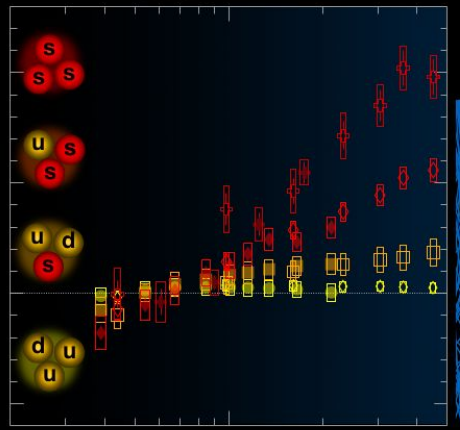
Strangeness hands-on

O2 Analysis Tutorial 16/10/2024*

Romain Schotter

Austrian Academy of Sciences, Austria

romain.schotter@cern.ch



* based on the hands-on session from the O2AT 08/11/2023 from Roman Nepeivoda

Outline

- In this hands-on session, we will focus on the analysis of cascades in Pb-Pb

- You will learn how to
  - Produce derived tables
  - Structure your analysis task
  - Reconstruct cascades
  - Apply (TPC and TOF) PID, topological and kinematic selections
  - Access MC information of the cascades

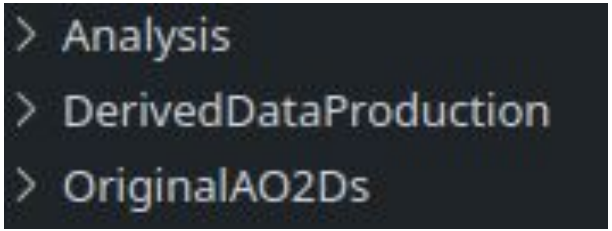
More information on the strangeness tables can be found in Tuesday's presentation

Romain Schotter (romain.schotter@cern.ch)

Preparation

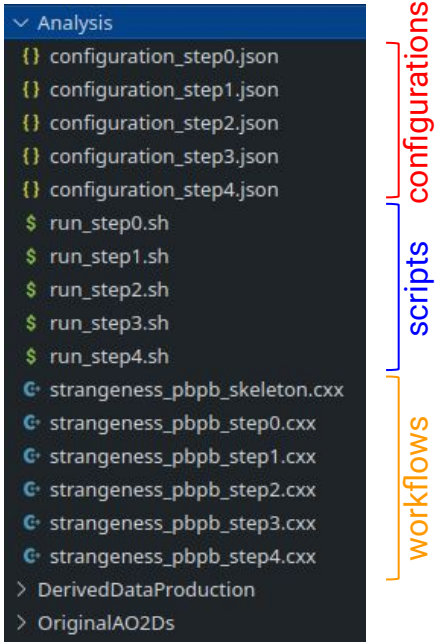
- All the necessary scripts and workflows for this tutorial can be found in the [O2Physics/Tutorials/PWGLF/Strangeness/PbPb/]

- The json files (and more!) can be found here: https://cernbox.cern.ch/s/2hFoAUY3wPolFZM



Preparation

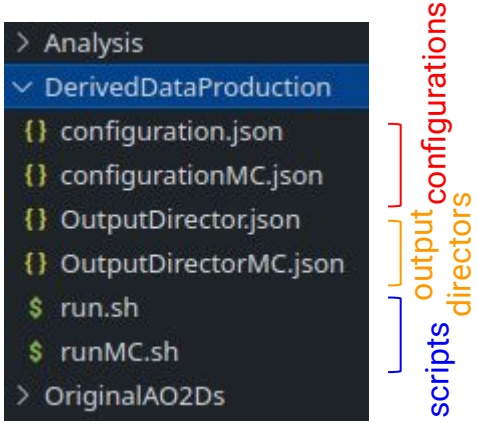
- All the necessary scripts and workflows for this tutorial can be found in the [O2Physics/Tutorials/PWGLF/Strangeness/PbPb/]
- The json files (and more!) can be found here: https://cernbox.cern.ch/s/2hFoAUY3wPolFZM



All files (scripts, workflows, configurations) for running the analysis task are stored in the **Analysis** folder

Preparation

- All the necessary scripts and workflows for this tutorial can be found in the [O2Physics/Tutorials/PWGLF/Strangeness/PbPb/]
- The json files (and more!) can be found here: https://cernbox.cern.ch/s/2hFoAUY3wPolFZM



All files (scripts, output directors, configurations) for producing the derived data are stored in the **DerivedDataProduction** folder

The original data are stored in the **OriginalAO2Ds** folder

Preparation

- All the necessary scripts and workflows for this tutorial can be found in the O2Physics/Tutorials/PWGLF/Strangeness/PbPb/
- The json files (and more!) can be found here: https://cernbox.cern.ch/s/2hFoAUY3wPolFZM
- Create a folder to store the original data
- Download original data folder (~4.8 GB):

```
rsync -r -u --progress USERNAME@lxplus.cern.ch:/afs/cern.ch/user/r/rschotte/cernbox/O2AT_2024/PbPb/OriginalAO2Ds .
```

- Enter your lxplus password (if required)
- Alternatively, download the file from cernbox manually
- Or use the script from cernbox to directly download the original data file from the grid

Preparation: derived data for LF

- **Derived data** are (*derived*) AO2D.root files produced with a task that creates given tables by processing other (*original*) AO2D.root files

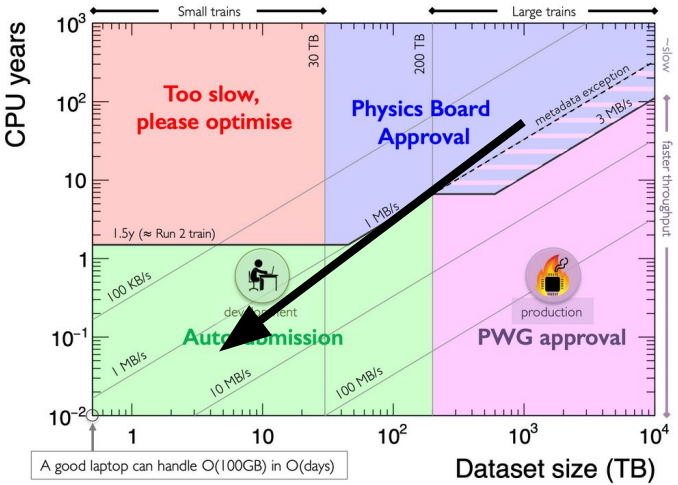
- Only information needed for your analysis are stored in derived data

  → reduce the size of AO2D.root files
  → **speed up the execution** by skipping part of the workflows of your analysis (typically the computational intensive parts, ie track propagation, etc)

- Two types of derived data:

  - **Self contained:** contain all the information needed for your analysis

  - **Linked:** contain additional information wrt the original AO2D.root files and hence require access to the parent AO2D.root files



Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

```
OPTION="-b --configuration json://configuration.json"

o2-analysis-pid-tof-base ${OPTION} |
o2-analysis-event-selection ${OPTION} |
o2-analysis-lf-lambdakzerobuilder ${OPTION} |
o2-analysis-lf-cascadebuilder ${OPTION} |
o2-analysis-multiplicity-table ${OPTION} |
o2-analysis-centrality-table ${OPTION} |
o2-analysis-lf-epvector ${OPTION} |
o2-analysis-centrality-qa ${OPTION} |
o2-analysis-ud-sgcand-producer ${OPTION} |
o2-analysis-timestamp ${OPTION} |
o2-analysis-ft0-corrected-table ${OPTION} |
o2-analysis-track-propagation ${OPTION} |
o2-analysis-pid-tpc-base ${OPTION} |
o2-analysis-pid-tpc ${OPTION} |
o2-analysis-trackselection ${OPTION} |
o2-analysis-pid-tof-full ${OPTION} |
o2-analysis-pid-tof-beta ${OPTION} |
o2-analysis-lf-strangederivedbuilder ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json > "$LOGFILE" 2>&1
```

Text file containing the path to the original AO2D.root file

JSON file specifying which table to save on disk

Romain Schotter (romain.schotter@cern.ch)

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

```
OPTION="-b --configuration json://configuration.json"

o2-analysis-pid-tof-base ${OPTION} |
o2-analysis-event-selection ${OPTION} |
o2-analysis-lf-lambdakzerobuilder ${OPTION} |
o2-analysis-lf-cascadebuilder ${OPTION} |
o2-analysis-multiplicity-table ${OPTION} |
o2-analysis-centrality-table ${OPTION} |
o2-analysis-lf-epvector ${OPTION} |
o2-analysis-centrality-qa ${OPTION} |
o2-analysis-ud-sgcand-producer ${OPTION} |
o2-analysis-timestamp ${OPTION} |
o2-analysis-ft0-corrected-table ${OPTION} |
o2-analysis-track-propagation ${OPTION} |
o2-analysis-pid-tpc-base ${OPTION} |
o2-analysis-pid-tpc ${OPTION} |
o2-analysis-trackselection ${OPTION} |
o2-analysis-pid-tof-full ${OPTION} |
o2-analysis-pid-tof-beta ${OPTION} |
o2-analysis-lf-strangederivedbuilder ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json > "$LOGFILE" 2>&1
```

Text file containing the path to the original AO2D.root file

JSON file specifying which table to save on disk

Workflow with all the helper tasks

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

```
#!/bin/bash
# log file where the terminal output will be saved
STEP="deriveddata"
LOGFILE="log-${STEP}.txt"

#directory of this script
DIR_THIS=$PWD

OPTION="-b --configuration json://configuration.json"

o2-analysis-pid-tof-base ${OPTION} |
o2-analysis-event-selection ${OPTION} |
o2-analysis-lf-lambdakzerobuilder ${OPTION} |
o2-analysis-lf-cascadebuilder ${OPTION} |
o2-analysis-multiplicity-table ${OPTION} |
o2-analysis-centrality-table ${OPTION} |
o2-analysis-lf-epvector ${OPTION} |
o2-analysis-centrality-qa ${OPTION} |
o2-analysis-ud-sgcand-producer ${OPTION} |
o2-analysis-timestamp ${OPTION} |
o2-analysis-ft0-corrected-table ${OPTION} |
o2-analysis-track-propagation ${OPTION} |
o2-analysis-pid-tpc-base ${OPTION} |
o2-analysis-pid-tpc ${OPTION} |
o2-analysis-trackselection ${OPTION} |
o2-analysis-pid-tof-full ${OPTION} |
o2-analysis-pid-tof-beta ${OPTION} |
o2-analysis-lf-strangederivedbuilder ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json > "$LOGFILE" 2>&1
```

Text file containing the path to the original AO2D.root file

JSON file specifying which table to save on disk

The whole log output is transferred into the **log-${STEP}.txt** file

Workflow with all the helper tasks

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

```
#!/bin/bash
# log file where the terminal output will be saved
STEP="deriveddata"
LOGFILE="log-${STEP}.txt"

#directory of this script
DIR_THIS=$PWD

OPTION="-b --configuration json://configuration.json"

o2-analysis-pid-tof-base ${OPTION} |
o2-analysis-event-selection ${OPTION} |
o2-analysis-lf-lambdakzerobuilder ${OPTION} |
o2-analysis-lf-cascadebuilder ${OPTION} |
o2-analysis-multiplicity-table ${OPTION} |
o2-analysis-centrality-table ${OPTION} |
o2-analysis-lf-epvector ${OPTION} |
o2-analysis-centrality-qa ${OPTION} |
o2-analysis-ud-sgcand-producer ${OPTION} |
o2-analysis-timestamp ${OPTION} |
o2-analysis-ft0-corrected-table ${OPTION} |
o2-analysis-track-propagation ${OPTION} |
o2-analysis-pid-tpc-base ${OPTION} |
o2-analysis-pid-tpc ${OPTION} |
o2-analysis-trackselection ${OPTION} |
o2-analysis-pid-tof-full ${OPTION} |
o2-analysis-pid-tof-beta ${OPTION} |
o2-analysis-lf-strangederivedbuilder ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json > "$LOGFILE" 2>&1

# report status
rc=$?
if [ $rc -eq ]; then
    echo "No problems!"
    mkdir -p ${DIR_THIS}/results/${STEP}
    mv AnalysisResults.root ${DIR_THIS}/results/${STEP}/AnalysisResults.root
    mv AO2D.root ${DIR_THIS}/results/${STEP}/AO2D.root
    mv dpl-config.json ${DIR_THIS}/jsonConfigs/${STEP}.json
else
    echo "Error: Exit code ${rc}"
    echo "Check the log file ${LOGFILE}"
    exit rc
fi
```

Text file containing the path to the original AO2D.root file

JSON file specifying which table to save on disk

The whole log output is transferred into the **log-${STEP}.txt** file

Workflow with all the helper tasks

If the task finishes successfully, **AnalysisResults.root**, **AO2D.root**, and **json files** are moved to the `results` folder

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

```
#!/bin/bash
# log file where the terminal output will be saved
STEP="deriveddata"
LOGFILE="log-${STEP}.txt"

#directory of this script
DIR_THIS=$PWD

OPTION="-b --configuration json://configuration.json"

o2-analysis-pid-tof-base ${OPTION} |
o2-analysis-event-selection ${OPTION} |
o2-analysis-lf-lambdakzerobuilder ${OPTION} |
o2-analysis-lf-cascadebuilder ${OPTION} |
o2-analysis-multiplicity-table ${OPTION} |
o2-analysis-centrality-table ${OPTION} |
o2-analysis-lf-epvector ${OPTION} |
o2-analysis-centrality-qa ${OPTION} |
o2-analysis-ud-sgcand-producer ${OPTION} |
o2-analysis-timestamp ${OPTION} |
o2-analysis-ft0-corrected-table ${OPTION} |
o2-analysis-track-propagation ${OPTION} |
o2-analysis-pid-tpc-base ${OPTION} |
o2-analysis-pid-tpc ${OPTION} |
o2-analysis-trackselection ${OPTION} |
o2-analysis-pid-tof-full ${OPTION} |
o2-analysis-pid-tof-beta ${OPTION} |
o2-analysis-lf-strangederivedbuilder ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json > "$LOGFILE" 2>&1

# report status
rc=$?
if [ $rc -eq ]; then
    echo "No problems!"
    mkdir -p ${DIR_THIS}/results/${STEP}
    mv AnalysisResults.root ${DIR_THIS}/results/${STEP}/AnalysisResults.root
    mv AO2D.root ${DIR_THIS}/results/${STEP}/AO2D.root
    mv dpl-config.json ${DIR_THIS}/jsonConfigs/${STEP}.json
else
    echo "Error: Exit code ${rc}"
    echo "Check the log file ${LOGFILE}"
    exit rc
fi
```

All shell running script follow the same structure in the tutorial!

Text file containing the path to the original AO2D.root file

JSON file specifying which table to save on disk

The whole log output is transferred into the **log-${STEP}.txt** file

Workflow with all the helper tasks

If the task finishes successfully, **AnalysisResults.root**, **AO2D.root**, and **json files** are moved to the `results` folder

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

→ `bash run.sh`

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

→ `bash run.sh`

`No problems!` must be printed

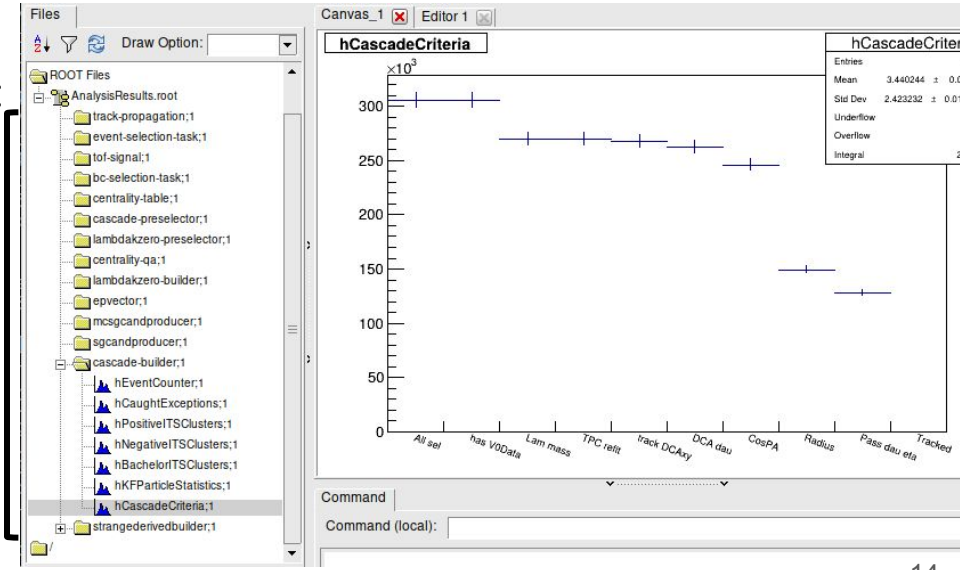
All output are contained in the **results** folder:

- `AnalysisResults.root`

Standard output

It contains the output of each task

Each task (including the intermediate tasks) in the workflow can create and fill **histograms**, which are **useful for QC**



Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

→ `bash run.sh`

`No problems!` must be printed

All output are contained in the **results** folder:

- `AnalysisResults.root`
- `AO2D.root`

Specific output to derived data production!

This is the ***derived*** **`AO2D.root` file**

Each derived table specified in the `OutputDirector.json` file has been produced and stored in this file

Preparation: how to produce derived data?

- **Locally,** run the workflows to produce the derived tables and specify the ones you want to save in the `OutputDirector.json` file

  → `bash run.sh`

  `No problems!` must be printed

  All output are contained in the **results** folder:

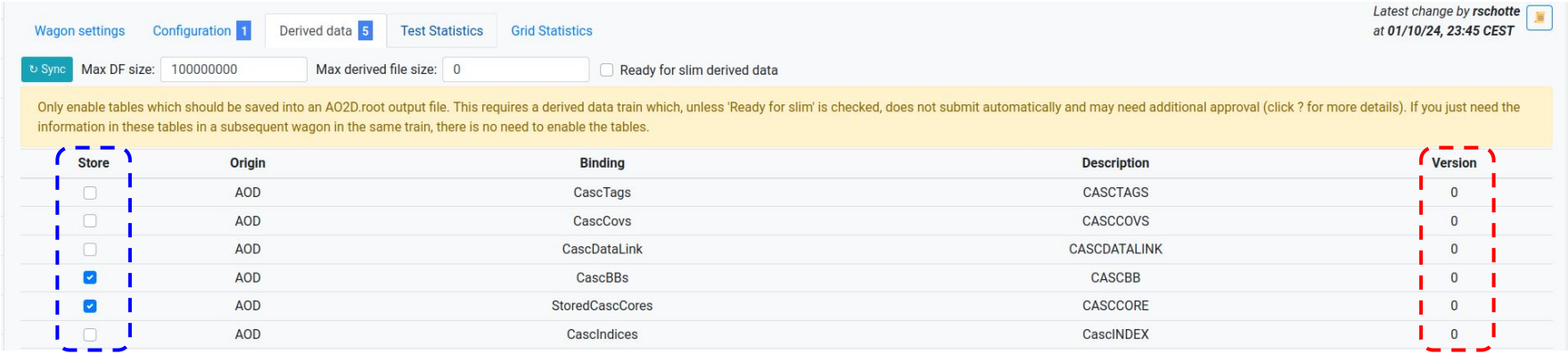
  - `AnalysisResults.root`
  - `AO2D.root`
  - `dpl-config.json`

  This is **a copy of the `configuration.json` file**

Romain Schotter (romain.schotter@cern.ch)

Preparation: how to produce derived data?

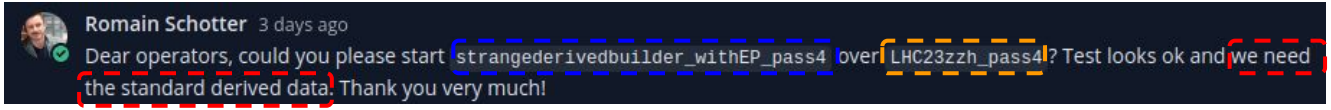
- **On Hyperloop,**
  - Select the tables you want to store/save on disk



Be careful about the table version!

  - Test your train on your desired dataset

  - Ask train operator on the O2 Hyperloop Operation channel to submit your train

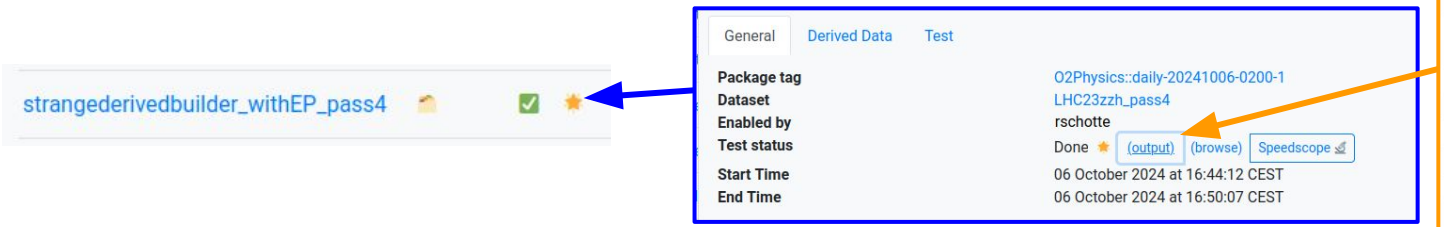


Key information:
- main wagon,
- dataset,
- derived data

Preparation: how to produce derived data?

- **Locally** as if it was on Hyperloop
  - Reproducing a train run on a local machine documentation page
  - Download the `run_train.sh` file

  - Go on Hyperloop and download the `full_config.json` file,
    → contains both the `configuration.json` and `OutputDirector.json` files



  - Run the workflows to produce the derived tables using the same `input_data.txt` file

```
bash run_train.sh --skip-perf
```

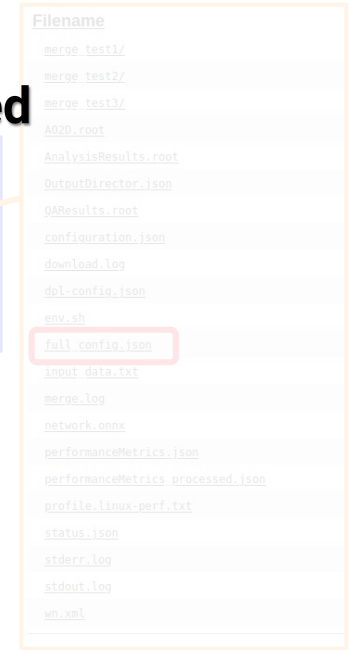
```
O2 exited with 0
```

Execution went fine!

Preparation: how to produce derived data?

- **Locally** as if it was on Hyperloop
  - Reproducing a train run on a local machine documentation page
  - Download the `run_train.sh` file
  - Go on Hyperloop and download the `full_config.json` file,
    → contains both the `configuration.json` and `OutputDirector.json` files

This can be used for every kind of train! (derived data production, analysis tasks,...)



  - Run the workflows to produce the derived tables using the same `input_data.txt` file

```
bash run_train.sh --skip-perf
```

```
O2 exited with 0
```

Execution went fine!

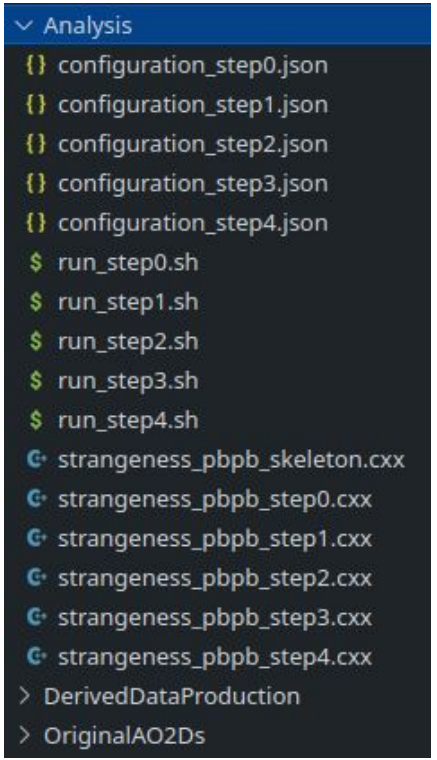
Preparation: running an analysis task

- Once the derived data have been produced, we can start writing the analysis task

- The idea is to start with the `strangeness_pbpb_skeleton.cxx` file and improve it up to step4

- Checkpoints for each step (X) are here for your convenience (`strangeness_pbpb_stepX.cxx`), with the running shell script (`run_stepX.sh`) and the corresponding configurations (`configuration_stepX.json`)

- Let's have a look at `strangeness_pbpb_skeleton.cxx`



Preparation: running an analysis task

`strangeness_pbpb_skeleton.cxx` loops over all events, selects the ones with |z| < 10 cm and `sel8` flag (FT0 vertex compatible with FT0-A and FT0-C timing info), fills an histogram with the *z*-vertex position

```cpp
#include "Framework/runDataProcessing.h"
#include "Framework/AnalysisTask.h"
#include "Common/DataModel/EventSelection.h"
#include "PWGLF/DataModel/LFStrangenessTables.h"
#include "Framework/O2DatabasePDGPlugin.h"

using namespace o2;
using namespace o2::framework;
using namespace o2::framework::expressions;

struct strangeness_pbpb_tutorial {
  // Histograms are defined with HistogramRegistry
  HistogramRegistry rEventSelection{"eventSelection", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};

  // Configurable for histograms
  Configurable<int> nBins{"nBins", 100, "N bins in all histos"};

  // Configurable for event selection
  Configurable<float> cutzvertex{"cutzvertex", 10.0f, "Accepted z-vertex range (cm)"};

  // PDG data base
  Service<o2::framework::O2DatabasePDG> pdgDB;

  void init(InitContext const&)
  {
    // Axes
    AxisSpec vertexZAxis = {nBins, -15., 15., "vrtx_{Z} [cm]"};

    // Histograms
    // Event selection
    rEventSelection.add("hVertexZRec", "hVertexZRec", {HistType::kTH1F, {vertexZAxis}});
  }
```

Include required **headers**

Histogram registries and **Configurables** are declared before the **init** function

Create axis and add needed **histograms** to the registries

Preparation: running an analysis task

```
51
52    // Defining filters for events (event selection)
53    // Processed events will be already fulfilling the event selection requirements
54    Filter eventFilter = (o2::aod::evsel::sel8 == true);
55    Filter posZFilter = (nabs(o2::aod::collision::posZ) < cutzvertex);
56
57    void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision)
58    {
59      // Fill the event counter
60      rEventSelection.fill(HIST("hVertexZRec"), collision.posZ());
61    }
62  };
63
64  WorkflowSpec defineDataProcessing(ConfigContext const& cfgc)
65  {
66    return WorkflowSpec{
67      adaptAnalysisTask<strangeness_pbpb_tutorial>(cfgc)};
68  }
```

Define **Filters** for basic event selection

Subscribe to an iterator of a table joining **aod::StraCollisions** (=basic collision properties) **and aod::StraEvSels** (= all necessary event selection variables) to loop over the selected collisions

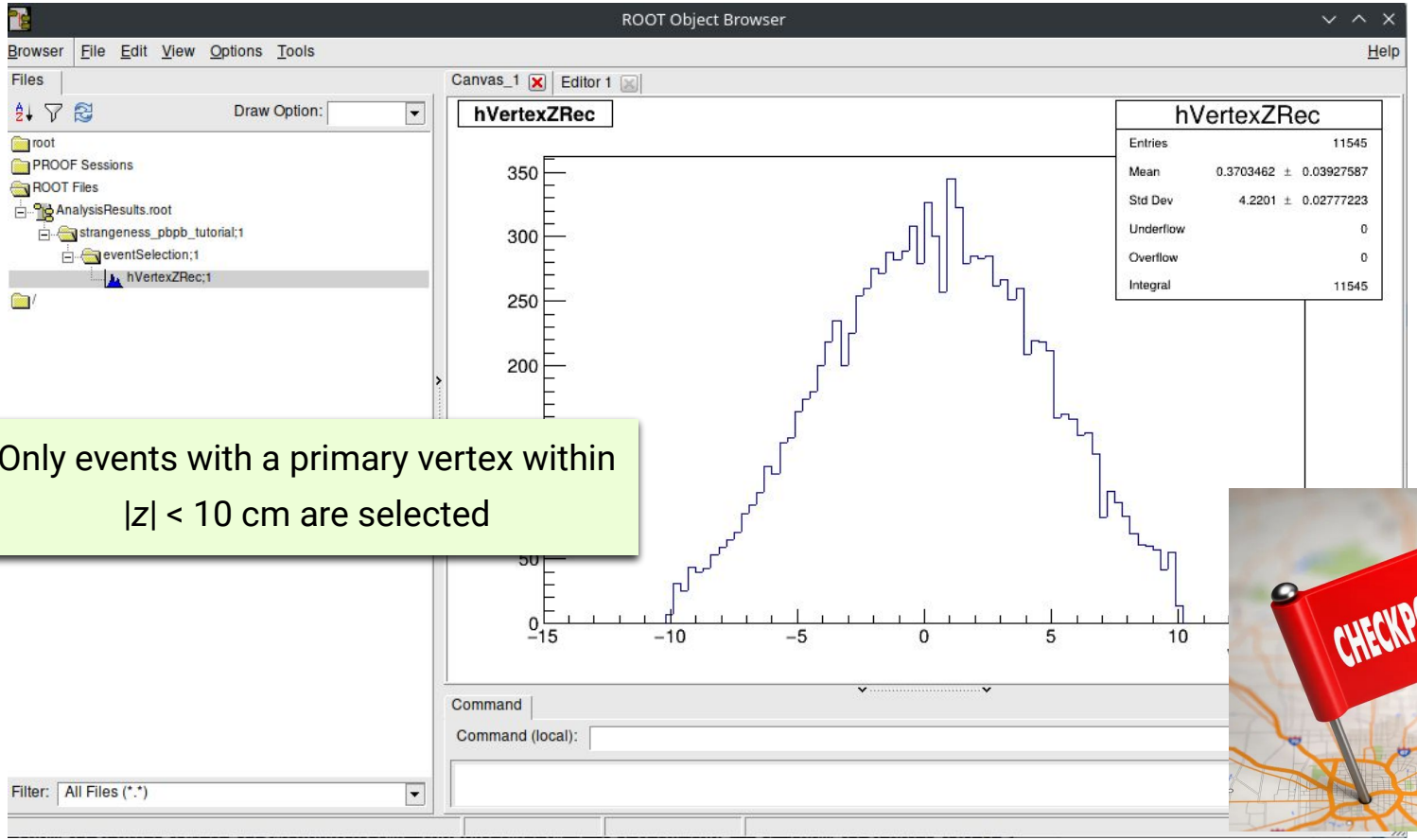
Because of the iterator, the **process()** function is called once for each selected event and fills the vertex *z*-position histogram

Let's run the analysis task:

```
bash run_skeleton.sh
```

```
OPTION="-b --configuration json://configuration_skeleton.json"
o2-analysistutorial-lf-strangeness-pbpb-skeleton ${OPTION} --aod-file @input_data.txt
```

Preparation: running an analysis task



Only events with a primary vertex within $|z| < 10$ cm are selected

Step 0: looping over cascades

- Starting from `strangeness_pbpb_skeleton.cxx`, we want to loop over all cascades in the selected events and fill a invariant mass histogram

- ***Instructions:***
  - Create two **HistogramRegistry** named “**xi**” for Xi and “**omega**” for Omega baryons
  - Subscribe to the cascade tables (have a look at `LFStrangenessTables.h`)
  - For each event, loop over all cascades,
  - For each mass hypothesis, fill a histogram with the invariant mass
  - Store the histograms in the corresponding **HistogramRegistry**

- ***Hints:***
  1. for the invariant mass histograms, you can use the following binning:
     - Xi: 100 bins, from 1.28 GeV/c2 to 1.36 GeV/c2
     - Omega: 100 bins, from 1.63 GeV/c2 to 1.70 GeV/c2
  2. for the cascade info, look at the **aod::CascCores** table

In case you are facing some difficulties, please do not hesitate to ask questions!

Step 0: looping over cascades

```
struct strangeness_pbpb_tutorial {
  // Histograms are defined with HistogramRegistry
  HistogramRegistry rEventSelection{"eventSelection", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};
  HistogramRegistry rXi{"xi", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};
  HistogramRegistry rOmega{"omega", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};
```

Create a “**xi**” and “**omega**” **HistogramRegistry**

```
  void init(InitContext const&)
  {
    // Axes
    AxisSpec XiMassAxis = {100, 1.28f, 1.36f, "#it{M}_{inv} [GeV/#it{c}^{2}]"};
    AxisSpec OmegaMassAxis = {100, 1.63f, 1.7f, "#it{M}_{inv} [GeV/#it{c}^{2}]"};
    AxisSpec vertexZAxis = {nBins, -15., 15., "vrtx_{Z} [cm]"};

    // Histograms
    // Event selection
    rEventSelection.add("hVertexZRec", "hVertexZRec", {HistType::kTH1F, {vertexZAxis}});

    // Xi/Omega reconstruction
    rXi.add("hMassXi", "hMassXi", {HistType::kTH1F, {XiMassAxis}});

    rOmega.add("hMassOmega", "hMassOmega", {HistType::kTH1F, {OmegaMassAxis}});
  }
```

Create dedicated axes for Xi and Omega

Create invariant mass histograms for Xi and Omega and store them in their corresponding **HistogramRegistry**

Step 0: looping over cascades

```cpp
void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             aod::CascCores const& Cascades)
{
  // Fill the event counter
  rEventSelection.fill(HIST("hVertexZRec"), collision.posZ());

  // Cascades
  for (const auto& casc : Cascades) {
    rXi.fill(HIST("hMassXi"), casc.mXi());
    rOmega.fill(HIST("hMassOmega"), casc.mOmega());
  }
}
```

Subscribe to the cascade tables (**aod::CascCores**)

For each event, loop over all cascades

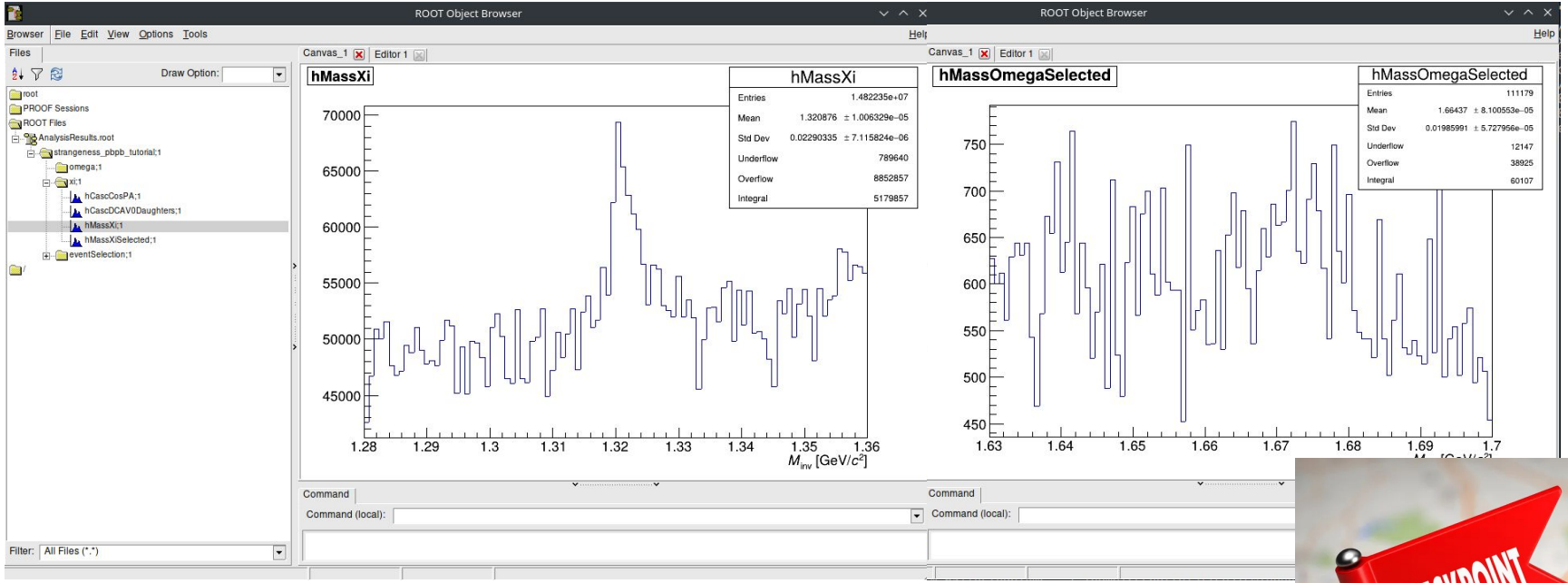
For each mass hypothesis, fill a histogram with the invariant mass

Require an extra task to enable the use of table with extended columns like **aod::CascCores**

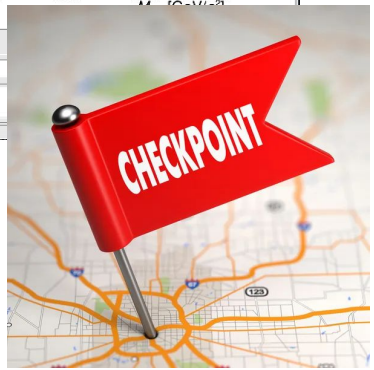
```
OPTION="-b --configuration json://configuration_step0.json"

o2-analysis-lf-cascadespawner ${OPTION} | o2-analysistutorial-lf-strangeness-pbpb-step0 ${OPTION} --aod-file @input_data.txt
```

```cpp
// extended table with expression columns that can be used as arguments of dynamic columns
DECLARE_SOA_EXTENDED_TABLE_USER(CascCores, StoredCascCores, "CascDATAEXT", //!
                                cascdataext::PxLambda, cascdataext::PyLambda, cascdataext::PzLambda, cascdataext::Pt, cascdataext::P, cascdataext::Eta,
                                cascdataext::Phi);
```

Step 0: looping over cascades



Invariant mass peak is already visible for Ξ with the derived data producer **pre-selections**, while no peak for Ω



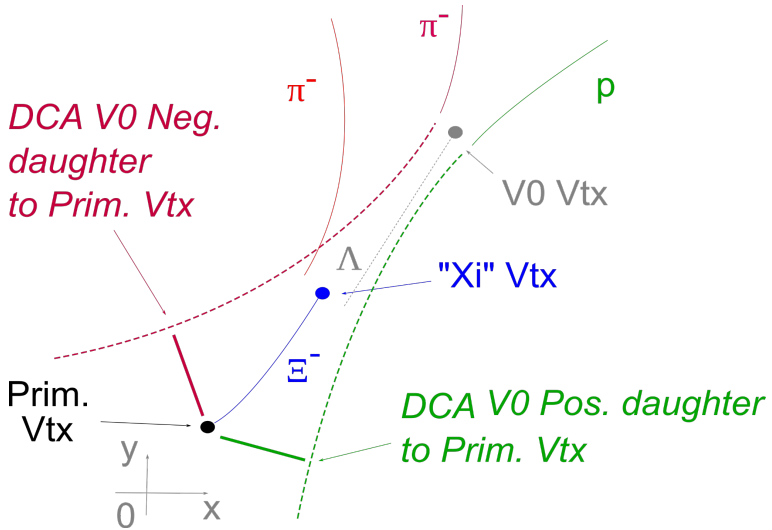
Step 1: apply topological selections

- We would like now to select more finely our cascade candidates based on their decay kinematics and topology

$$\begin{cases} \Lambda \to \mathrm{p}\pi^- \\ \overline{\Lambda} \to \overline{\mathrm{p}}\pi^+ \end{cases}$$ c τ = 7.89 cm B.R. 63.9%

$$\begin{cases} \Xi^- \to \Lambda\pi^- \\ \overline{\Xi}^+ \to \overline{\Lambda}\pi^+ \end{cases}$$ c τ = 4.91 cm B.R. 99.9%

$$\begin{cases} \Omega^- \to \Lambda\mathrm{K}^- \\ \overline{\Omega}^+ \to \overline{\Lambda}\mathrm{K}^+ \end{cases}$$ c τ = 2.461 cm B.R. 67.8%

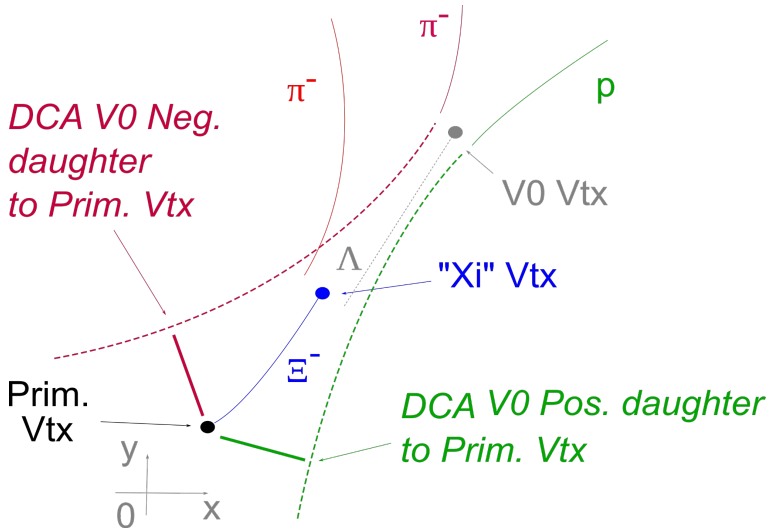


Step 1: apply topological selections

- We would like now to select more finely our candidates based on their decay kinematics and topology

V0 mass window	< 0.008 GeV/c2
DCA pos./neg. to prim. vtx	> 0.06 cm
DCA V0 to prim vtx	> 1.0 cm
DCA between V0. daughters	< 1.0 cm
V0 decay radius	> 1.2 cm
V0 cos of pointing angle	> 0.97

Competing mass rejection (only for Ω)	> 0.008 GeV/c2
DCA bach. to prim. vtx	> 0.06 cm
DCA between casc. daughters	< 1.0 cm
Casc. decay radius	> 0.5 cm
Casc. cos of pointing angle	> 0.998



Step 1: apply topological selections

- We would like now to select more finely our cascade candidates based on their decay kinematics and topology

- ***Instructions:***
  - Add a **Configurable** for each cascade selection (12)
  - Select the candidates passing the topological selections
  - For each mass hypothesis, fill a histogram with the invariant mass after cascade selection

- ***Hints:***
  1. Have a look at `LFStrangenessTables.h` to find the columns storing the selection variables
  2. you can use **Filter** to select the candidates but it cannot cut on dynamic columns
  3. for the cascade info, look at the **aod::Casccores** table

In case you are facing some difficulties, please do not hesitate to ask questions!

Step 1: apply topological selections

```
// Configurable parameters for cascade selection
Configurable<double> cascadesetting_cospa{"cascadesetting_cospa", 0.98, "Casc CosPA"};
Configurable<double> cascadesetting_v0cospa{"cascadesetting_v0cospa", 0.97, "V0 CosPA"};
Configurable<float> cascadesetting_dcacascdau{"cascadesetting_dcacascdau", 1.0, "DCA cascade daughters"};
Configurable<float> cascadesetting_dcav0dau{"cascadesetting_dcav0dau", 1.0, "DCA v0 daughters"};
Configurable<float> cascadesetting_dcabachtopv{"cascadesetting_dcabachtopv", 0.06, "DCA bachelor to PV"};
Configurable<float> cascadesetting_dcapostopv{"cascadesetting_dcapostopv", 0.06, "DCA positive to PV"};
Configurable<float> cascadesetting_dcanegtopv{"cascadesetting_dcanegtopv", 0.06, "DCA negative to PV"};
Configurable<float> cascadesetting_mindcav0topv{"cascadesetting_mindcav0topv", 0.01, "minimum V0 DCA to PV"};
Configurable<float> cascadesetting_cascradius{"cascadesetting_cascradius", 0.5, "cascradius"};
Configurable<float> cascadesetting_v0radius{"cascadesetting_v0radius", 1.2, "v0radius"};
Configurable<float> cascadesetting_v0masswindow{"cascadesetting_v0masswindow", 0.01, "v0 mass window"};
Configurable<float> cascadesetting_competingmassrej{"cascadesetting_competingmassrej", 0.008, "Competing mass rejection"};
```

Add **Configurable** for each cascade selection

```
// Filters on Cascades
// Cannot filter on dynamic columns
Filter preFilterCascades = (aod::cascdata::dcaV0daughters < cascadesetting_dcav0dau &&
                            nabs(aod::cascdata::dcapostopv) > cascadesetting_dcapostopv &&
                            nabs(aod::cascdata::dcanegtopv) > cascadesetting_dcanegtopv &&
                            nabs(aod::cascdata::dcabachtopv) > cascadesetting_dcabachtopv &&
                            aod::cascdata::dcacascdaughters < cascadesetting_dcacascdau);

void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             soa::Filtered<aod::CascCores> const& Cascades)
```

Filter on **regular** columns

Step 1: apply topological selections

```
// Cut on dynamic columns
if (casc.casccosPA(collision.posX(), collision.posY(), collision.posZ()) < cascadesetting_cospa)
  continue;
if (casc.v0cosPA(collision.posX(), collision.posY(), collision.posZ()) < cascadesetting_v0cospa)
  continue;
if (TMath::Abs(casc.mLambda() - pdgDB->Mass(3122)) > cascadesetting_v0masswindow)
  continue;
if (casc.dcav0topv(collision.posX(), collision.posY(), collision.posZ()) < cascadesetting_mindcav0topv)
  continue;
if (casc.cascradius() < cascadesetting_cascradius)
  continue;
if (casc.v0radius() < cascadesetting_v0radius)
  continue;
```

Cut on **dynamic** columns in the loop

```
// Fill histograms! (if possible)
rXi.fill(HIST("hMassXiSelected"), casc.mXi());

if (TMath::Abs(casc.mXi() - pdgDB->Mass(3312)) > cascadesetting_competingmassrej) { // competing mass rejection, only in case of Omega
  rOmega.fill(HIST("hMassOmegaSelected"), casc.mOmega());
```

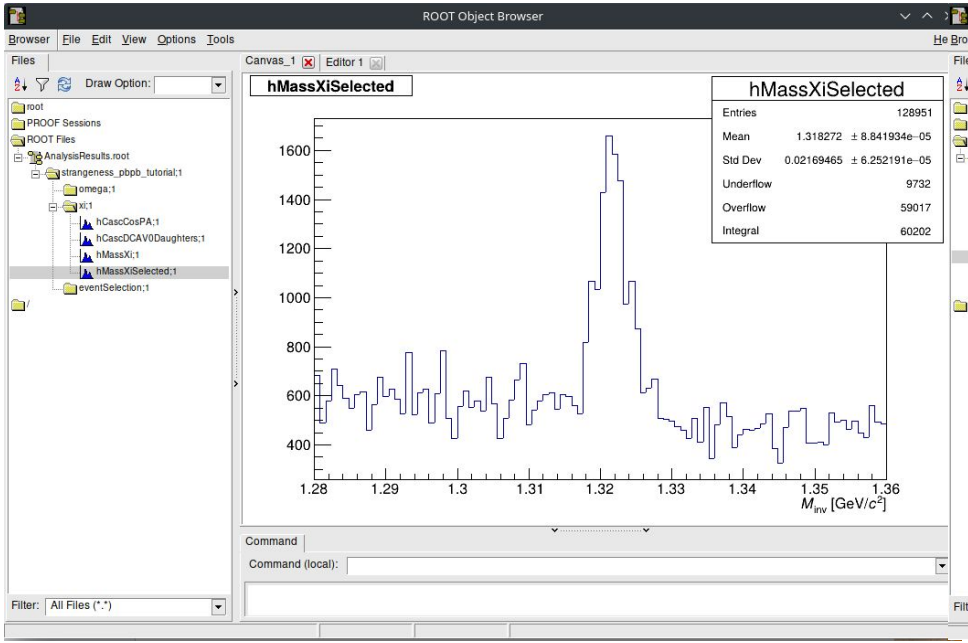
Fill invariant mass histograms <u>after</u> the topological selections

```
"strangeness_pbpb_tutorial": {
    "nBins": "100",
    "cutzvertex": "10",
    "cascadesetting_cospa": "0.998",
    "cascadesetting_v0cospa": "0.97",
    "cascadesetting_dcacascdau": "1",
    "cascadesetting_dcav0dau": "1",
    "cascadesetting_dcabachtopv": "0.06",
    "cascadesetting_dcapostopv": "0.06",
    "cascadesetting_dcanegtopv": "0.06",
    "cascadesetting_mindcav0topv": "0.01",
    "cascadesetting_cascradius": "0.5",
    "cascadesetting_v0radius": "1.2",
    "cascadesetting_v0masswindow": "0.008",
    "cascadesetting_competingmassrej": "0.008"
},
```

Provide the appropriate cuts in the **json configuration file**

The variables must have the same name as their corresponding configurable!

Step 1: apply topological selections



- Background reduces after applying topological selections

Step 2: apply TPC PID selections

- We would like now to apply TPC particle identification (PID) selections to our cascade candidates

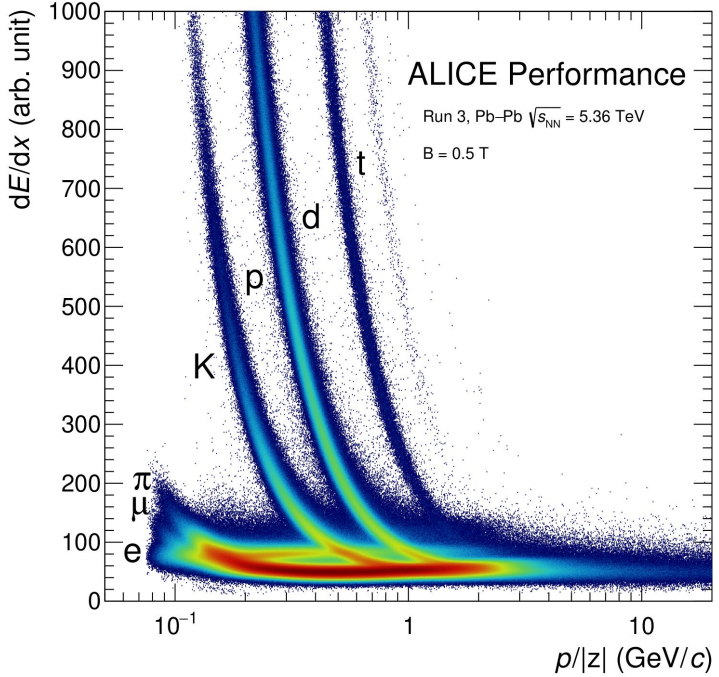
Mean energy loss is parametrized by Bethe-Bloch formula:

$$\begin{cases} \langle -\frac{dE}{dx} \rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2 m_e c^2 \beta^2 \gamma^2 T_{\max}}{I} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \\ \beta\gamma = \frac{p}{Mc} \end{cases}$$

The nature of the cascade decay daughter can be determined using the following PID estimator:

$$n_\sigma = \frac{\langle \mathrm{d}E/\mathrm{d}x \rangle_{\text{meas.}} - \langle \mathrm{d}E/\mathrm{d}x \rangle_{\text{exp.},i}}{\sigma_{\text{TPC}}}$$

with $i = \mathrm{e},\ \mu,\ \pi,\ \mathrm{K},\ \mathrm{p},\ \mathrm{d},\ {}^3\mathrm{He},\ {}^4\mathrm{He}$



Step 2: apply TPC PID selections

- We would like now to apply TPC particle identification (PID) selections to our cascade candidates

- ***Instructions:***
  - Add 3 **Configurable**s, one for each PID selection (*ie*, nσ selections for pion, kaon, proton)
  - De-reference (= typecast cascade to **DauTrackExtras**) each cascade daughter tracks
  - Apply TPC PID selections (**n = 3σ**) on each decay daughters
  - For each mass hypothesis, fill a histogram with the invariant mass after cascade+PID selections

- ***Hints:***
  1. The cascade tables contain the indices to the **aod::DauTrackExtras**, and thus allows to recover the **aod::DauTrackExtras** of each decay daughter using **casc.bachTrackExtra_as<???>()**
  2. The **aod::DauTrackExtras** table does not store TPC PID information. These are stored in a different table which is joinable with **aod::DauTrackExtras**.
     Have a look at `LFStrangenessPIDTables.h` to find the table **aod::DauTrackTPCPIDs** storing the TPC PID information
  3. To get access to a (joined) table (e.g. **aod::DauTrackExtras**), do not forget to subscribe to it

In case you are facing some difficulties, please do not hesitate to ask questions!

Step 2: apply TPC PID selections

```
#include "PWGLF/DataModel/LFStrangenessPIDTables.h"
```

Header needed
`LFStrangenessPIDTables.h`

```
// Configurable parameters for PID selection
Configurable<float> NSigmaTPCPion{"NSigmaTPCPion", 4, "NSigmaTPCPion"};
Configurable<float> NSigmaTPCKaon{"NSigmaTPCKaon", 4, "NSigmaTPCKaon"};
Configurable<float> NSigmaTPCProton{"NSigmaTPCProton", 4, "NSigmaTPCProton"};
```

Add **Configurable** for each PID selection

```
// Defining the type of the daughter tracks
using dauTracks = soa::Join<aod::DauTrackExtras, aod::DauTrackTPCPIDs>;

void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             soa::Filtered<soa::Join<aod::CascCores, aod::CascExtras>> const& Cascades,
             dauTracks const&)
```

Subscribe to **aod::DauTrackTPCPIDs** table

```
// Cascades
for (const auto& casc : Cascades) {
  const auto& bachDaughterTrackCasc = casc.bachTrackExtra_as<dauTracks>();
  const auto& posDaughterTrackCasc = casc.posTrackExtra_as<dauTracks>();
  const auto& negDaughterTrackCasc = casc.negTrackExtra_as<dauTracks>();
```

De-reference each cascade daughter tracks

Step 2: apply TPC PID selections

```
// PID selection
if (casc.sign() < 0) {
  if (TMath::Abs(posDaughterTrackCasc.tpcNSigmaPr()) > NSigmaTPCProton) {
    continue;
  }
  if (TMath::Abs(negDaughterTrackCasc.tpcNSigmaPi()) > NSigmaTPCPion) {
    continue;
  }
} else {
  if (TMath::Abs(negDaughterTrackCasc.tpcNSigmaPr()) > NSigmaTPCProton) {
    continue;
  }
  if (TMath::Abs(posDaughterTrackCasc.tpcNSigmaPi()) > NSigmaTPCPion) {
    continue;
  }
}
```

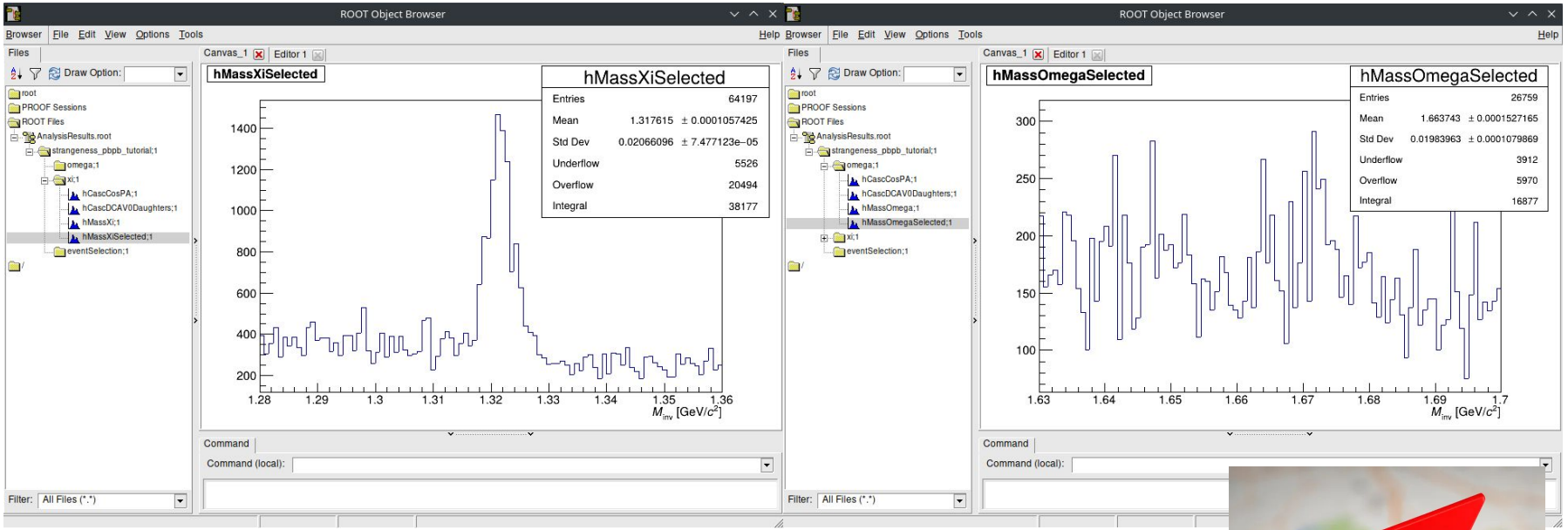
```
// Fill histograms! (if possible)
if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaPi()) < NSigmaTPCPion) { // Xi case
  rXi.fill(HIST("hMassXiSelected"), casc.mXi());
```

```
if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaKa()) < NSigmaTPCKaon) { // Omega case
  if (TMath::Abs(casc.mXi() - pdgDB->Mass(3312)) > cascadesetting_competingmassrej) { // competing mass rejection, only in case of Omega
    rOmega.fill(HIST("hMassOmegaSelected"), casc.mOmega());
```

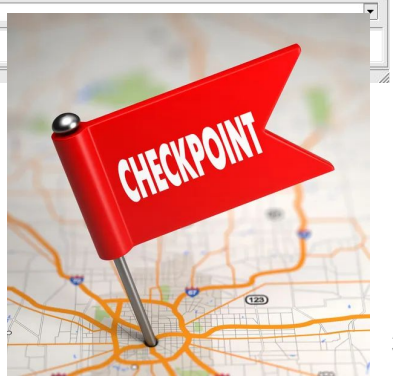
Apply TPC PID selections

Romain Schotter (romain.schotter@cern.ch)

Step 2: apply TPC PID selections



- Background is further reduced using TPC PID (**n = 3σ**) selections



Step 3: apply TOF PID selections

- We would like now to apply TOF PID selections on each decay daughters, but only if they do have signal in the TOF

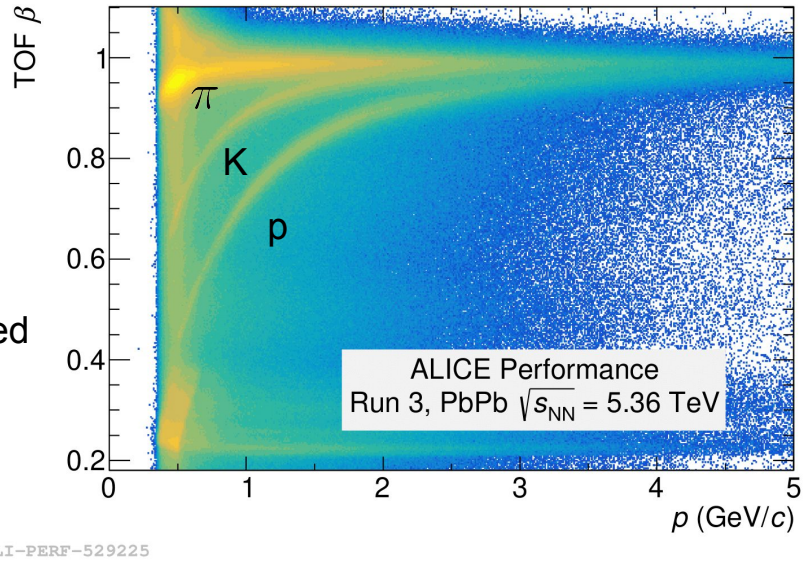
By measuring the particle velocity, the expected time-of-flight can be estimated:

$$t_{\mathrm{exp.},i} = L \frac{\sqrt{p^2 + m_i^2}}{cp}$$

The nature of the cascade decay daughter can be determined using the following PID estimator:

$$n_\sigma = \frac{t_{\mathrm{meas.}} - t_{\mathrm{ev}} - t_{\mathrm{exp.},i}}{\sigma}$$

with $i = \mathrm{e},\ \mu,\ \pi,\ \mathrm{K},\ \mathrm{p},\ \mathrm{d},\ {}^3\mathrm{He},\ {}^4\mathrm{He}$



Romain Schotter (romain.schotter@cern.ch)

Step 3: apply TOF PID selections

- We would like now to apply TOF PID selections on each decay daughters, but only if they do have signal in the TOF

- ***Instructions:***
  - Add 3 **Configurable**s, one for each TOF PID selection (*ie*, nσ selections for pion, kaon, proton)
  - De-reference (= typecast cascade to **DauTrackExtras**) each cascade daughter tracks
  - Apply TOF PID selections **(n = 3σ)** on each decay daughters, **if** it has a matched hit in the TOF
  - For each mass hypothesis, fill a histogram with the invariant mass after cascade+PID selections

- ***Hints:***
  1. There already exists a dynamic column to check whether a track has a hit in the TOF. Have a look at **aod::DauTrackExtras** in `LFStrangenessTables.h`
  2. The TOF PID information of daughter tracks are stored in a table called **aod::CascTOFNSigmas** in `LFStrangenessPIDTables.h` joinable with **aod::CascCores**

In case you are facing some difficulties, please do not hesitate to ask questions!

Step 3: apply TOF PID selections

```
// Configurable parameters for TOF PID selection
Configurable<float> NSigmaTOFPion{"NSigmaTOFPion", 3, "NSigmaTOFPion"};
Configurable<float> NSigmaTOFKaon{"NSigmaTOFKaon", 3, "NSigmaTOFKaon"};
Configurable<float> NSigmaTOFProton{"NSigmaTOFProton", 3, "NSigmaTOFProton"};
```

Add **Configurable** for each TOF PID selection

```
void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             soa::Filtered<soa::Join<aod::CascCores, aod::CascExtras, aod::CascTOFNSigmas>> const& Cascades,
             dauTracks const&)
```

Subscribe to **aod::CascTOFNSigmas** table

```
// TOF PID check
bool xiPassTOFSelection = true;
bool omegaPassTOFSelection = true;
if (casc.sign() < 0) {
  if (posDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPr()) > NSigmaTOFProton) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPr()) > NSigmaTOFProton) {
      omegaPassTOFSelection &= false;
    }
  }
  if (negDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPi()) > NSigmaTOFPion) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPi()) > NSigmaTOFPion) {
      omegaPassTOFSelection &= false;
    }
  }
```

Apply TOF PID selections **if** the track has a hit in TOF

Step 3: apply TOF PID selections

```
// TOF PID check
bool xiPassTOFSelection = true;
bool omegaPassTOFSelection = true;
if (casc.sign() < 0) {
  if (posDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPr()) > NSigmaTOFProton) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPr()) > NSigmaTOFProton) {
      omegaPassTOFSelection &= false;
    }
  }
  if (negDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPi()) > NSigmaTOFPion) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPi()) > NSigmaTOFPion) {
      omegaPassTOFSelection &= false;
    }
  }
} else {
  if (posDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPi()) > NSigmaTOFPion) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPi()) > NSigmaTOFPion) {
      omegaPassTOFSelection &= false;
    }
  }
  if (negDaughterTrackCasc.hasTOF()) {
    if (TMath::Abs(casc.tofNSigmaXiLaPr()) > NSigmaTOFProton) {
      xiPassTOFSelection &= false;
    }
    if (TMath::Abs(casc.tofNSigmaOmLaPr()) > NSigmaTOFProton) {
      omegaPassTOFSelection &= false;
    }
  }
}

if (bachDaughterTrackCasc.hasTOF()) {
  if (TMath::Abs(casc.tofNSigmaXiPi()) > NSigmaTOFPion) {
    xiPassTOFSelection &= false;
  }
  if (TMath::Abs(casc.tofNSigmaOmKa()) > NSigmaTOFKaon) {
    omegaPassTOFSelection &= false;
  }
}
```

Apply TOF PID selections **if** the track has a hit in TOF

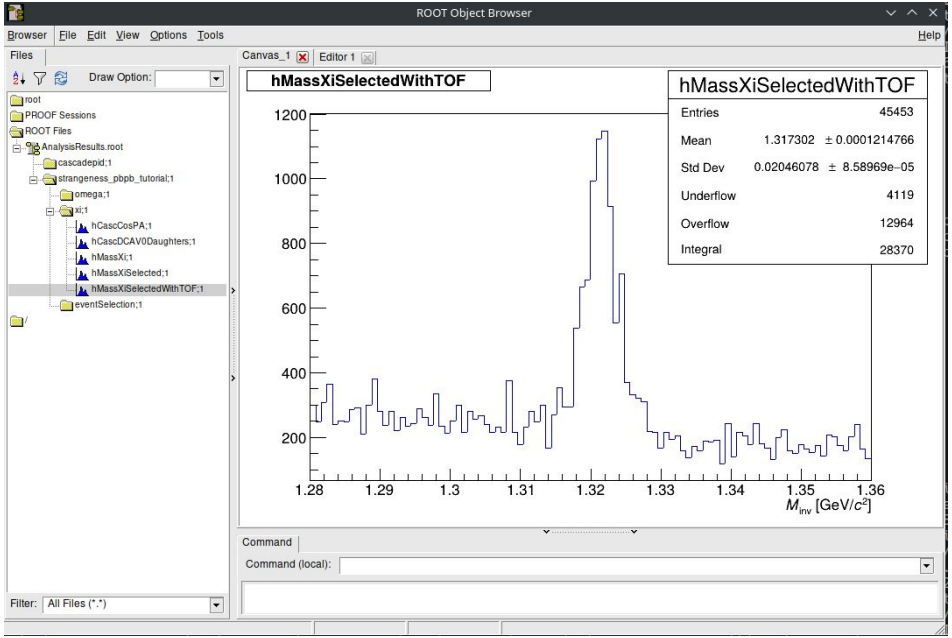
Fill invariant mass histograms <u>after</u> selections

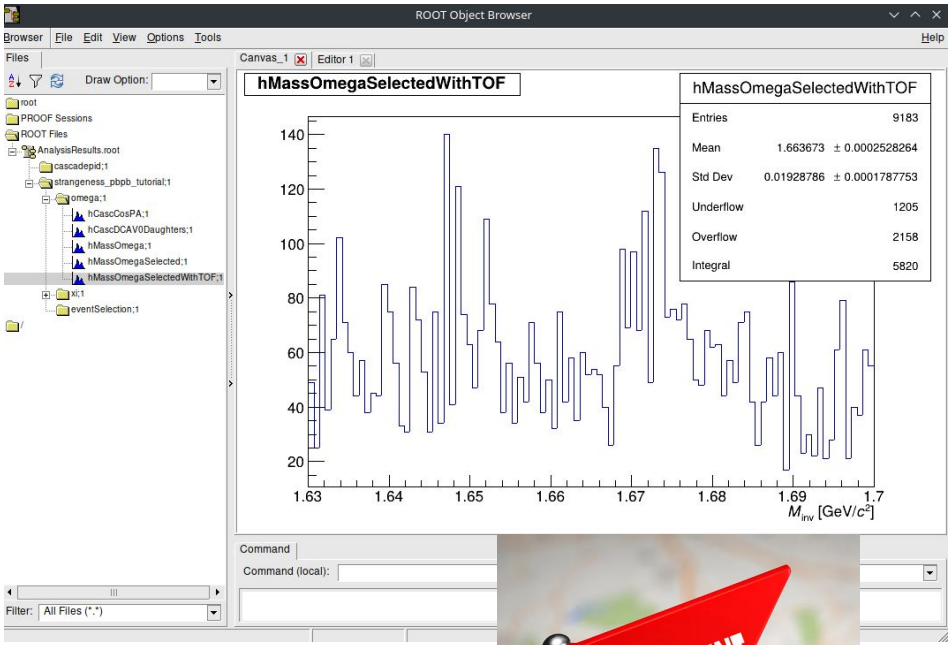
```
// Fill histograms! (if possible)
if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaPi()) < NSigmaTPCPion) { // Xi case
  rXi.fill(HIST("hMassXiSelected"), casc.mXi());
  if (xiPassTOFSelection)
    rXi.fill(HIST("hMassXiSelectedWithTOF"), casc.mXi());
```

```
if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaKa()) < NSigmaTPCKaon) { // Omega case
  if (TMath::Abs(casc.mXi() - pdgDB->Mass(3312)) > cascadesetting_competingmassrej) {
    rOmega.fill(HIST("hMassOmegaSelected"), casc.mOmega());
    if (omegaPassTOFSelection)
      rOmega.fill(HIST("hMassOmegaSelectedWithTOF"), casc.mOmega());
```


Romain Schotter (romain.schotter@cern.ch)

Step 3: apply TOF PID selections





- TOF (**n = 3σ**) selections help further suppressing combinatorial background, a peak starts to emerge close to PDG mass of the Ω

Step 4: access MC information of cascades

- From the previous steps, one can reconstruct the Xi and Omega with excellent purities, and evaluate their raw yields
- In order to converge towards corrected results, we would like now to move towards determining their efficiencies → **need to access MC information**
- Let's have a look at the number of reconstructed true cascade as a function of transverse momentum

aod::CascCores

→ **reconstructed** info about cascades

Romain Schotter (romain.schotter@cern.ch)

Step 4: access MC information of cascades

- From the previous steps, one can reconstruct the Xi and Omega with excellent purities, and evaluate their raw yields
- In order to converge towards corrected results, we would like now to move towards determining their efficiencies → **need to access MC information**
- Let's have a look at the number of reconstructed true cascade as a function of transverse momentum

aod::CascCores

→ **reconstructed** info about cascades

aod::CascMCCores

→ **generated** info about cascades

Romain Schotter (romain.schotter@cern.ch)

Step 4: access MC information of cascades

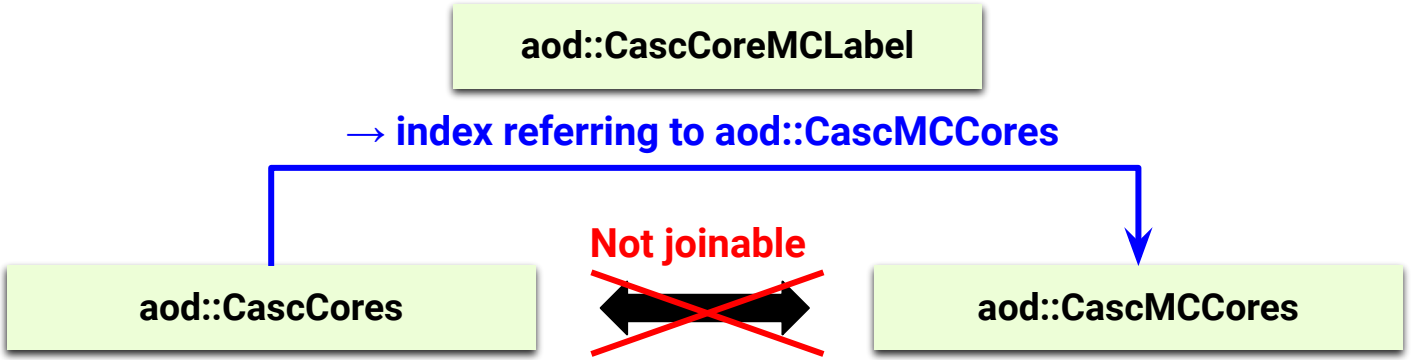
- From the previous steps, one can reconstruct the Xi and Omega with excellent purities, and evaluate their raw yields
- In order to converge towards corrected results, we would like now to move towards determining their efficiencies → **need to access MC information**
- Let's have a look at the number of reconstructed true cascade as a function of transverse momentum

→ **reconstructed** info about cascades

→ **generated** info about cascades

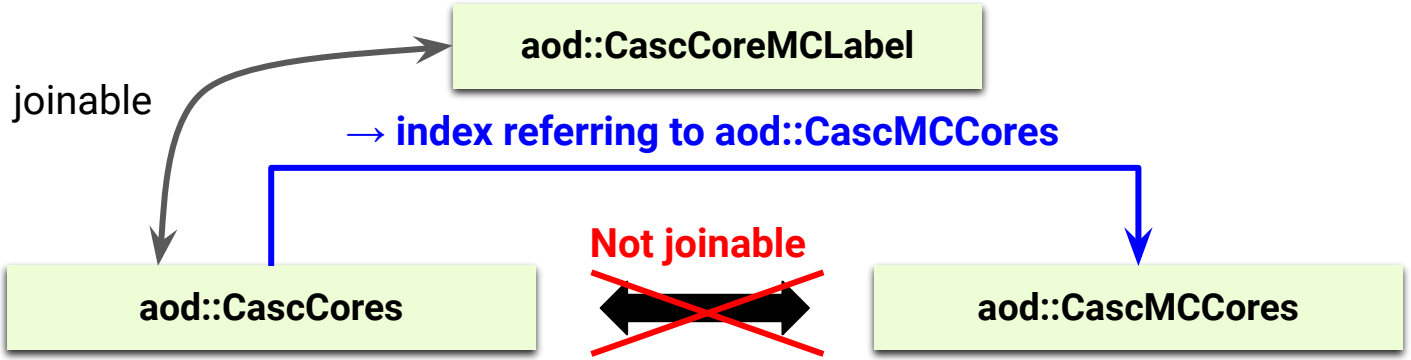
Step 4: access MC information of cascades

- From the previous steps, one can reconstruct the Xi and Omega with excellent purities, and evaluate their raw yields
- In order to converge towards corrected results, we would like now to move towards determining their efficiencies → **need to access MC information**
- Let's have a look at the number of reconstructed true cascade as a function of transverse momentum



Step 4: access MC information of cascades

- From the previous steps, one can reconstruct the Xi and Omega with excellent purities, and evaluate their raw yields
- In order to converge towards corrected results, we would like now to move towards determining their efficiencies → **need to access MC information**
- Let's have a look at the number of reconstructed true cascade as a function of transverse momentum



→ **reconstructed** info about cascades

→ **generated** info about cascades

Step 4: access MC information of cascades

```
void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             soa::Filtered<soa::Join<aod::CascCores, aod::CascExtras, aod::CascTOFNSigmas, aod::CascCoreMCLabels>> const& Cascades,
             dauTracks const&,
             aod::CascMCCores const& /*cascmccores*/)
```

Subscribe to **aod::CascCoreMCLabels** and **aod::CascMCCores** tables

```
// MC truth info
if(!casc.has_cascMCCore()) {
  continue;
}
auto cascmccore = casc.cascMCCore_as<aod::CascMCCores>();
```

Check that cascades come from a gen. particles via **has_cascMCCore()** and recover the gen. info

- ***Instructions:***
  - Produce the derived MC data or download them from cernbox
  - For each mass hypothesis, fill a histogram with the transverse momentum of the true reconstructed Ξ or Ω

- ***Hints:***
  1. Have a look at `LFStrangenessTable.h` to find the appropriate columns
  2. To separate a true Ξ from a true Ω, use their **PDG code**: **3312** for Ξ, **3334** for Ω
  3. For the binning in pT, use the following binning: 100 bins, from 0 to 10 GeV/*c*

In case you are facing some difficulties, please do not hesitate to ask questions!

Step 4: access MC information of cascades

```
void process(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
             soa::Filtered<soa::Join<aod::CascCores, aod::CascExtras, aod::CascTOFNSigmas, aod::CascCoreMCLabels>> const& Cascades,
             dauTracks const&,
             aod::CascMCCores const& /*cascmccores*/)
```

```
// MC truth info
if (!casc.has_cascMCCore()) {
  continue;
}
auto cascmccore = casc.cascMCCore_as<aod::CascMCCores>();

// Checking that the cascade is a true Xi
if (TMath::Abs(cascmccore.pdgCode()) == 3312) {
  if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaPi()) < NSigmaTPCPion) { // Xi case
    rXi.fill(HIST("hMassXiTrueRec"), casc.mXi());
    rXi.fill(HIST("hPtXiTrueRec"), casc.pt());
    if (xiPassTOFSelection) {
      rXi.fill(HIST("hMassXiTrueRecWithTOF"), casc.mXi());
      rXi.fill(HIST("hPtXiTrueRecWithTOF"), casc.pt());
    }
  }
}
if (TMath::Abs(cascmccore.pdgCode()) == 3334) {
  if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaKa()) < NSigmaTPCKaon) {
    if (TMath::Abs(casc.mXi() - pdgDB->Mass(3312)) > cascadesetting_competingmassrej) {
      rOmega.fill(HIST("hMassOmegaTrueRec"), casc.mOmega());
      rOmega.fill(HIST("hPtOmegaTrueRec"), casc.pt());
      if (omegaPassTOFSelection) {
        rOmega.fill(HIST("hMassOmegaTrueRecWithTOF"), casc.mOmega());
        rOmega.fill(HIST("hPtOmegaTrueRecWithTOF"), casc.pt());
      }
    }
  }
}
```

Subscribe to **aod::CascCoreMCLabels** and **aod::CascMCCores** tables

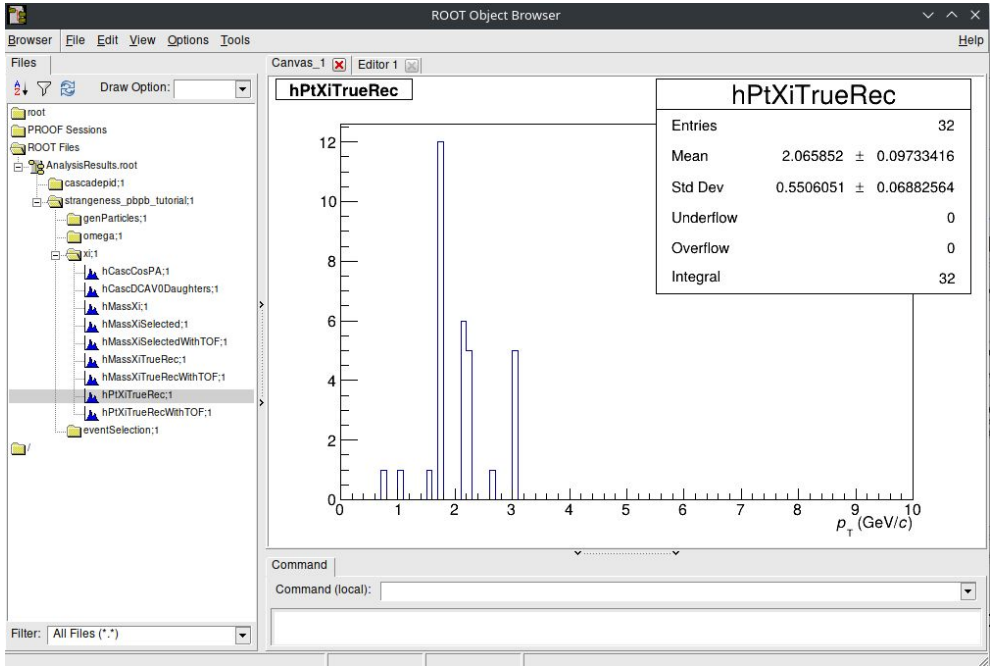
Check that cascades come from a gen. particles via **has_cascMCCore()** and recover the gen. info

+

De-reference the corresponding entry in the **aod::CascMCCores** table

Check PDG code of gen. cascades via **pdgCode()**

Step 4: access MC information of cascades



- The number of reconstructed Ξ has been extracted as a function of the transverse momentum
- Not enough statistics for the Ω (note this is done only with 1 derived data file)

Step 4: loop over generated events

- We have the number of reconstructed true cascades as a function of transverse momentum
- Now, we need the **number of generated cascades as a function of transverse momentum**
  → requires a new process function that iterates over MC collisions

```
void processRecMC(soa::Filtered<soa::Join<aod::StraCollisions, aod::StraEvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::CascCores, aod::CascExtras, aod::CascTOFNSigmas, aod::CascCoreMCLabels>> const& Cascades,
                  dauTracks const&,
                  aod::CascMCCores const& /*cascmccores*/)
```

Rename the previous **process()** function in **processRecMC()**

```
void processGenMC(soa::Filtered<aod::StraMCCollisions>::iterator const& mcCollision,
                  const soa::SmallGroups<soa::Join<aod::StraCollisions, o2::aod::StraEvSels, aod::StraCollLabels>>& collisions,
                  const soa::SmallGroups<soa::Join<aod::CascMCCores, aod::CascMCCollRefs>>& cascMC)
```

Create an additional function **processGenMC()**

Use **soa::SmallGroups** to select reconstructed collisions and gen. cascades associated to this MC collision

```
PROCESS_SWITCH(strangeness_pbpb_tutorial, processRecMC, "Process Run 3 mc, reconstructed", true);
PROCESS_SWITCH(strangeness_pbpb_tutorial, processGenMC, "Process Run 3 mc, generated", true);
```

Add **PROCESS_SWITCH** to allow for 2 process functions

Romain Schotter (romain.schotter@cern.ch)

Step 4: loop over generated events

- The next step is to:
  - check that the MC collision have been reconstructed at least once using the **.size()**
  - Loop over all generated cascades
  - Fill a histogram with the generated transverse momentum for each mass hypothesis

```
void processGenMC(soa::Filtered<aod::StraMCCollisions>::iterator const& mcCollision,
                  const soa::SmallGroups<soa::Join<aod::StraCollisions, o2::aod::StraEvSels, aod::StraCollLabels>>& collisions,
                  const soa::SmallGroups<soa::Join<aod::CascMCCores, aod::CascMCCollRefs>>& cascMC)
```


Romain Schotter (romain.schotter@cern.ch)

Step 4: loop over generated events

- The next step is to:
  - check that the MC collision have been reconstructed at least once using the **.size()**
  - Loop over all generated cascades
  - Fill a histogram with the generated transverse momentum for each mass hypothesis

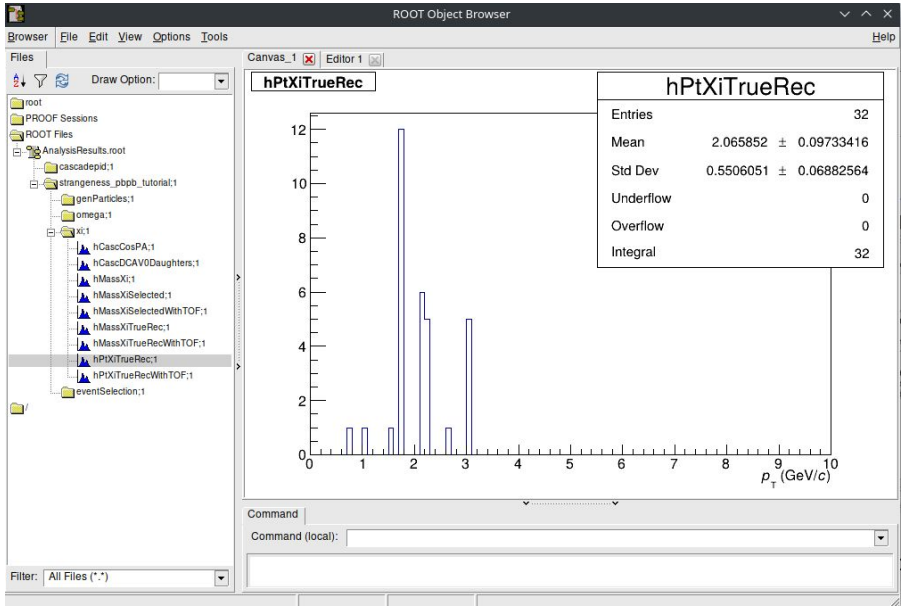
```
void processGenMC(soa::Filtered<aod::StraMCCollisions>::iterator const& mcCollision,
                  const soa::SmallGroups<soa::Join<aod::StraCollisions, o2::aod::StraEvSels, aod::StraCollLabels>>& collisions,
                  const soa::SmallGroups<soa::Join<aod::CascMCCores, aod::CascMCCollRefs>>& cascMC)
{
  if (collisions.size() < 1) // to process generated collisions that've been reconstructed at least once
    return;
  rEventSelection.fill(HIST("hVertexZGen"), mcCollision.posZ());

  for (const auto& cascmc : cascMC) {
    if (TMath::Abs(cascmc.pdgCode()) == 3312) {
      rGenParticles.fill(HIST("hPtXiGen"), cascmc.ptMC());
    }
    if (TMath::Abs(cascmc.pdgCode()) == 3334) {
      rGenParticles.fill(HIST("hPtOmegaGen"), cascmc.ptMC());
    }
  }
}
```

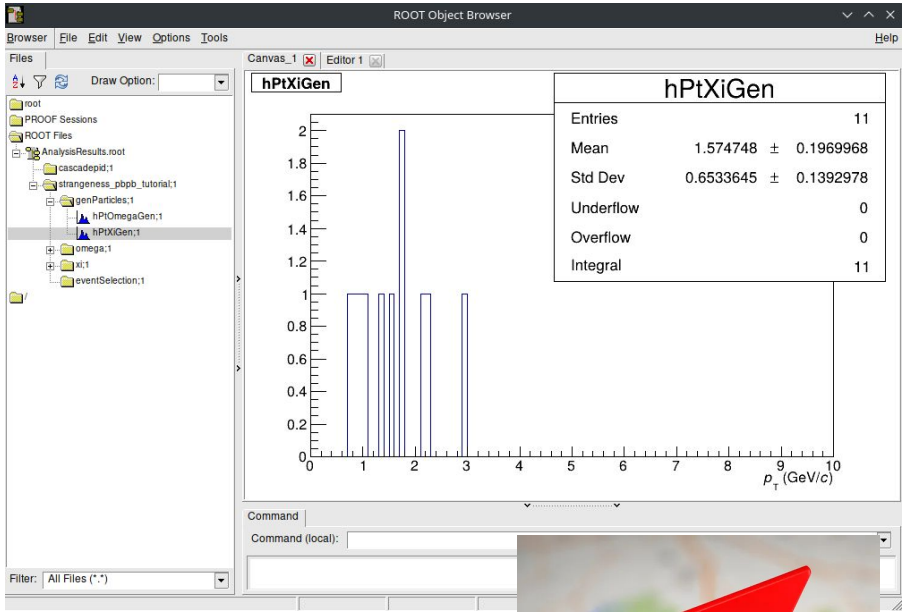
Loop over all generated cascades

Fill a histogram with the gen. pt

Romain Schotter (romain.schotter@cern.ch)

Step 4: loop over generated events





- The number of reconstructed and generated Ξ have been extracted as a function of the transverse momentum
- One step away from having the efficiency corrections…

Summary

In this hands-on session you learned:

- produce strangeness derived tables
- how to subscribe to the derived tables in your analysis
- apply simple topological and kinematic selections
- apply TPC and TOF PID selections
- perform an analysis on MC

This was just a taste of an analysis on (multi-)strange hadrons!

Summary

There are many other analysis!

Feel free to contact analysers and have a peek at their task!

Decay topology / Particle	System@Energy	Main observables	People	Analysis Note
V0s	pp@900 GeV	spectra vs. multiplicity	Francesca Ercolessi, Nicolò Jacazio	
Cascades	pp@900 GeV	spectra vs. multiplicity	Chiara De Martin, Francesca Ercolessi, Roman Nepeivoda, Upasana Sharma	AN-1446
V0s	pp@13.6 TeV	spectra vs. multiplicity	Francesca Ercolessi, Nicolò Jacazio	
Cascades	pp@13.6 TeV	spectra vs. multiplicity	Chiara De Martin, Francesca Ercolessi, Roman Nepeivoda, Upasana Sharma	AN-1446
V0s and Cascades	pp@13.6 TeV	h-V0, h-Casc correlations	Kai Cui, David Chinellato, Lucia Tarasovicová, Zhong-Bao Yin, Chiara De Martin	AN-1550
Cascades	pp@13.6 TeV	correlations	Rik Spijkers	
Hyperons	pp@13.6 TeV	hyperon absorption in the beam pipe and detector material	Alberto Caliva, Francesco Mazzaschi, Maximiliano Puccio	
Hyperons	pp@13.6 TeV	elastic scattering of hyperons with nuclei in ITS	Alberto Caliva	
Cascades	pp@13.6 TeV, Pb-Pb@5.36 TeV	prompt and non-prompt cascades	Andrea Sofia Triolo, Maximiliano Puccio	
Sigma	pp@13.6 TeV	kink analysis, spectra	Paraskevi Ganoti	
Sigma	pp@13.6 TeV	Sigma -> Λ + gamma	Gianni Shigeru Setoue Liveraro, Jun Takahashi, David Dobrigkeit Chinellato	
V0s and Cascades	pp@13.6 TeV	production in jets with jet finder	Alberto Calivà, Francesca Ercolessi, Chiara De Martin	
V0s	pp@13.6 TeV	Jet fragmentation into V0	Gijs Van Weelden	
V0s	pp@13.6 TeV	yields vs charged particle flattenicity	Suraj Prasad	
V0s	Pb-Pb@5.36 TeV	spectra vs. centrality	Romain Schotter, David Dobrigkeit Chinellato	AN-1540
Cascades	Pb-Pb@5.36 TeV	spectra vs. centrality	Lucia Tarasovicová	AN-1512
Cascades	Pb-Pb@5.36 TeV	v2 (v3) and polarization	Sourav Kundu, Maximiliano Puccio, Chiara De Martin	AN-1521
Hyperons	Pb-Pb@5.36 TeV	Hyperon polarization	Junlee Kim	AN-1561
Cascades	pp@13.6 TeV	K0s - phi correlation	Stefano Cannito, Valentina Zaccolo	AN-1563

Thank you very much!

Romain Schotter (romain.schotter@cern.ch)

Additional materials

Analysis Level Tables: aod::V0Data(Ext)

#include "PWGLF/DataModel/LFStrangenessTables.h" V0Datas = soa::Join<**V0Indices**, V0TrackXs, **V0Cores**>

	Name	Getter
Bound	v0::PosTrackId	posTrackId
	v0::NegTrackId	negTrackId
	v0::CollisionId	collisionId
	v0::V0	v0Id
Momenta of daughter tracks	v0data::PxPos	pxpos
	v0data::PyPos	pypos
	v0data::PzPos	pzpos
	v0data::PxNeg	pxneg
	v0data::PyNeg	pyneg
	v0data::PzNeg	pxneg
Calculated	v0data::X	x
Decay vertex position	v0data::Y	y
	v0data::Z	z
Topological variables	v0data::DCAV0Daughters	dcav0daughters
	v0data::DCAPosToPV	dcapostopv
	v0data::DCANegToPV	dcanegtopv

Derived (dynamic)	v0data::Pt	pt
	v0data::V0Radius	v0radius
	v0data::V0CosPA	v0cospa
	v0data::DCAV0ToPV	dcav0topv
	v0data::MLambda	mLambda
	v0data::MAntiLambda	mAntiLambda
	v0data::MK0Short	mK0Short
	v0data::YK0Short	yK0Short
	v0data::YLambda	yLambda
	v0data::Eta	eta
	v0data::Phi	phi
	...	...
Derived (expression)	v0dataext::Px	px
	v0dataext::Py	py
	v0dataext::Pz	pz

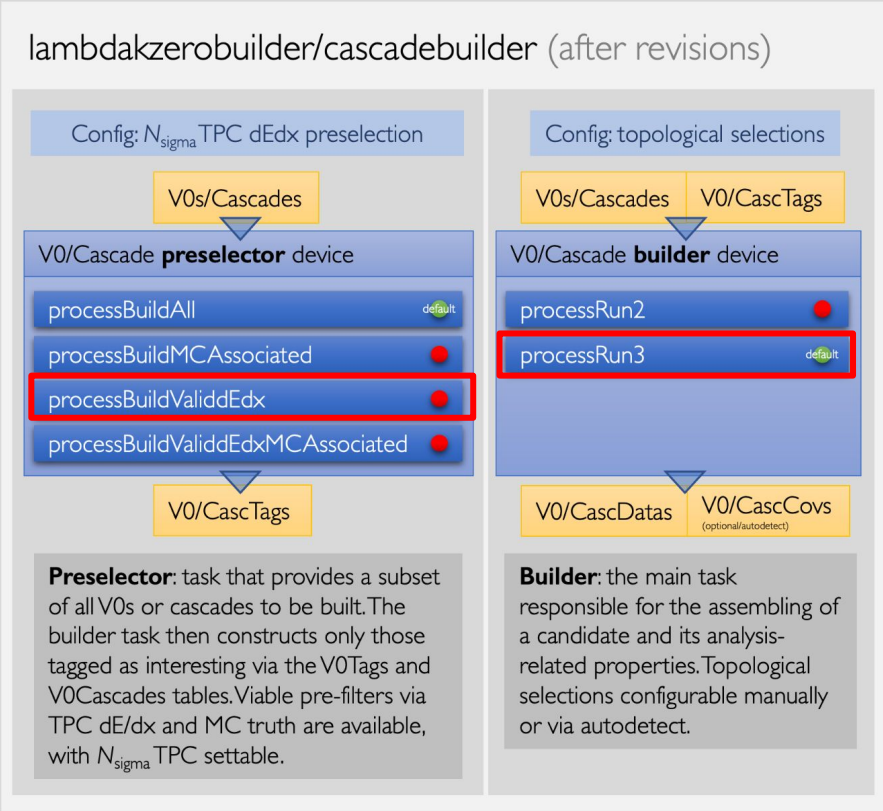
Analysis Level Tables: aod::CascData(Ext)

#include "PWGLF/DataModel/LFStrangenessTables.h" CascDatas = soa::Join<**CascIndices**, CascBBs, **CascCores**>

	Name	Getter
Bound	cascade::V0Id	v0Id
	cascade::BachelorId	bachelorId
	cascade::CollisionId	collisionId
Momenta of daughter tracks	cascdata::Sign	sign
	cascdata::PxPos(Neg)	pxpos(neg)
	cascdata::PyPos(Neg)	pypos(neg)
	cascdata::PzPos(Neg)	pzpos(neg)
	cascdata::PxBach	pxbach
	cascdata::PyBach	pybach
	cascdata::PzBach	pzbach
Calculated	cascdata::X	x
Decay vertex position	cascdata::Y	y
	cascdata::Z	z
Topological variables	cascdata::DCAV0Daughters	dcav0daughters
	cascdata::DCACascDaughters	dcascascdaughters
	cascdata::DCAPosToPV	dcapostopv

	cascdata::DCANegToPV	dcanegtopv
	cascdata::DCABachToPV	dcabachtopv
	cascdata::Pt	pt
	...	...
Derived (dynamic)	cascdata::V0Radius	v0radius
	cascdata::CascRadius	cascradius
	cascdata::V0CosPA	v0cosPA
	cascdata::CascCosPA	casccosPA
	cascdata::MLambda	mLambda
	cascdata::MXi	mXi
	cascdata::MOmega	mOmega
	cascdata::YXi	yXi
	...	...
Derived (expression)	cascdataext::Pt	pt
	cascdataext::P	p
	cascdataext::PxLambda	pxlambda
	...	...

V0/cascade table producer task



- V0/cascade tables produced in the `lambdakzerobuilder.cxx` and `cascadebuilder.cxx` files
- **Pre-selector** device:
  - **processBuild** functions → select V0s/casc. based on TPC N σ selections
  - **Skipper** options to optionally skip V0s not used in casc.
- **Builder** device:
  - Look at candidates tagged as interesting → **buildStrangenessTables()**
- Output:
  - **V0Datas**: regular V0 analysis table
  - **CascDatas**: regular cascade analysis table
  - **TraCascDatas**: tracked cascade analysis table
  - **KFCascDatas**: KF cascade analysis table