * 1E-4, 0.8, 0.5, 0.5, 1000. * 1E-4, 1000. * 1E-4, -5000. * 1E-8
                                              //[...]
                                            {0.400, 300. * 1E-4, 1.0, 0.7, 0.7, 1000. * 1E-4, 1000. * 1E-4, 999999. * 1E-
                                            {0.400, 300. * 1E-4, 1.0, 0.6, 0.6, 1000. * 1E-4, 1000. * 1E-4, 999999. * 1E-

// column labels
static const std::vector<std::string> cutVarLabels = {"m", "DCA", "cos theta*", "pT K", "pT Pi", "d0K", "d0pi", "d0d0", "
} // namespace hf_cuts_d0_topik
```

CONNECTION WITH ANALYSIS TASK

Inside the analysis task:

- ○ Define a Filter which acts on the table filled by the selector, posing a condition to the selection flag(s)
- ○ In the process, subscribe to a 'filtered' table, which joins the candidate table and the selector output table

In this way, only the rows (-->candidates) which satisfy the filter condition are effectively considered and processed

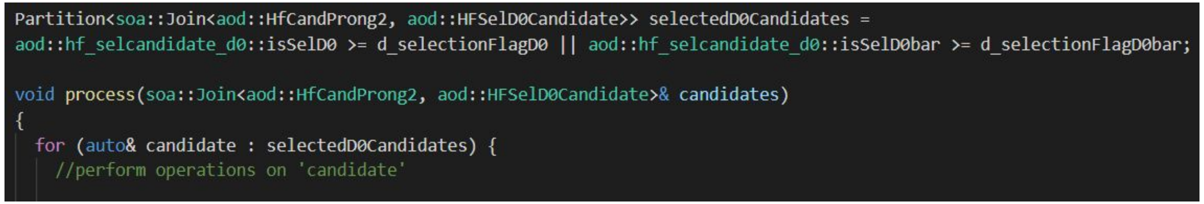


```
Filter filterSelectCandidates = (aod::hf_selcandidate_lc::isSelLcpKpi >= d_selectionFlagLc ||
aod::hf_selcandidate_lc::isSelLcpiKp >= d_selectionFlagLc);

//[...]

void process(const o2::aod::Collision& collision, const soa::Join<aod::Tracks, aod::TracksExtended>& tracks,
soa::Filtered<soa::Join<aod::HfCandProng3, aod::HFSelLcCandidate>> const& candidates)
{
  for (auto& candidate : candidates) {
    //perform operation on 'candidate'
```

Example for the Lc, which uses Filters - but the format is channel-independent

Alternatively, a Partition can be used

- The process subscribes to the whole candidate table, but the loop is performed only on the elements in the partition



```
Partition<soa::Join<aod::HfCandProng2, aod::HFSelD0Candidate>> selectedD0Candidates =
aod::hf_selcandidate_d0::isSelD0 >= d_selectionFlagD0 || aod::hf_selcandidate_d0::isSelD0bar >= d_selectionFlagD0bar;

void process(soa::Join<aod::HfCandProng2, aod::HFSelD0Candidate>& candidates)
{
  for (auto& candidate : selectedD0Candidates) {
    //perform operations on 'candidate'
```

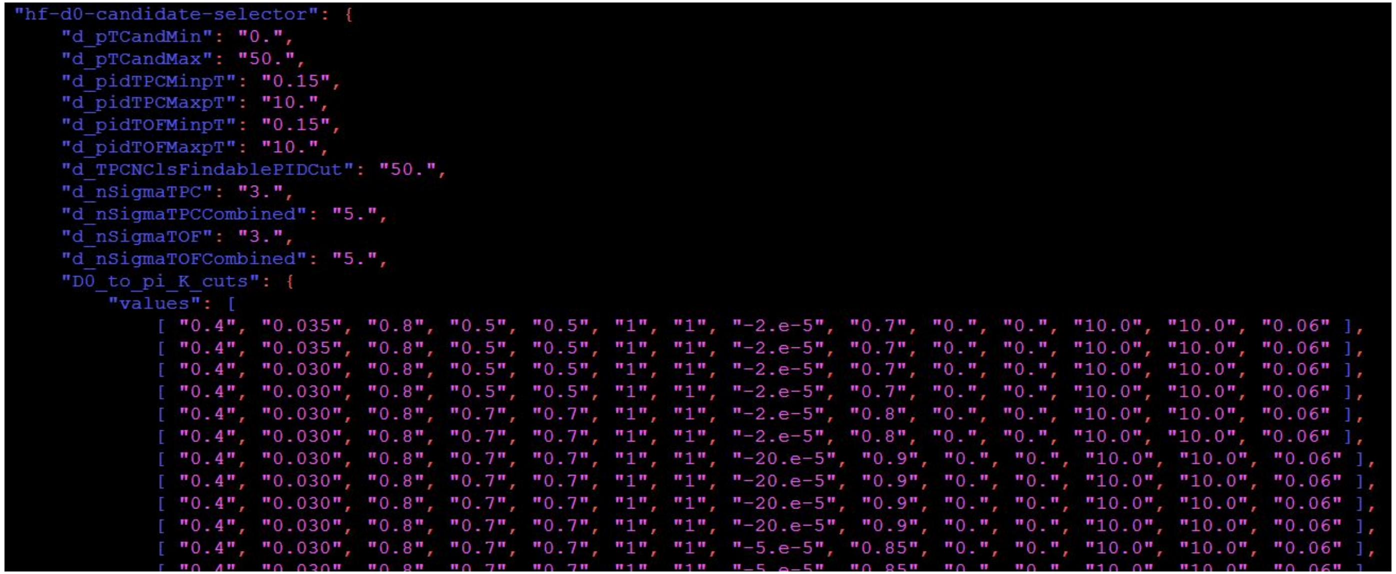
When possible, prefer this upfront selection w.r.t. 'classical' selection based e.g. on 'if' conditions

SETTING THE CUT VALUES

When launching the analysis workflow, the cut values can be modified by setting the related Configurable of the specific candidate selector task via json file, as following:

Shown here (not exhaustive):

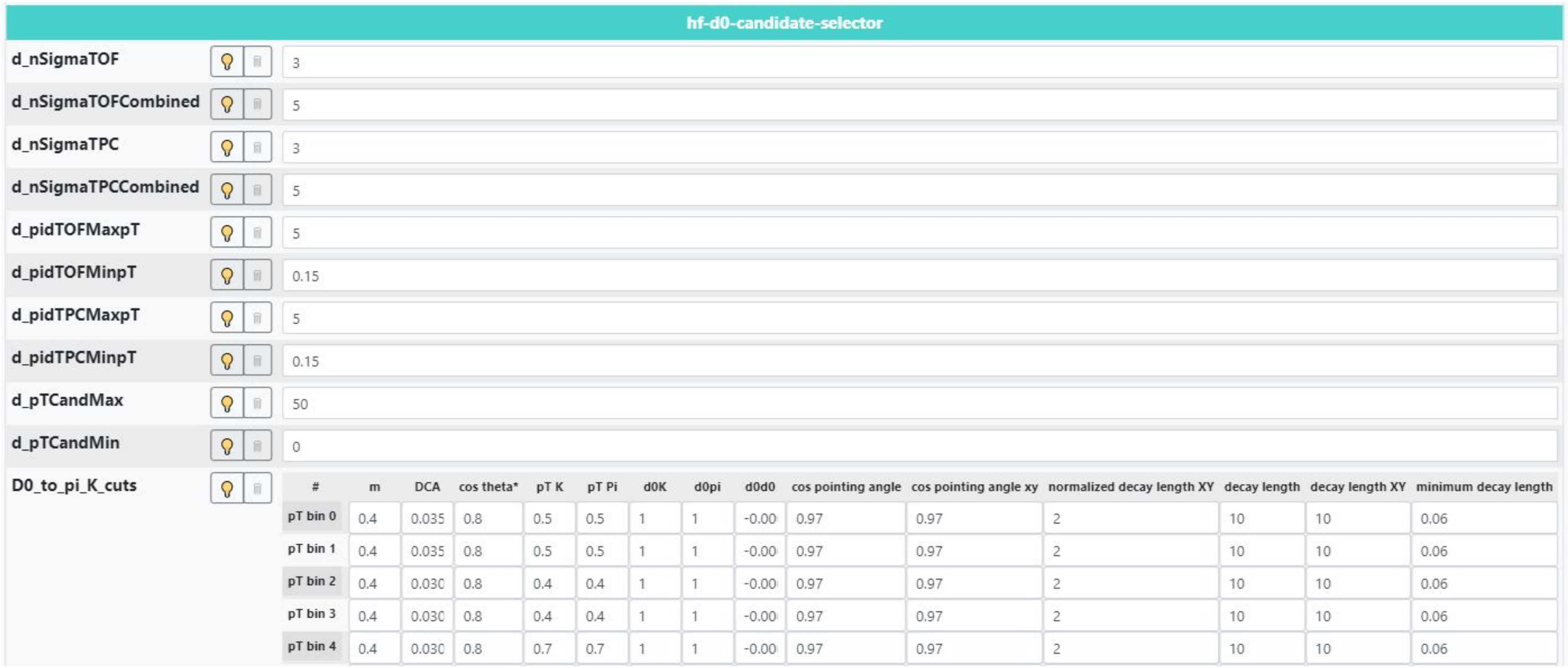
- pT ranges for TOF and TPC PID
- nSigma values for TOF and TPC PID
- Topological cut values (2D array)



```
"hf-d0-candidate-selector": {
    "d_pTCandMin": "0.",
    "d_pTCandMax": "50.",
    "d_pidTPCMinpT": "0.15",
    "d_pidTPCMaxpT": "10.",
    "d_pidTOFMinpT": "0.15",
    "d_pidTOFMaxpT": "10.",
    "d_TPCNClsFindablePIDCut": "50.",
    "d_nSigmaTPC": "3.",
    "d_nSigmaTPCCombined": "5.",
    "d_nSigmaTOF": "3.",
    "d_nSigmaTOFCombined": "5.",
    "D0_to_pi_K_cuts": {
        "values": [
            [ "0.4", "0.035", "0.8", "0.5", "0.5", "1", "1", "-2.e-5", "0.7", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.035", "0.8", "0.5", "0.5", "1", "1", "-2.e-5", "0.7", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.5", "0.5", "1", "1", "-2.e-5", "0.7", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.5", "0.5", "1", "1", "-2.e-5", "0.7", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-2.e-5", "0.8", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-2.e-5", "0.8", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-20.e-5", "0.9", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-20.e-5", "0.9", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-20.e-5", "0.9", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-20.e-5", "0.9", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-5.e-5", "0.85", "0.", "0.", "10.0", "10.0", "0.06" ],
            [ "0.4", "0.030", "0.8", "0.7", "0.7", "1", "1", "-5.e-5", "0.85", "0.", "0.", "10.0", "10.0", "0.06" ]
```

SETTING THE CUT VALUES

If running on AliHyperloop, the same Configurators can be set using the fields in the ‘configuration’ tab, of the selector wagon configuration



hf-d0-candidate-selector

d_nSigmaTOF	3
d_nSigmaTOFCombined	5
d_nSigmaTPC	3
d_nSigmaTPCCombined	5
d_pidTOFMaxpT	5
d_pidTOFMinpT	0.15
d_pidTPCMaxpT	5
d_pidTPCMinpT	0.15
d_pTCandMax	50
d_pTCandMin	0

D0_to_pi_K_cuts

#	m	DCA	cos theta*	pT K	pT Pi	d0K	d0pi	d0d0	cos pointing angle	cos pointing angle xy	normalized decay length XY	decay length	decay length XY	minimum decay length
pT bin 0	0.4	0.035	0.8	0.5	0.5	1	1	-0.00	0.97	0.97	2	10	10	0.06
pT bin 1	0.4	0.035	0.8	0.5	0.5	1	1	-0.00	0.97	0.97	2	10	10	0.06
pT bin 2	0.4	0.030	0.8	0.4	0.4	1	1	-0.00	0.97	0.97	2	10	10	0.06
pT bin 3	0.4	0.030	0.8	0.4	0.4	1	1	-0.00	0.97	0.97	2	10	10	0.06
pT bin 4	0.4	0.030	0.8	0.7	0.7	1	1	-0.00	0.97	0.97	2	10	10	0.06

MISSING ELEMENTS/POSSIBLE UPGRADES

Critical points of the current code:

… which act also as key points for the discussion

- Full definition of the selection strategy and variables
    - Some topological variable definition still missing for several channels
    - PID strategy also still under development
- The values of the topological variables, as currently defined, are subject to a possible bias
    - The primary vertex is not recalculated after discarding the daughter tracks
        - All variables connected to primary vertex position could be biased, especially in low-mult events
    - More a candidate creator-related point, but of course affects the selection
- It would be good to preserve the values used for the selection for bookkeeping (maybe storing them in the output file in some format?)
- What to do in case an analysis needs a specific track quality selection?
    - In AliPhysics, track selection could be completely configured in the same cut object, processed together with topological and PID selection
    - The track selection is currently performed directly in the skimmer and is quite rigid
        - e.g. for Run3, just check IsGlobalTrack status (currently under implementation)

MISSING ELEMENTS/POSSIBLE UPGRADES

Critical points of the current code:

… which act also as key points for the discussion

- How to integrate the selector framework (based on rectangular cuts) with a ML-based selection (based on models)? Another way to go could be employing a Bayesian-based selection (tried for PID in Run2, combining the info from the different PID detector)
  - Is it possible to build a single, general structure that can deal with all the above approaches?
- How to apply multiple selections in the same analysis run? This was typically done for systematic studies in the past, where looser/tighter cut sets were probed to check the selection stability
  - Not possible to do with current implementation: cannot activate two selectors with different configuration, because they would produce the same table
  - Also not favoured due to too large resource consumption (especially in Pb-Pb)
    - Possible strategies:
      - Apply loosest selection, store the trees and apply tighter cuts offline on the trees
      - Substitute the selection flag with a bitmap, where each bit corresponds to a certaing selection (anyway, the selector need to support multiple input selections)

→ **Any further input is welcome!**