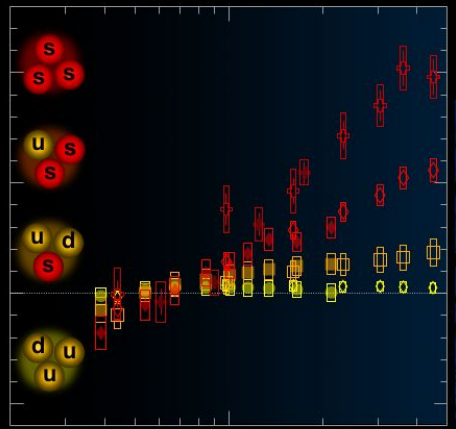
Strangeness hands-on

O2 Analysis Tutorial 08/11/2023



Nepeivoda Roman

Lund University, Sweden

nepeivoda.roman@cern.ch

Outline

- In this hands-on session you will write a simple task in 5 steps

  to analyse V0-s and cascades
- You will learn how to
  - Structure your analysis task
  - Reconstruct V0-s and cascades
  - Apply PID, topological and kinematic selections
  - Access MC information of the cascades, V0-s and their daughters

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

Preparation

- The workflow to run the task at each of the 5 steps can be taken from the github repository: nepeivodaRS/StrangenessHands-on

```
git clone https://github.com/nepeivodaRS/StrangenessHands-on
cd StrangenessHands-on
```

Preparation

- The workflow to run the task at each of the 5 steps can be taken from the github repository: nepeivodaRS/StrangenessHands-on

```
git clone https://github.com/nepeivodaRS/StrangenessHands-on
cd StrangenessHands-on
```

Configurations (stepX.json) are stored in the **jsonConfigs** folder

Run scripts (runStepX.sh) are stored in the **StrangenessHands-on** folder

Preparation

- The workflow to run the task at each of the 5 steps can be taken from the github repository: nepeivodaRS/StrangenessHands-on

```
git clone https://github.com/nepeivodaRS/StrangenessHands-on
cd StrangenessHands-on
```

- Create folder to store the data
- Download data file (~618 MB) to a **data** folder:

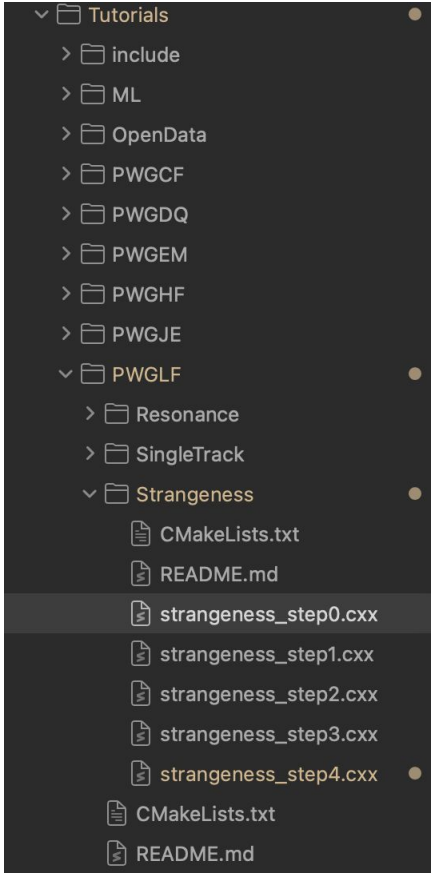
```
mkdir data
rsync -r -u --progress USERNAME@lxplus.cern.ch:/eos/user/r/rnepeivo/tutorialData/AO2D.root data
```

- Enter your lxplus password (if required)

- Alternatively, download the file from CERNBOX manually

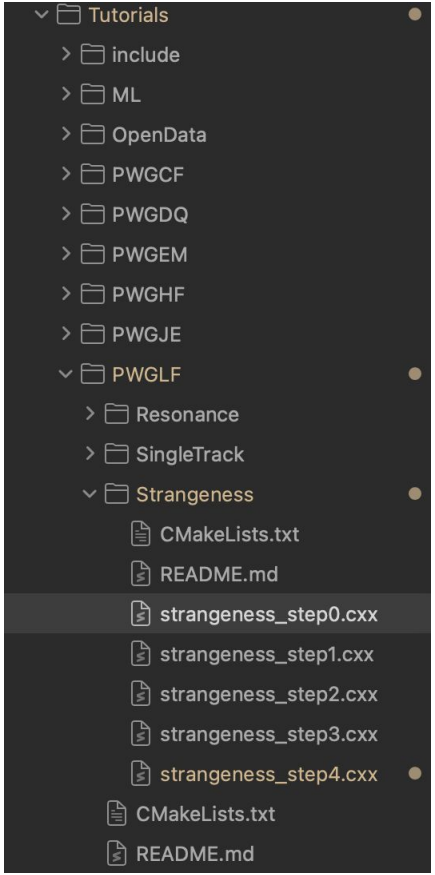
Preparation

- The task is located in: O2Physics/Tutorials/PWGLF/Strangeness
- The idea is to start with step0 file and improve it up to step4 (or even further!)
- Checkpoints for each step are here for your convenience (self-check)



Preparation

- The task is located in: O2Physics/Tutorials/PWGLF/Strangeness
- The idea is to start with step0 file and improve it up to step4 (or even further!)
- Checkpoints for each step are here for your convenience (self-check)
- Let’s have a look at the starting point (**strangeness_step0.cxx**)



Step 0: loop over V0 candidates

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

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

// STEP 0
// Starting point: loop over all V0s and fill invariant mass histogram

struct strangeness_tutorial {
  // Histograms are defined with HistogramRegistry
  HistogramRegistry rEventSelection{"eventSelection", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};
  HistogramRegistry rKzeroShort{"kzeroShort", {}, 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)"};

  void init(InitContext const&)
  {
    // Axes
    AxisSpec K0ShortMassAxis = {200, 0.45f, 0.55f, "#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}});

    // K0s reconstruction
    rKzeroShort.add("hMassK0Short", "hMassK0Short", {HistType::kTH1F, {K0ShortMassAxis}});
  }
```

Include required **headers**

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

Add needed **histograms** to the registries

Step 0: loop over V0 candidates

```
  // Defining filters for events (event selection)
  // Processed events will be already fulfilling the event selection requirements
  Filter eventFilter = (o2::aod::evsel::sel8 == true);
  Filter posZFilter = (nabs(o2::aod::collision::posZ) < cutzvertex);

  void process(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
               aod::V0Datas const& V0s)
  {
    // Fill the event counter
    rEventSelection.fill(HIST("hVertexZRec"), collision.posZ());

    for (const auto& v0 : V0s) {
      rKzeroShort.fill(HIST("hMassK0Short"), v0.mK0Short());
    }
  }
};

WorkflowSpec defineDataProcessing(ConfigContext const& cfgc)
{
  return WorkflowSpec{
    adaptAnalysisTask<strangeness_tutorial>(cfgc)};
}
```

Define **Filters** for basic event selection

Subscribe to **aod::V0Datas** to loop over the V0s associated to each of the selected collisions

Perform a loop over V0-s and fill the invariant mass histogram

- **Converter** tasks are required for our AO2D
- **o2-analysis-lf-lambdakzerobuilder** produces **aod::V0Datas** table

o2-analysis-timestamp
o2-analysis-event-selection
o2-analysis-bc-**converter**
o2-analysis-tracks-extra-**converter**
o2-analysis-lf-lambdakzerobuilder
o2-analysistutorial-lf-strangeness-step0

The Structure of a shell running script

```
#!/bin/bash
# log file where the terminal output will be saved
STEP="step0"
LOGFILE="log-${STEP}.txt"

# directory of this script
DIR_THIS="$(dirname "$(realpath "$0")")"

OPTION="-b --configuration json:/$DIR_THIS/jsonConfigs/$STEP.json"

o2-analysis-timestamp ${OPTION} \
| o2-analysis-event-selection ${OPTION} \
| o2-analysis-bc-converter ${OPTION} \
| o2-analysis-tracks-extra-converter  ${OPTION} \
| o2-analysis-track-propagation ${OPTION} \
| o2-analysis-lf-lambdakzerobuilder ${OPTION} \
| o2-analysis-tutorial-lf-strangeness-$STEP ${OPTION} \
> "$LOGFILE" 2>&1

# report status
rc=$?
if [ $rc -eq 0 ]; then
  echo "No problems!"
  mkdir -p "$DIR_THIS/results/$STEP"
  mv "AnalysisResults.root" "$DIR_THIS/results/$STEP/AnalysisResults.root"
  mv "dpl-config.json" "$DIR_THIS/jsonConfigs/$STEP.json"
else
  echo "Error: Exit code $rc"
  echo "Check the log file $LOGFILE"
  exit $rc
fi
```

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** and **json file** are moved to the intended folders

Step 0: loop over V0 candidates

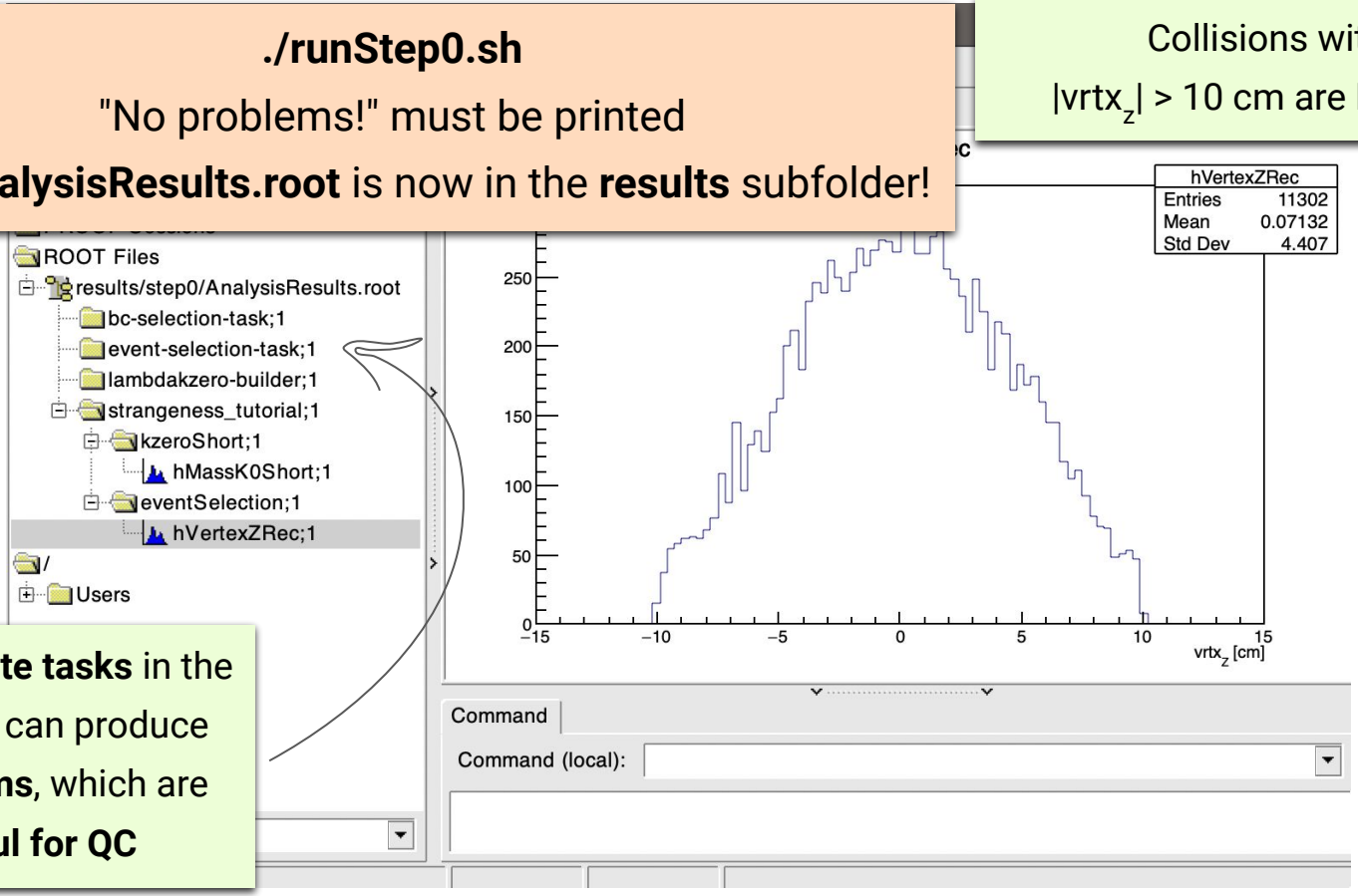
./runStep0.sh

"No problems!" must be printed

your **AnalysisResults.root** is now in the **results** subfolder!

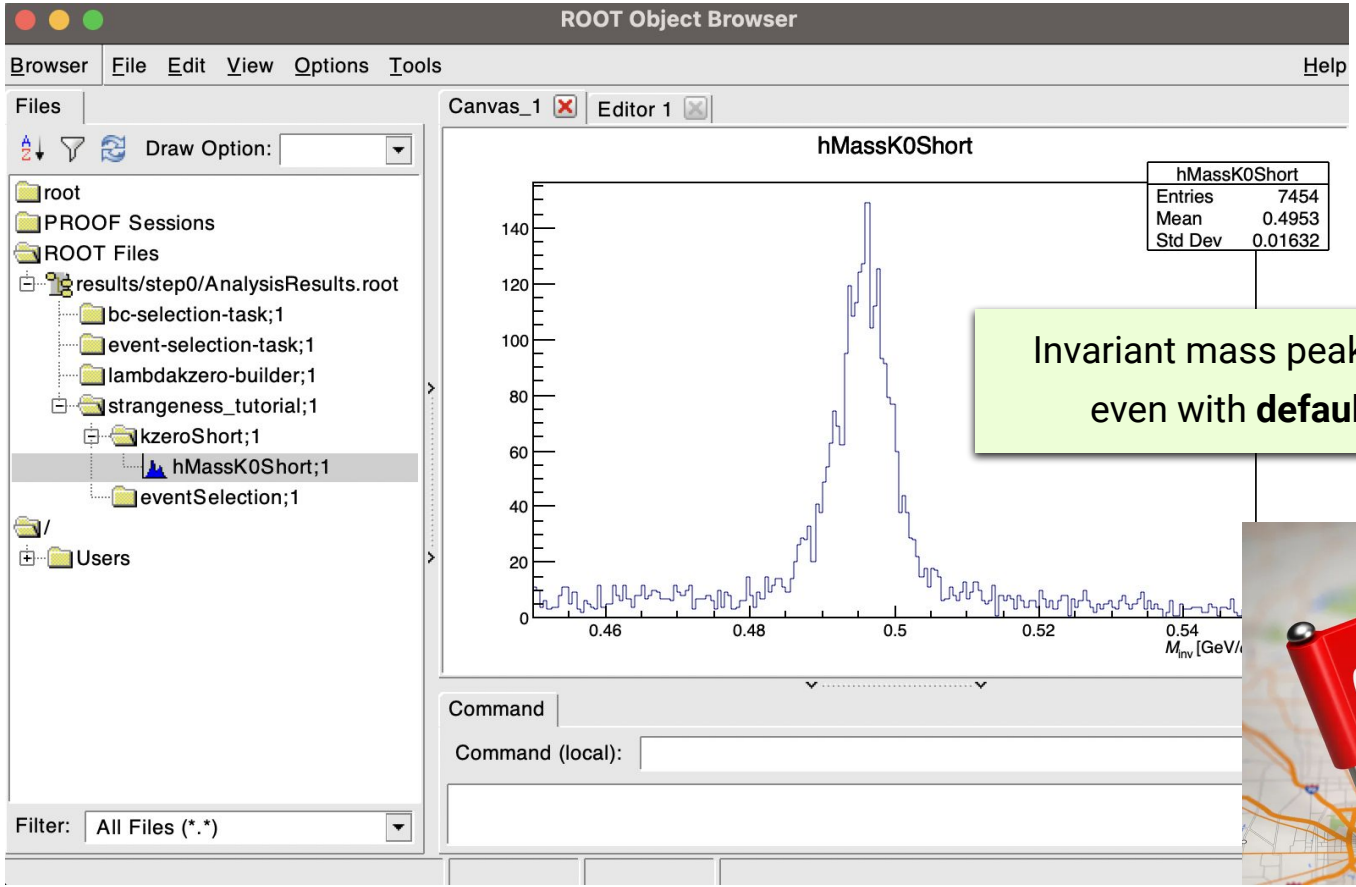
Collisions with

$|vrtx_z| > 10$ cm are **Filtered**



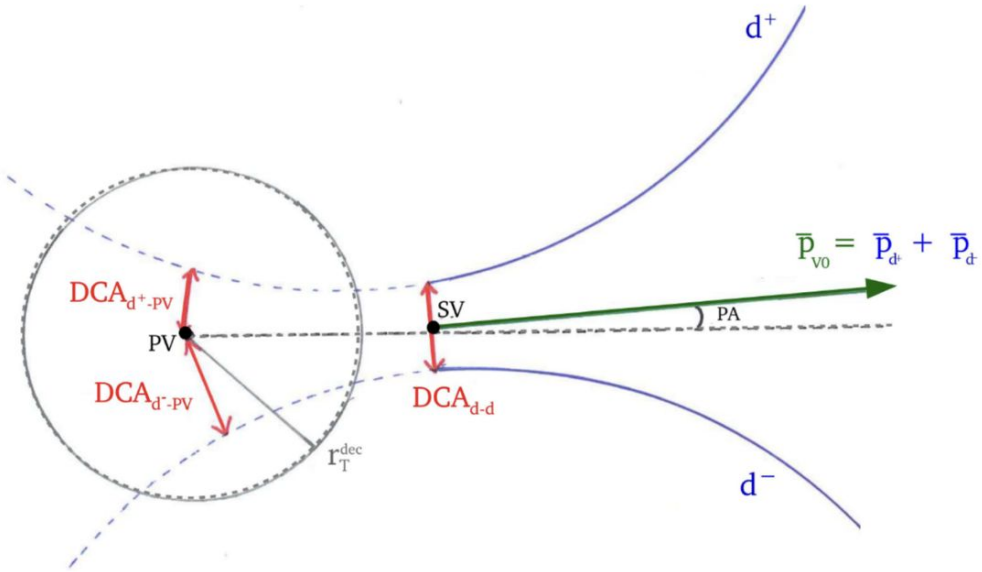
intermediate tasks in the workflow can produce **histograms**, which are **useful for QC**

Step 0: loop over V0 candidates



Invariant mass peak is already pure even with **default** set of cuts

Step 1: topological selections



$$K_S^0 \rightarrow \pi^+ + \pi^-$$

$$\Lambda\,(\overline{\Lambda}) \rightarrow p\,(\overline{p}) + \pi^-\,(\pi^+)$$

DCA - distance of closest approach

PA - pointing angle

$$M^{\text{inv}} = \sqrt{\left(\sum_i E_i\right)^2 - \left|\sum_i \mathbf{p}_i\right|^2}$$

PV – primary vertex

SV – secondary vertex

Step 1: topological selections

```
// Configurable parameters for V0 selection
Configurable<float> v0setting_dcav0dau{"v0setting_dcav0dau", 1, "DCA V0 Daughters"};
Configurable<float> v0setting_dcapostopv{"v0setting_dcapostopv", 0.06, "DCA Pos To PV"};
Configurable<float> v0setting_dcanegtopv{"v0setting_dcanegtopv", 0.06, "DCA Neg To PV"};
Configurable<double> v0setting_cospa{"v0setting_cospa", 0.98, "V0 CosPA"};
Configurable<float> v0setting_radius{"v0setting_radius", 0.5, "v0radius"};
```

define dedicated **configurables**

```
// Filters on V0s
// Cannot filter on dynamic columns, so we cut on DCA to PV and DCA between daughters only
Filter preFilterV0 = (nabs(aod::v0data::dcapostopv) > v0setting_dcapostopv &&
                      nabs(aod::v0data::dcanegtopv) > v0setting_dcanegtopv &&
                      aod::v0data::dcaV0daughters < v0setting_dcav0dau);
```

pre-Filter on **regular** columns

```
    // Cut on dynamic columns
    if (v0.v0cosPA(collision.posX(), collision.posY(), collision.posZ()) < v0setting_cospa)
      continue;
    if (v0.v0radius() < v0setting_radius)
      continue;
```

cut on **dynamic** columns in the loop

Step 1: topological selections

```
// Configurable
Configurable<flo
Configurable<flo
Configurable<flo
Configurable<dou
Configurable<flo
```

Make sure to add **short** but **informative comments** to your code not to forget what you've done!

...e dedicated
...figurables

```
// Filters on V0
// Cannot filter
Filter preFilter
```

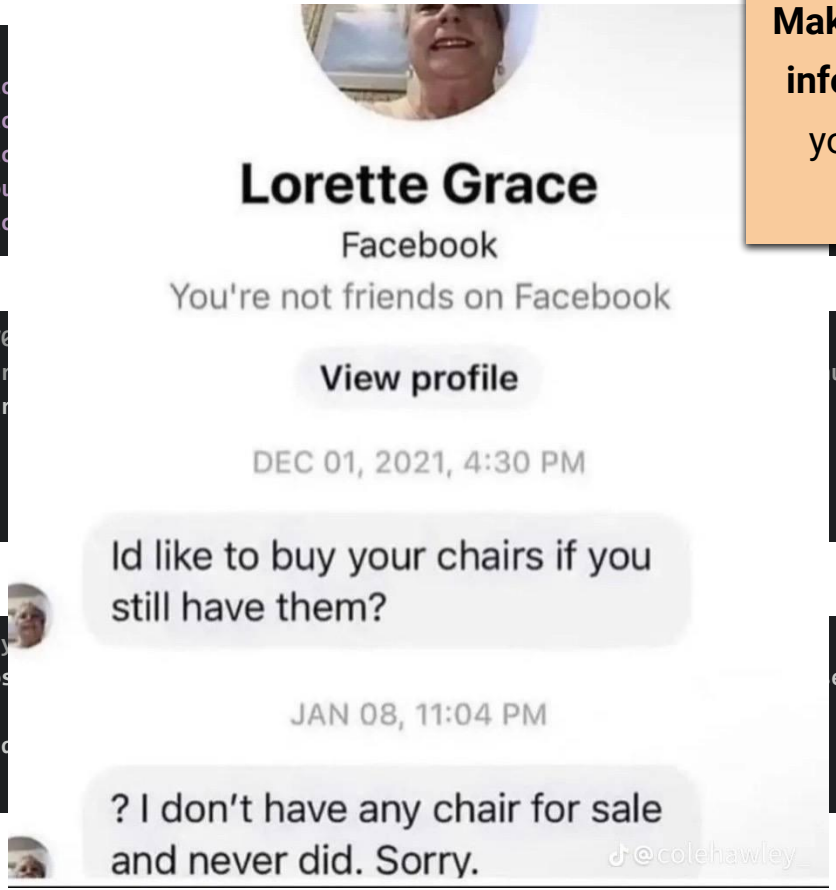
...ughters only

pre-Filter on **regular** columns

```
// Cut on dy
if (v0.v0cos
  continue;
if (v0.v0rad
  continue;
```

...etting_cospa)

cut on **dynamic** columns in the loop



Step 1: topological selections

- **Topological** selections are also applied when building V0 candidates!

```
"lambdakzero-builder": {
        "dcav0dau": "1",
        "dcapostopv": "0.1",
        "dcanegtopv": "0.1",
        "v0cospa": "0.995",
        "v0radius": "0.9"
}
```

```
"strangeness_tutorial": {
    "v0setting_dcav0dau": "1",
    "v0setting_dcapostopv": "0.06",
    "v0setting_dcanegtopv": "0.06",
    "v0setting_cospa": "0.98",
    "v0setting_radius": "0.5"
}
```

Step 1: topological selections

- **Topological** selections are also applied when building V0 candidates!

```
"lambdakzero-builder": {
        "dcav0dau": "1",
        "dcapostopv": "0.1",
        "dcanegtopv": "0.1",
        "v0cospa": "0.995",
        "v0radius": "0.9"
}
```

```
"strangeness_tutorial": {
    "v0setting_dcav0dau": "1",
    "v0setting_dcapostopv": "0.06",
    "v0setting_dcanegtopv": "0.06",
    "v0setting_cospa": "0.98",
    "v0setting_radius": "0.5"
}
```

- Carefully check that in the **lambdakzero-builder** configuration the selections are looser or equal to those you apply in your task

Step 1: topological selections

- **Topological** selections are also applied when building V0 candidates!

The variables must have these names!

```
"lambdakzero-builder": {
        "dcav0dau": "1",
        "dcapostopv": "0.1",
        "dcanegtopv": "0.1",
        "v0cospa": "0.995",
        "v0radius": "0.9"
        "d_UseAutodetectMode": "true"
}
```

```
"strangeness_tutorial": {
    "v0setting_dcav0dau": "1",
    "v0setting_dcapostopv": "0.06",
    "v0setting_dcanegtopv": "0.06",
    "v0setting_cospa": "0.98",
    "v0setting_radius": "0.5"
}
```

- Carefully check that in the **lambdakzero-builder** configuration the selections are looser or equal to those you apply in your task
- Smarter way is to use the **d_UseAutodetectMode** mode of the **lambdakzero-builder**! The builder will take as input values those chosen in your analysis task (or the loosest ones among several tasks where the cuts are present)

Step 1: topological selections

- **Topological** selections are also applied when building V0 candidates!

The variables must have these names!

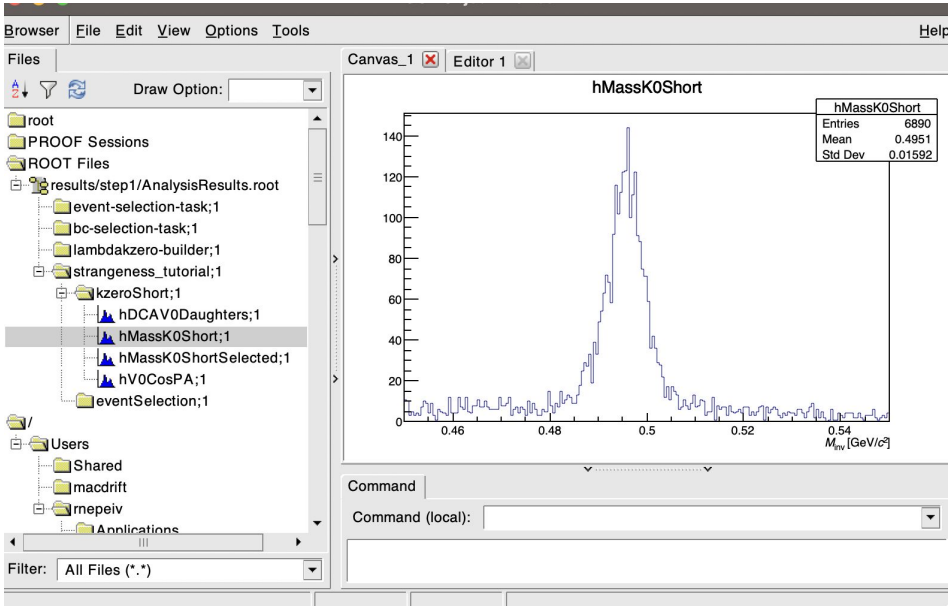
```
"lambdakzero-builder": {
    "dcav0da
    "dcapost
    "dcanegt
    "v0cospa
    "v0radius
    "d_UseAu
}
```

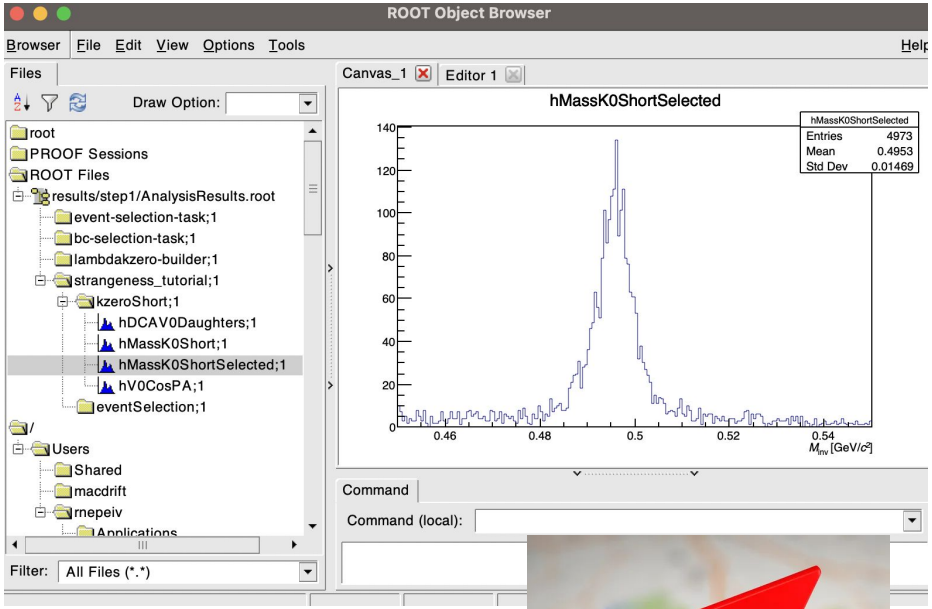
```
. {
au": "1",
opv": "0.06",
opv": "0.06",
"0.98",
"0.5"
}
```

Try to **play with** different **configurations** and see how the invariant mass histogram changes after applying the selections in your task!

- Carefully check that in the **lambdakzero-builder** configuration the selections are looser or equal to those you apply in your task
- Smarter way is to use the **d_UseAutodetectMode** mode of the **lambdakzero-builder**! The builder will take as input values those chosen in your analysis task (or the loosest ones among several tasks where the cuts are present)

Step 1: topological selections





- without autodetection the number of selected kaons should be less than it was before the selections
- with autodetection it should be exactly the same (why?)

Step 2: PID selections

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

```
// Defining the type of the daughter tracks
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod::pidTPCPi>;

void process(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
             soa::Filtered<aod::V0Datas> const& V0s,
             DaughterTracks const&)
{
  // Fill the event counter
  rEventSelection.fill(HIST("hVertexZRec"), collision.posZ());

  for (const auto& v0 : V0s) {

    const auto& posDaughterTrack = v0.posTrack_as<DaughterTracks>();
    const auto& negDaughterTrack = v0.negTrack_as<DaughterTracks>();

    rKzeroShort.fill(HIST("hMassK0Short"), v0.mK0Short());

    // Cut on dynamic columns
    if (v0.v0cosPA(collision.posX(), collision.posY(), collision.posZ()) < v0setting_cospa)
      continue;
    if (v0.v0radius() < v0setting_radius)
      continue;

    if (TMath::Abs(posDaughterTrack.tpcNSigmaPi()) > NSigmaTPCPion) {
      continue;
    }
    if (TMath::Abs(negDaughterTrack.tpcNSigmaPi()) > NSigmaTPCPion) {
      continue;
    }
```

define dedicated **configurable**

subscribe to **aod::pidTPCPi** table

typecast v0 to daughter track type in order to obtain PID information of daughters

cut on number of TPC σ-s

Step 2: PID selections

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

```
// Defining the type of the daughter tracks
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod:

void process(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::i
             soa::Filtered<aod::V0Datas> const& V0s,
             DaughterTracks const&)
{
  // Fill the event counter
  rEventSelection.fill(HIST("hVertexZRec")
  for (const auto& v0 : V0s) {

    const auto& posDaughterTrack = v0.pos
    const auto& negDaughterTrack = v0.neg

    rKzeroShort.fill(HIST("hMassK0Short")

    // Cut on dynamic columns
    if (v0.v0cosPA(collision.posX(), coll
      continue;
    if (v0.v0radius() < v0setting_radius)
      continue;

    if (TMath::Abs(posDaughterTrack.tpcNS
      continue;
    }
    if (TMath::Abs(negDaughterTrack.tpcNSigmaPi()) > NSigmaTPCPion) {
      continue;
    }
```

Header needed:
#include "Common/DataModel/PIDResponse.h"

define dedicated
able

e to
aod::pidTPCPi table

- o2-analysis-timestamp
- o2-analysis-event-selection
- o2-analysis-bc-converter
- o2-analysis-tracks-extra-converter
- **o2-analysis-pid-tpc**
- **o2-analysis-pid-tpc-base**
- **o2-analysis-track-propagation**
- o2-analysis-lf-lambdakzerobuilder
- o2-analysistutorial-lf-strangeness-step2

typecast v0 to daughter track type in order to obtain PID information of daughters

cut on number of TPC σ-s

Step 2: accessing daughter tracks

- The V0 table (aod::V0Data) references a track table that does not contain all possibly useful track information for the analysis, for example it does not contain PID information
- → if you try to check PID of daughter tracks with reference to that table the code will fail!

```
float nsigma_neg = v0.negTrack.tpcNSigmaPi();
float nsigma_pos = v0.posTrack.tpcNSigmaPi();
```

Fails!

- You need to **de-reference the track** via a "***_as<>***" call, this allows you to look at the same index position in the more complete track table

```
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod::pidTPCPi>;

float nsigma_neg = v0.negTrack_as<DaughterTracks>().tpcNSigmaPi();
float nsigma_pos = v0.posTrack_as<DaughterTracks>().tpcNSigmaPi();
```

Works!

Step 2: PID information

```
// Filling the PID of the V0 daughters in the region of the K0 peak
if (0.45 < v0.mK0Short() && v0.mK0Short() < 0.55) {
  rKzeroShort.fill(HIST("hNSigmaPosPionFromK0s"), posDaughterTrack.tpcNSigmaPi(), posDaughterTrack.tpcInnerParam());
  rKzeroShort.fill(HIST("hNSigmaNegPionFromK0s"), negDaughterTrack.tpcNSigmaPi(), negDaughterTrack.tpcInnerParam());
}
```

- Try to fill the histograms with the PID information of the kaon candidate's daughters

What is **tpcInnerParam()** ?

Look it up in the documentation!

Hint is to look at the **o2::aod::TracksExtra**

Step 3: access MC information of V0-s

```
// Defining the type of the daughter tracks
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod::pidTPCPi, aod::McTrackLabels>;

void processRecMC(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::V0Datas, aod::McV0Labels>> const& V0s,
                  DaughterTracks const&,
                  aod::McParticles const&)
```

Subscribe to **aod::McTrackLabels** and **aod::McParticles** tables

```
if (posDaughterTrack.has_mcParticle() && negDaughterTrack.has_mcParticle()) {
  auto posParticle = posDaughterTrack.mcParticle();
  auto negParticle = negDaughterTrack.mcParticle();
  if (posParticle.pdgCode() == 211 && negParticle.pdgCode() == -211) {
    rKzeroShort.fill(HIST("hMassK0ShortSelectedTruePions"), v0.mK0Short());
  }
}

// Checking that the V0 is a true K0s
if (v0.has_mcParticle()) {
  auto v0mcParticle = v0.mcParticle();
  if (v0mcParticle.pdgCode() == 310) {
    rKzeroShort.fill(HIST("hMassK0ShortTrueRec"), v0.mK0Short());
    rKzeroShort.fill(HIST("hPtK0ShortTrueRec"), v0.pt());
  }
}
```

Check that tracks come from particles and are not fake via **has_mcParticle()**

Then, access associated generated particle and compare pdg code

Step 3: access MC information of V0-s

```
// Defining the type of the daughter tracks
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod::pidTPCPi, aod::McTrackLabels>;

void processRecMC(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::V0Datas, aod::McV0Labels>> const& V0s,
                  DaughterTracks const&,
                  aod::McParticles const&)
```

Subscribe to **aod::McTrackLabels** and **aod::McParticles** tables

```
if (posDaughterTrack.has_mcParticle() && negDaughterTrack.has_mcParticle()) {
  auto posParticle = posDaughterT
  auto negParticle = negDaughterT
  if (posParticle.pdgCode() == 21
    rKzeroShort.fill(HIST("hMassK
  }
}

// Checking that the V0 is a true
if (v0.has_mcParticle()) {
  auto v0mcParticle = v0.mcPartic
  if (v0mcParticle.pdgCode() == 3
    rKzeroShort.fill(HIST("hMassK
    rKzeroShort.fill(HIST("hPtK0S
  }
}
```

- o2-analysis-timestamp
- o2-analysis-event-selection
- o2-analysis-bc-converter
- o2-analysis-tracks-extra-converter
- o2-analysis-pid-tpc
- o2-analysis-pid-tpc-base
- o2-analysis-track-propagation
- o2-analysis-lf-lambdakzerobuilder
- **o2-analysis-lf-lambdakzerolabelbuilder**
- o2-analysistutorial-lf-strangeness-step2

Check that tracks come from particles and are not fake via **has_mcParticle()**

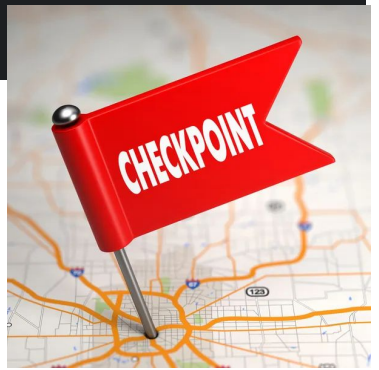
Then, access associated generated particle and compare pdg code

Step 3: iterate over generated MC collisions

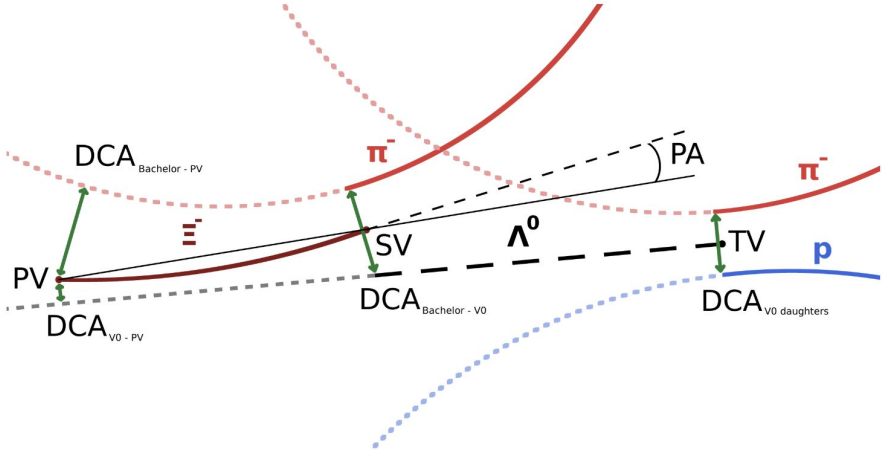
```
void processGenMC(soa::Filtered<aod::McCollisions>::iterator const& mcCollision,
                  const soa::SmallGroups<soa::Join<o2::aod::Collisions, o2::aod::McCollisionLabels, o2::aod::EvSels>>& collisions,
                  aod::McParticles const& mcParticles)
{
  if (collisions.size() < 1) // to process generated collisions that've been reconstructed at least once
    return;
  rEventSelection.fill(HIST("hVertexZGen"), mcCollision.posZ());
  for (const auto& mcParticle : mcParticles) {
    if (mcParticle.pdgCode() == 310) {
      rGenParticles.fill(HIST("hPtK0ShortGen"), mcParticle.pt());
    }
  }
}

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

use **soa::SmallGroups** to loop only over generated collisions that have been reconstructed at least once!

add **PROCESS_SWITCH** to allow for 2 process functions: for generated and reconstructed MC levels



Step 4: loop over cascade candidates



$$\Xi^- (\overline{\Xi}^+) \rightarrow \Lambda (\overline{\Lambda}) + \pi^- (\pi^+)$$
$$\Omega^- (\overline{\Omega}^+) \rightarrow \Lambda (\overline{\Lambda}) + \mathrm{K}^- (\mathrm{K}^+)$$

DCA - distance of closest approach

PA - pointing angle

$$M^{\mathrm{inv}} = \sqrt{\left(\sum_i E_i\right)^2 - \left|\sum_i \mathbf{p}_i\right|^2}$$

PV – primary vertex

SV – secondary vertex

TV - tertiary vertex

Step 4: loop over cascade candidates

- Create an instance of a PDG data base (outside init)

Header needed:
#include "Framework/O2DatabasePDGPlugin.h"

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

- Create a histogram registry for Xi candidates

```
HistogramRegistry rXi{"xi", {}, OutputObjHandlingPolicy::AnalysisObject, true, true};
```

- Declare configurables for the topological selections

```
// Configurable parameters for cascade selection
Configurable<double> cascadesetting_cospa{"cascadesetting_cospa", 0.98, "Casc CosPA"};
Configurable<float> cascadesetting_dcacascdau{"cascadesetting_dcacascdau", 1.0, "DCA cascade daughters"};
Configurable<float> cascadesetting_dcabachtopv{"cascadesetting_dcabachtopv", 0.1, "DCA bachelor 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_v0masswindow{"cascadesetting_v0masswindow", 0.01, "v0 mass window"};
Configurable<float> cascade_dcanegtopv{"dcanegtopv", 0.06, "DCA Neg To PV"};
Configurable<float> cascade_dcapostopv{"dcapostopv", 0.06, "DCA Pos To PV"};
```

Step 4: loop over cascade candidates

- Add several invariant mass and topo histograms for cascades:

```
AxisSpec XiMassAxis = {200, 1.28f, 1.36f, "#it{M}_{inv} [GeV/#it{c}^{2}]"};

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

```
// Xi topological cuts
rXi.add("hCascDCAV0Daughters", "hCascDCAV0Daughters", {HistType::kTH1F, {{55, 0.0f, 2.2f}}});
rXi.add("hCascCosPA", "hCascCosPA", {HistType::kTH1F, {{100, 0.95f, 1.f}}});
```

- Define configurable for PID selection of protons

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

Step 4: loop over cascade candidates

```
void processRecMC(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::CascDatas, aod::McCascLabels>> const& Cascades,
                  soa::Filtered<soa::Join<aod::V0Datas, aod::McV0Labels>> const& V0s,
                  aod::V0Datas const&,
                  aod::V0sLinked const&,
                  DaughterTracks const&,
                  aod::McParticles const&)
{
```

Subscribe to **aod::CascDatas** and **aod::V0sLinked** tables

```
// Filters on Cascades
Filter preFilterCascades = (nabs(aod::cascdata::dcabachtopv) > cascadesetting_dcabachtopv &&
                            aod::cascdata::dcaV0daughters < v0setting_dcav0dau &&
                            nabs(aod::cascdata::dcapostopv) > cascade_dcapostopv &&
                            nabs(aod::cascdata::dcanegtopv) > cascade_dcanegtopv &&
                            nabs(aod::cascdata::dcabachtopv) > cascadesetting_dcabachtopv &&
                            aod::cascdata::dcacascdaughters < cascadesetting_dcacascdau);
```

pre-Filter on **regular** columns

Step 4: loop over cascade candidates

```
void processRecMC(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::CascDatas, aod::McCascLabels>> const& Cascades,
                  soa::Filtered<soa::Join<aod::V0Datas, aod::McV0Labels>> const& V0s,
                  aod::V0Datas const&,
                  aod::V0sLinked const&,
                  DaughterTracks const&,
                  aod::McParticles const&)
{
```

Subscribe to a full **aod::V0Datas** table in case you apply different selection on primary and secondary V0-s

Subscribe to **aod::CascDatas** and **aod::V0sLinked** tables

```
//
Fil                                achtopv) > cascadesetting_dcabachtopv &&
                    aod::cascdata::dcaV0daughters < v0setting_dcav0dau &&
                    nabs(aod::cascdata::dcapostopv) > cascade_dcapostopv &&
                    nabs(aod::cascdata::dcanegtopv) > cascade_dcanegtopv &&
                    nabs(aod::cascdata::dcabachtopv) > cascadesetting_dcabachtopv &&
                    aod::cascdata::dcacascdaughters < cascadesetting_dcacascdau);
```

Step 4: loop over cascade candidates

```
void processRecMC(soa::Filtered<soa::Join<aod::Collisions, aod::EvSels>>::iterator const& collision,
                  soa::Filtered<soa::Join<aod::CascDatas, aod::McCascLabels>> const& Cascades,
                  soa::Filtered<soa::Join<aod::V0Datas, aod::McV0Labels>> const& V0s,
                  aod
                  aod
                  Dau
                  aod
{
```

Subscribe to a f
you apply differ
s

- o2-analysis-timestamp
- o2-analysis-event-selection
- o2-analysis-bc-converter
- o2-analysis-tracks-extra-converter
- o2-analysis-pid-tpc
- o2-analysis-pid-tpc-base
- o2-analysis-track-propagation
- o2-analysis-lf-lambdakzerobuilder
- o2-analysis-lf-lambdakzerolabelbuilder
- **o2-analysis-lf-cascadebuilder**
- **o2-analysis-lf-cascadelabelbuilder**
- o2-analysistutorial-lf-strangeness-step2

Subscribe to
aod::CascDatas and
aod::V0sLinked tables

```
//
Fil                                                    ting_dcabachtopv &&
                                                       v0dau &&
                                                       postopv &&
                                                       negtopv &&
                                                       ting_dcabachtopv &&
                                                       ting_dcacascdau);
```

Step 4: loop over cascade candidates

```
const auto& v0index = casc.v0_as<aod::V0sLinked>();
if (!(v0index.has_v0Data())) {
  continue; // skip those cascades for which V0 doesn't exist
}

const auto& v0Casc = v0index.v0Data(); // de-reference index to correct v0data in case it exists
const auto& bachDaughterTrackCasc = casc.bachelor_as<DaughterTracks>();
const auto& posDaughterTrackCasc = v0Casc.posTrack_as<DaughterTracks>();
const auto& negDaughterTrackCasc = v0Casc.negTrack_as<DaughterTracks>();
```

Dereference index to correct **V0Data**

typecast V0 and cascade to our daughter track type (note **aod::pidTPCPr**)

```
// Defining the type of the daughter tracks
using DaughterTracks = soa::Join<aod::TracksIU, aod::TracksExtra, aod::pidTPCPi, aod::pidTPCPr, aod::McTrackLabels>;
```

Step 4: loop over cascade candidates

```
// Cut on dynamic columns
if (casc.casccosPA(collision.posX(), collision.posY(), collision.posZ()) < cascadesetting_cospa)
  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;
```

Cut on **dynamic** columns inside the loop

Access the PDG mass via the **DPG data base**

Step 4: loop over cascade candidates

```
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;
  }
}
if (TMath::Abs(bachDaughterTrackCasc.tpcNSigmaPi()) > NSigmaTPCPion) {
  continue;
}
```

Select the daughter tracks via the PID

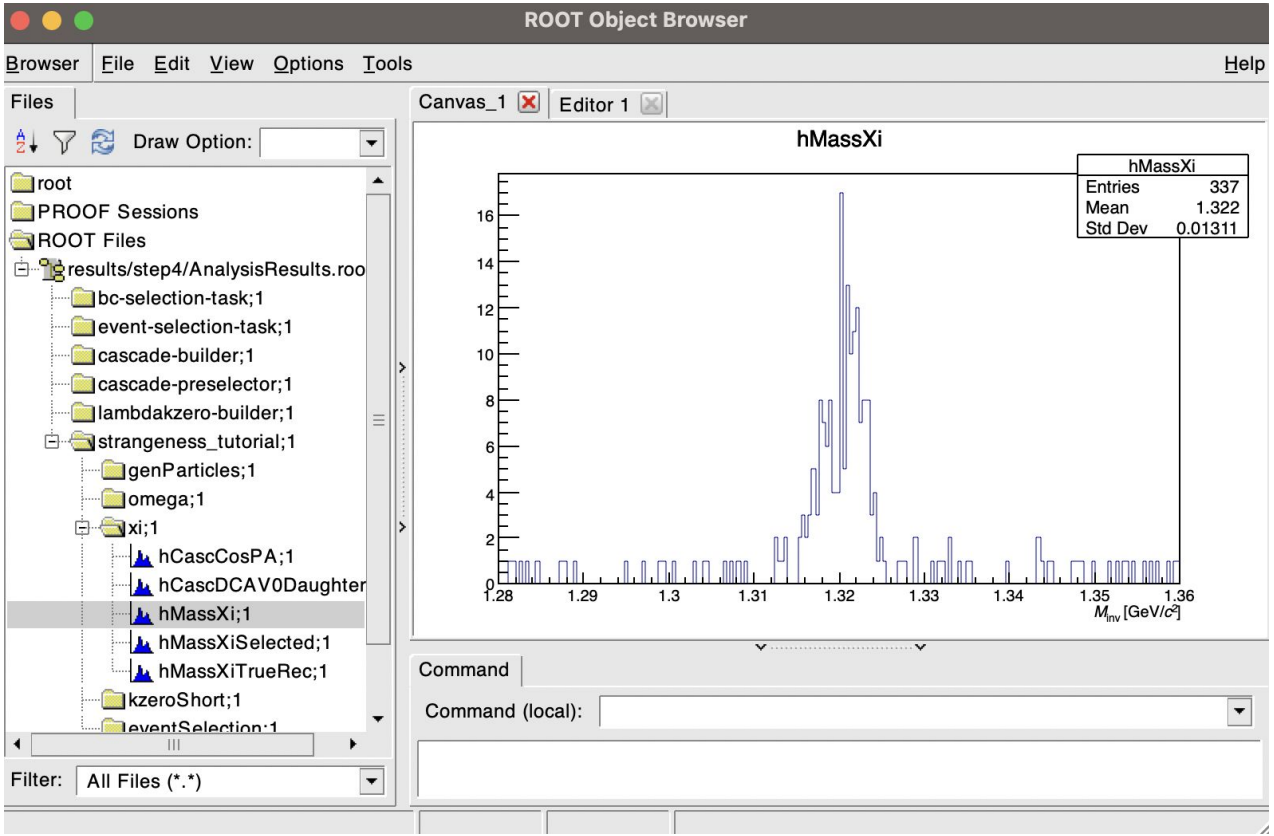
Step 4: loop over cascade candidates

```
// Checking that the cascade is a true Xi
if (casc.has_mcParticle()) {
  const auto cascmcParticle = casc.mcParticle();
  if (TMath::Abs(cascmcParticle.pdgCode()) == 3312) {
    rXi.fill(HIST("hMassXiTrueRec"), casc.mXi());
  }
}
```

Check that cascade came from particle that is not fake via **has_mcParticle()**

Then, access associated generated particle and compare pdg code

Step 4: loop over cascade candidates

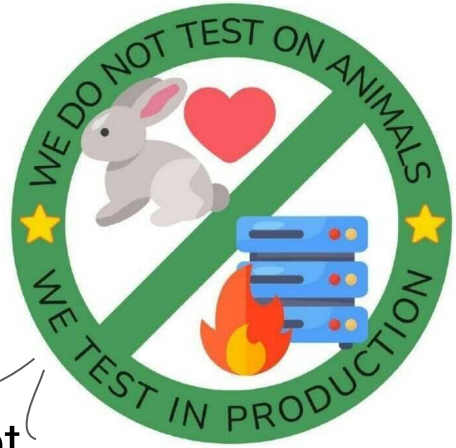


You must observe a nice invariant mass peak!

Summary

In this hands-on session you learned:

- **strangeness tables** and how to **subscribe** to them in your analysis
- **apply simple** topological and kinematic **selections**
- **apply PID** selections
- perform an **analysis on MC**



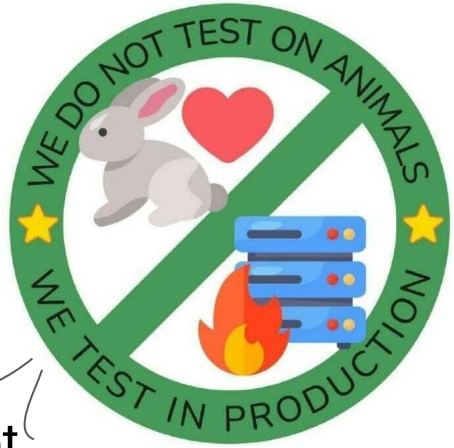
*do not

Summary

In this hands-on session you learned:

- **strangeness tables** and how to **subscribe** to them in your analysis
- **apply simple** topological and kinematic **selections**
- **apply PID** selections
- perform an **analysis on MC**

Thank You



*do not

Additional materials

Analysis Level Tables: aod::V0Data(Ext)

#include "PWGLF/DataModel/LFStrangenessTables.h"

	Name	Getter
Bound	v0::PosTrackId	posTrackId
	v0::NegTrackId	negTrackId
	v0::CollisionId	collisionId
	v0::V0	v0Id
Calculated	v0data::PxPos	pxpos
	v0data::PyPos	pypos
	v0data::PzPos	pzpos
	v0data::PxNeg	pxneg
	v0data::PyNeg	pyneg
	v0data::PzNeg	pxneg
	v0data::X	x
	v0data::Y	y
	v0data::Z	z
	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::V0Data(Ext)

#include "PWGLF/DataModel/LFStrangenessTables.h"

	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: Decay vertex position	v0data::X	x
	v0data::Y	y
	v0data::Z	z
Topological variables	v0data::DCAV0Daughters	dcav0daughters
	v0data::DCAPosToPV	dcapostopv
	v0data::DCANegToPV	dcanegtopv

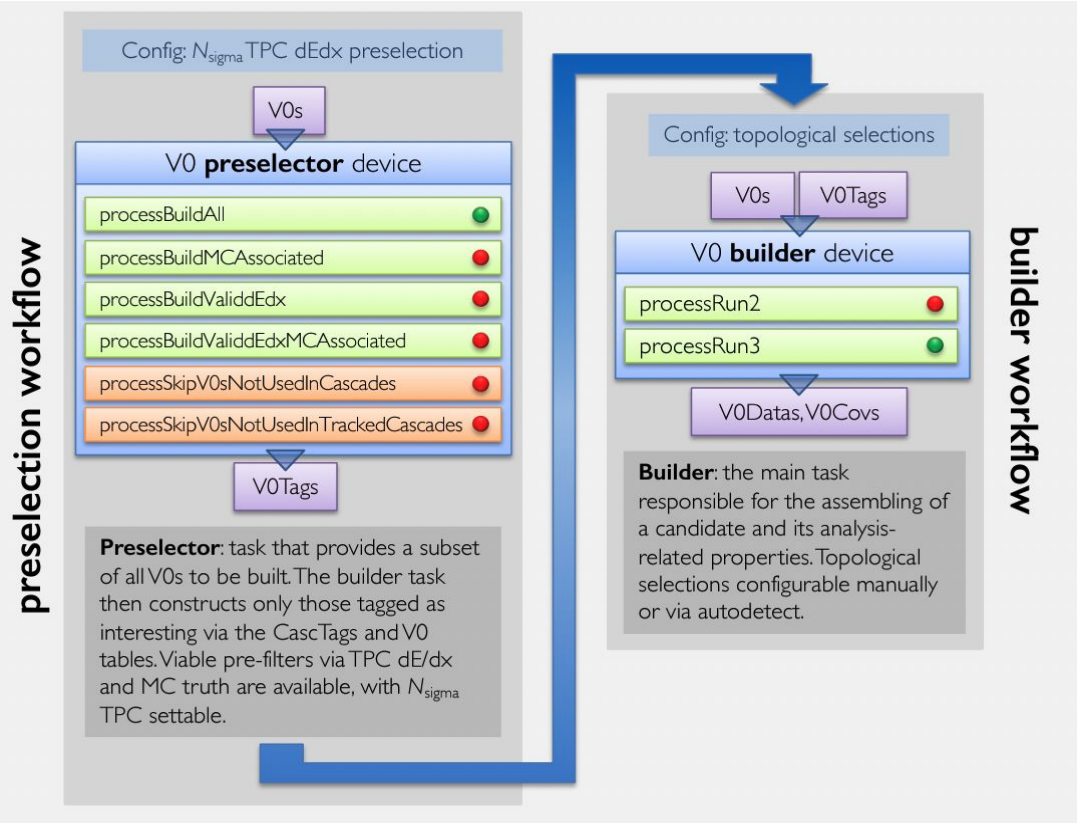
Derived (dynamic)	v0data::Pt	pt
	v0data::V0Radius	v0radius
	v0data::V0CosPA	v0cospa
	v0data::DCAV0ToPV	dcav0topv
	v0data::MLambda	mLambda
	v0data::MAntiLambda	mAntiLambda
	v0data::MK0Short	mK0Short
	v0data::YK0Short	yK0Short
	v0data::YLambda	yLambda
	v0data::Eta	eta
	v0data::Phi	phi
	...	...
Derived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

Analysis Level Tables: aod::CascData(Ext)

#include "PWGLF/DataModel/LFStrangenessTables.h"

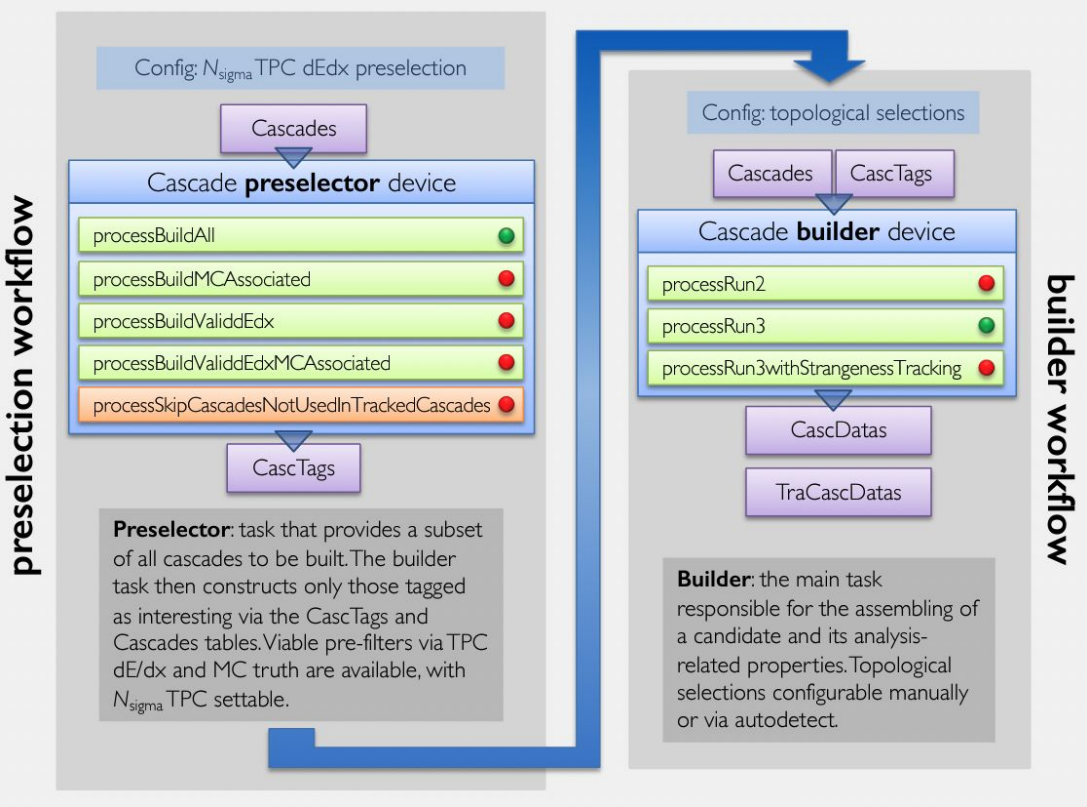
	Name	Getter
Bound	cascade::V0Id	v0Id
	cascade::BachelorId	bachelorId
	cascade::CollisionId	collisionId
	cascdata::Sign	sign
Momenta of daughter tracks	cascdata::PxPos(Neg)	pxpos(neg)
	cascdata::PyPos(Neg)	pypos(neg)
	cascdata::PzPos(Neg)	pzpos(neg)
	cascdata::PxBach	pxbach
	cascdata::PyBach	pybach
	cascdata::PzBach	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
Derived (dynamic)	cascdata::Pt	pt
	cascdata::V0Radius	v0radius
	cascdata::CascRadius	cascradius
	cascdata::V0CosPA	v0cosPA
	cascdata::CascCosPA	casccosPA
	cascdata::MLambda	mLambda
	cascdata::MXi	mXi
	cascdata::MOmega	mOmega
	cascdata::YXi	yXi
	cascdata::YOmega	yOmega
	...	...
Derived (expression)	cascdataext::Px	px
	cascdataext::Py	py
	cascdataext::Pz	pz
	...	...

V0 table producer task



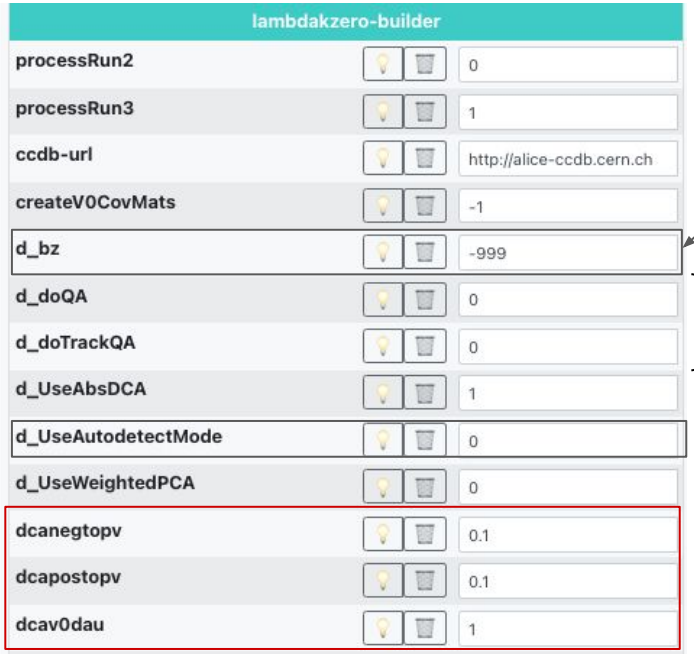
- Preselection scheme is implemented following the “mask” logic to reduce the CPU usage, make sure to use it if possible in your analysis
- Preselection usage:
  - Mandatory: choose one **processBuild**
  - Optional: choose one of the **SkipV0s**
- If this isn’t followed, expect a printout to explain what can and cannot be done!
- The logic is such that an AND of “being used in cascades” and the processBuild conditions will be applied if a **SkipV0** option is used

Cascade table producer task



- Preselection scheme following the “mask” logic is implemented to reduce the CPU usage, make sure to use it if possible in your analysis
- Preselection usage:
  - Mandatory: choose one **processBuild**
  - Optional: skip untracked cascades
- **CascDatas**: regular cascade analysis table
- **TraCascDatas**: tracked cascade analysis table
- Simultaneous generation of regular and tracked cascades:
  processRun3withStrangenessTracking
- Sole generation of regular cascades

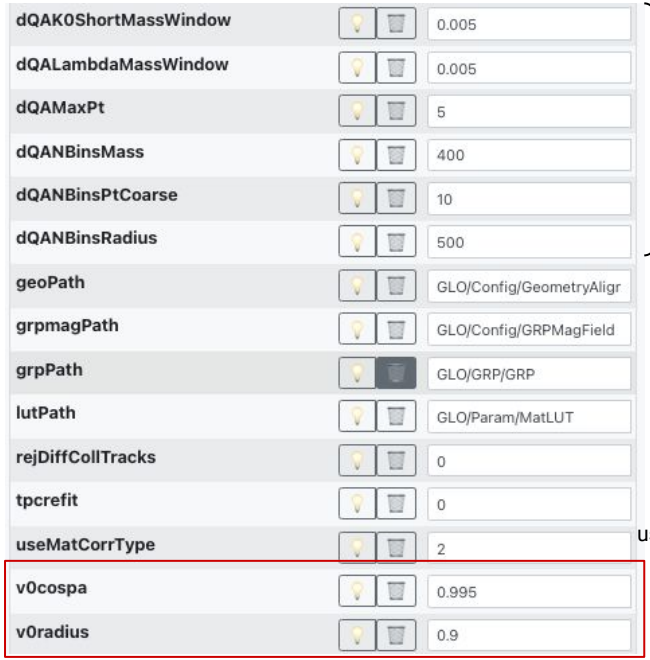
V0 builder configuration



process switch

magnetic field:
if -999, the correct value is automatically retrieved from ccdb

extraQA

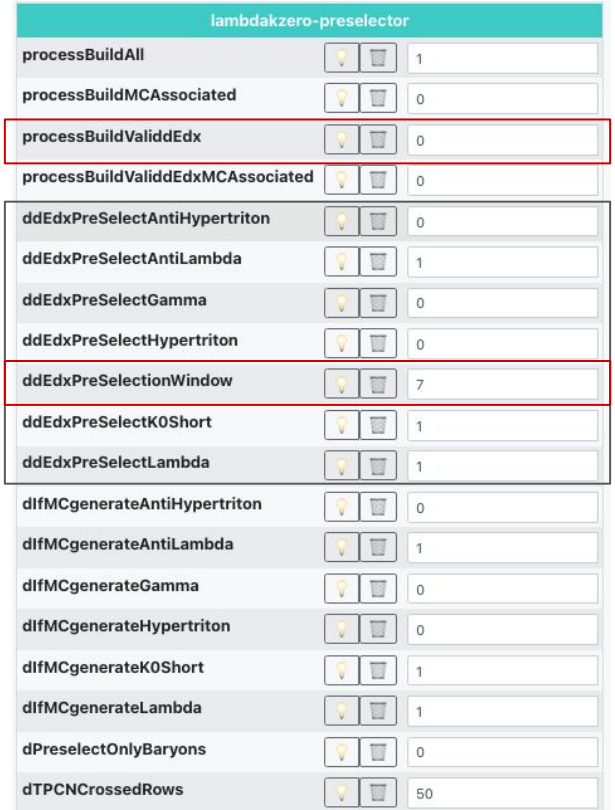


extraQA

use material correction:
0: none, 1: TGeo, 2: LUT

selection on candidates

V0 preselector

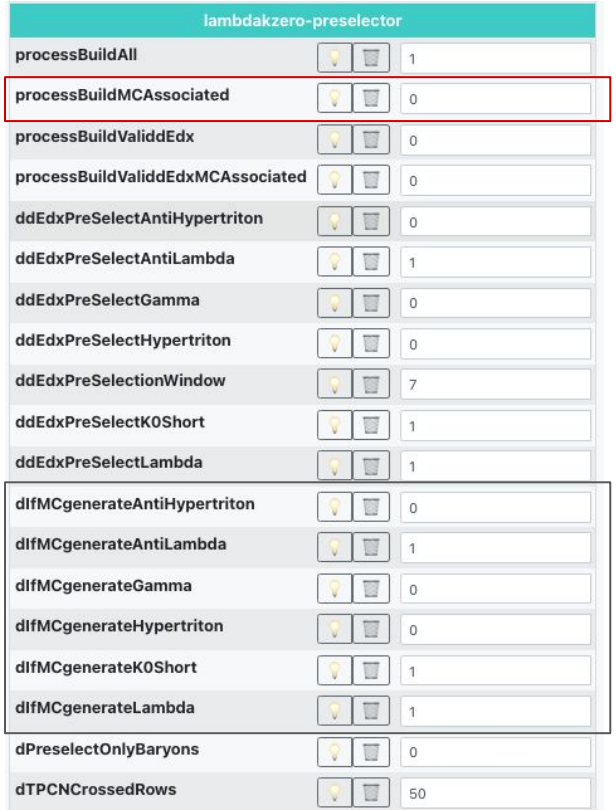


By default, all V0s are built

BuildValiddEdx: only candidates whose daughter tracks are correctly identified in the TPC are built, according to the pre-selection window and to the pre-selection choice

Example:
ddEdxPreSelectWindow = 4
ddEdxPreSelecK0Short = 1 (0 for all others)
→ the candidate is stored in the V0Datas table only if the two daughter tracks have |NSigmaTPC| < 4 in the pion hypothesis

V0 preselector



By default, all V0s are built

BuildMCAssoc: only candidates which correspond to a MC particle (K0s, Lambda,..) are built

Example:
dIfMCgenerateK0Short = 1
→ the candidate is stored in the V0Datas table only if it is a true K0s.