Analysis Framework of Resonance for Run3

Introduction to `LFResonanceInitializer`

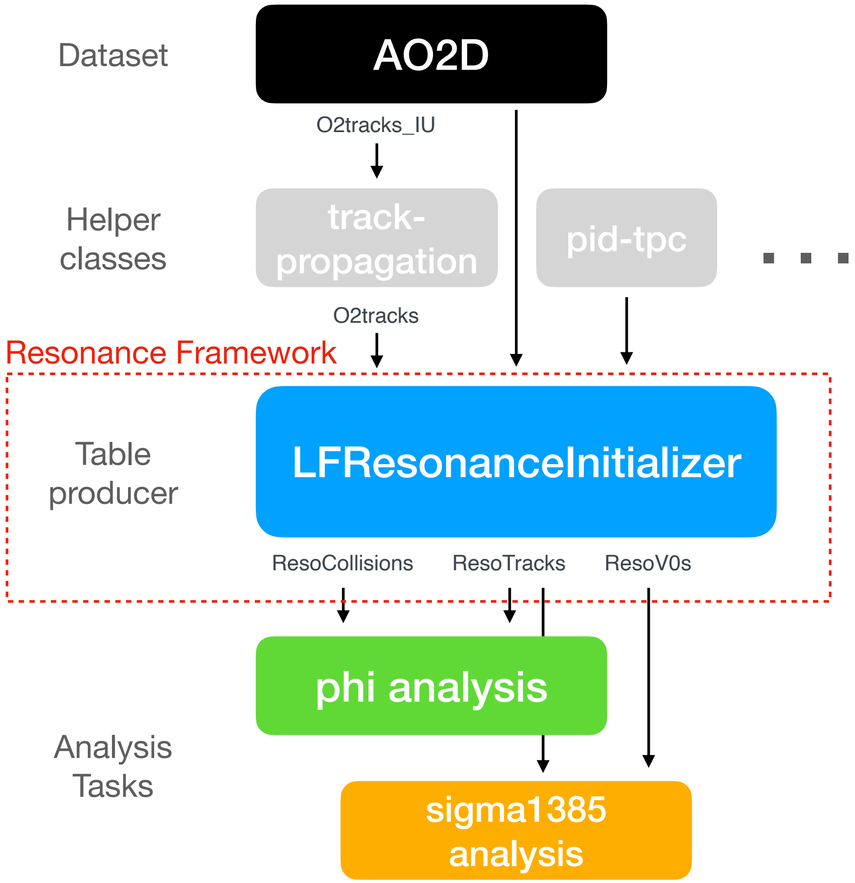
Bong-Hwi Lim (University and INFN Torino)

O2 Analysis Tutorial 3.0

08/11/2023

Analysis framework workflow

Concept: General table producer for whole resonance analyses



- Unlike RUN2 analysis, most of analysis data have to be processed **on the fly.**
  - This process is done by "**Helper class**"
  - ✓ Eg. track propagation for ITS track, Lambda/K0s/Cascade reconstruction
- **Resonance Framework** refines all information needed for resonance analysis
  - Event(collision) information: **ResoCollisions**
  - Daughter(Track/V0/Cascade) information: **ResoTracks/ResoV0s/ ResoCascades**
  - MC information: **ResoMCParents/ResoMCTracks/ResoMCV0s/ ResoMCCascades**
- **Analysis tasks** takes only input from the output of the framework.
  - Having similar shape / common selections with the other resonance analysis.
  - Possibility to process with '**derived dataset**' produced by the framework.

Resonance Initializer

Analysis framework

- Resonance Initializer (`lf-reso2initializer`, O2Physics/PWGLF/TableProducer/LFResonanceInitializer)
  - Main table producer, producing 6 tables:
    - **ResoCollisions**, **ResoTracks/ResoV0s/ResoCascades/ ResoMCParents/ResoMCTracks/ResoMCV0s/ResoMCCascades** MC related tables
    - Table definition: O2Physics/PWGLF/DataModel/LFResonanceTables.h
  - Applying common basic selections: **event cuts, track cuts, v0 selection.**
  - 6 individual processes: *Trk/V0/Cascade w/wo MC*

Run only 1 process

- **processTrackData:** ResoCollisions, ResoTracks
- **processTrackV0Data:** ResoCollisions, ResoTracks, ResoV0s
- **processTrackMC:** ResoCollisions, ResoTracks, ResoMCTracks, ResoMCParents
- **processTrackV0MC:** ResoCollisions, ResoTracks, ResoV0s , ResoMCTracks, ResoMCParents, ResoMCV0s

MC related processes

- Resonance analyses based on decay mode:
  - **Track + Track decay:** K^{*0}, Φ, $\Lambda(1520)$
  - **Track + V0 decay:** $K^{*\pm}$, $\Sigma(1385)^{\pm}$, $\Xi(1820)$
  - Track + Cascade decay: $\Xi(1530)^{0}$, $\Omega(2012)$
  - V0 + V0 decay: $\Xi(1820)$
  - V0 + Cascade decay: $\Omega(2012)$

Resonance Initializer

Analysis framework

- Resonance Initializer (`lf-reso2initializer`, O2Physics/PWGLF/TableProducer/LFResonanceInitializer)
  - Main table producer, producing 6 tables:
    - **ResoCollisions**, **ResoTracks/ResoV0s/ResoCascades/ ResoMCParents/ResoMCTracks/ResoMCV0s/ResoMCCascades** MC related tables
    - Table definition: O2Physics/PWGLF/DataModel/LFResonanceTables.h
  - Applying common basic selections: **event cuts, track cuts, v0 selection.**
  - 6 individual processes: *Trk/V0/Cascade w/wo MC*
    - **processTrackData:** ResoCollisions, ResoTracks
    - **processTrackV0Data:** ResoCollisions, ResoTracks, ResoV0s
    - **processTrackMC:** ResoCollisions, ResoTracks, ResoMCTracks, ResoMCParents
    - **processTrackV0MC:** ResoCollisions, ResoTracks, ResoV0s , ResoMCTracks, ResoMCParents, ResoMCV0s

Run only 1 process

MC related processes

Data processing

MC processing

- Resonance analyses based on decay mode:
  - **Track + Track decay:** K^{*0}, Φ, $\Lambda(1520)$
  - **Track + V0 decay:** $K^{*\pm}$, $\Sigma(1385)^{\pm}$, $\Xi(1820)$
  - Track + Cascade decay: $\Xi(1530)^{0}$, $\Omega(2012)$
  - V0 + V0 decay: $\Xi(1820)$
  - V0 + Cascade decay: $\Omega(2012)$

Resonance Initializer

Analysis framework

- Resonance Initializer (`lf-reso2initializer`, O2Physics/PWGLF/TableProducer/LFResonanceInitializer)
  - Main table producer, producing 6 tables:
    - **ResoCollisions**, **ResoTracks/ResoV0s/ResoCascades/ ResoMCParents/ResoMCTracks/ResoMCV0s/ResoMCCascades** MC related tables
    - Table definition: O2Physics/PWGLF/DataModel/LFResonanceTables.h
  - Applying common basic selections: **event cuts, track cuts, v0 selection.**
  - 6 individual processes: *Trk/V0/Cascade w/wo MC*
    - **processTrackData:** ResoCollisions, ResoTracks
    - **processTrackV0Data:** ResoCollisions, ResoTracks, ResoV0s
    - **processTrackMC:** ResoCollisions, ResoTracks, ResoMCTracks, ResoMCParents
    - **processTrackV0MC:** ResoCollisions, ResoTracks, ResoV0s , ResoMCTracks, ResoMCParents, ResoMCV0s

Run only 1 process

MC related processes

Data processing

MC processing

- Resonance analyses based on decay mode:
  - **Track + Track decay:** K^{*0}, Φ, $\Lambda(1520)$
  - **Track + V0 decay:** $K^{*\pm}$, $\Sigma(1385)^{\pm}$, $\Xi(1820)$
  - Track + Cascade decay: $\Xi(1530)^{0}$, $\Omega(2012)$
  - V0 + V0 decay: $\Xi(1820)$
  - V0 + Cascade decay: $\Omega(2012)$

Resonance Initializer

Data tables

ResoCollisions

```
collision::PosX,
collision::PosY,
collision::PosZ,
resocollision::MultV0M,
resocollision::MultFT0,
resocollision::Spherocity,
resocollision::BMagField,
timestamp::Timestamp
```

- **Basic event information**
  - Vertex positions
  - Multiplicity Percentile
  - Multiplicity
  - Sphericity
  - Magnet information
    Disabled by default
  - Timestamp of the current event

ResoTracks

```
resodaughter::ResoCollisionId,
resodaughter::Pt,
resodaughter::Px,
resodaughter::Py,
resodaughter::Pz,
resodaughter::Eta,
resodaughter::Phi,
resodaughter::Sign,
resodaughter::TPCNClsCrossedRows,
o2::aod::track::DcaXY,
o2::aod::track::DcaZ,
o2::aod::track::X,
o2::aod::track::Alpha,
resodaughter::TPCPIDselectionFlag,
resodaughter::TOFPIDselectionFlag,
o2::aod::pidtpc::TPCNSigmaPi,
o2::aod::pidtpc::TPCNSigmaKa,
o2::aod::pidtpc::TPCNSigmaPr,
o2::aod::pidtof::TOFNSigmaPi,
o2::aod::pidtof::TOFNSigmaKa,
o2::aod::pidtof::TOFNSigmaPr,
o2::aod::track::TPCSignal,
o2::aod::track::PassedITSRefit,
o2::aod::track::PassedTPCRefit,
resodaughter::IsGlobalTrackWoDCA,
resodaughter::IsPrimaryTrack,
resodaughter::IsPVContributor,
resodaughter::TPCCrossedRowsOverFindableCls,
o2::aod::track::ITSChi2NCl,
o2::aod::track::TPCChi2NCl
```

```
mcparticle::PdgCode,
resodaughter::MothersId,
resodaughter::MotherPDG,
resodaughter::SiblingIds,
resodaughter::IsPhysicalPrimary,
resodaughter::ProducedByGenerator
```

ResoMCTracks

ResoV0s

```
resodaughter::ResoCollisionId,
resodaughter::Pt,
resodaughter::Px,
resodaughter::Py,
resodaughter::Pz,
resodaughter::Eta,
resodaughter::Phi,
resodaughter::Indices,
resodaughter::V0CosPA,
resodaughter::DaughDCA,
resodaughter::MLambda,
resodaughter::MAntiLambda,
resodaughter::TransRadius,
resodaughter::DecayVtxX,
resodaughter::DecayVtxY,
resodaughter::DecayVtxZ
```

```
mcparticle::PdgCode,
resodaughter::MothersId,
resodaughter::MotherPDG,
resodaughter::DaughterPDG1,
resodaughter::DaughterPDG2,
resodaughter::IsPhysicalPrimary,
resodaughter::ProducedByGenerator
```

ResoMCV0s

ResoMCParents

```
resodaughter::ResoCollisionId,
resodaughter::McParticleId,
mcparticle::PdgCode,
resodaughter::DaughterPDG1,
resodaughter::DaughterPDG2,
resodaughter::IsPhysicalPrimary,
resodaughter::ProducedByGenerator,
resodaughter::Pt,
resodaughter::Px,
resodaughter::Py,
resodaughter::Pz,
resodaughter::Eta,
resodaughter::Phi,
mcparticle::Y
```

- MC true particles without selection.
- Can be used for counting total number of resonances **to get the reconstruction efficiency**.

Hands-on 1: Execute resonance initializer

Run the script

1. **Download these files**
   - **AO2D file**: https://cernbox.cern.ch/s/GZsMG5ibR8BgXOF
     - or `$alien_cp /alice/data/2022/LHC22o/526860/apass4/2340/o2_ctf_run00526860_orbit0192456960_tf0000122707_epn171/001/AO2D.root file://AO2D_LHC22o_526860_apass4_2340_o2_ctf_run00526860_orbit0192456960_tf0000122707_epn171_001.root`
   - **Configuration file**: https://cernbox.cern.ch/s/vZNZjr5tqnOTQ73
2. **Prepare the input data list**
   - Make a inputdata.txt file and add the input file
     - or `$echo AO2D_LHC22o_526860_apass4_2340_o2_ctf_run00526860_orbit0192456960_tf0000122707_epn171_001.root > inputdata.txt`
3. **Run the task with all helper tasks**
   - **o2-analysis-lf-reso2initializer**
   - o2-analysis-pid-tpc
   - o2-analysis-pid-tof
   - o2-analysis-pid-tof-base
   - o2-analysis-pid-tpc-base

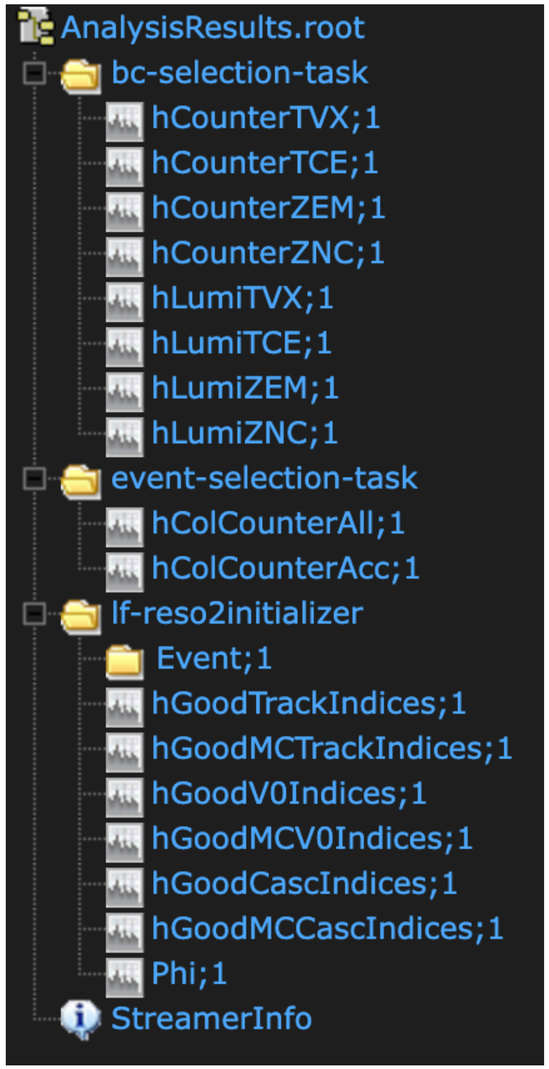
     PID
   - o2-analysis-centrality-table
   - o2-analysis-multiplicity-table

     Centrality
   - o2-analysis-event-selection
   - o2-analysis-ft0-corrected-table
   - o2-analysis-trackselection
   - o2-analysis-tracks-extra-converter
   - o2-analysis-track-propagation
   - o2-analysis-bc-converter (not needed in the 2023 dataset)
   - o2-analysis-timestamp
   - **Bash shell script: https://cernbox.cern.ch/s/IurH2USBmznXWBe**

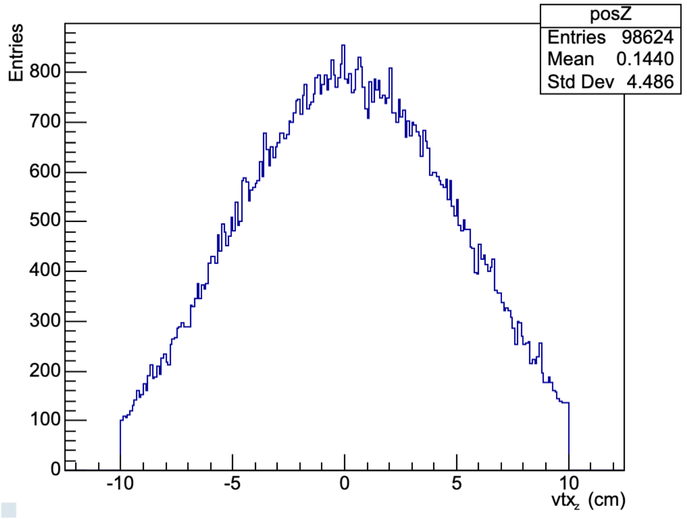
```
o2-analysis-lf-reso2initializer -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tpc -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tof -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tof-base -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tpc-base -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-centrality-table -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-multiplicity-table -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-event-selection -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-ft0-corrected-table -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-trackselection -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-tracks-extra-converter -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-track-propagation -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-bc-converter -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-timestamp -b --configuration json://configuration_resoinitializer_pp.json --aod-file @input_data.txt
```

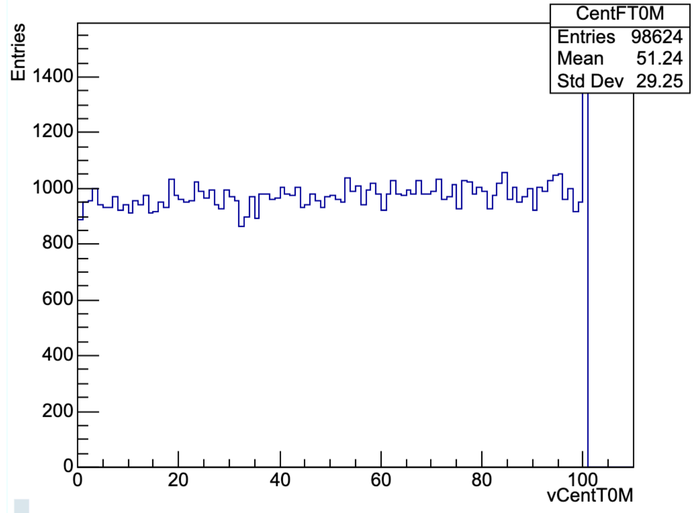
Hands-on 1: Execute resonance initializer

Output



- Inside the `AnalysisResults.root` file...
  - bc-selection-task outputs
  - event-selection-task outputs
  - **lf-reso2initializer outputs**
    - Event
      - Vertex-z position histogram
      - Centrality from various estimators
      - Multiplicity (signal intensity) from T0A/C/M
    - QA outputs
      only if we select `ConfFillQA` option





Hands-on 2: Produce resonance table

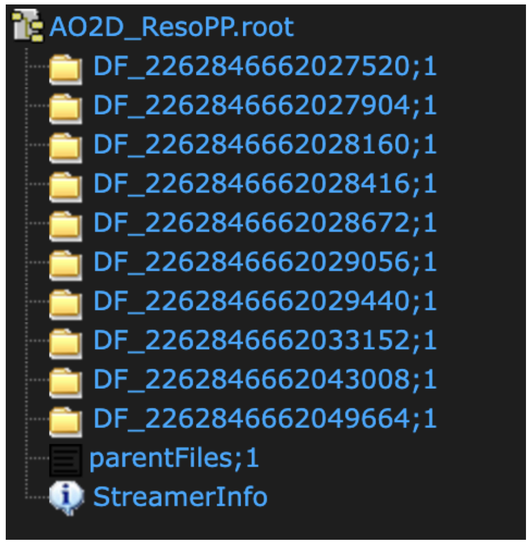
- **Same as before, but adding this option to the `o2-analysis-lf-reso2initializer`**
  - `--aod-writer-resfile AO2D_ResoPP  --aod-writer-keep dangling`
  - Bash shell script: https://cernbox.cern.ch/s/YpkCY8ABmSbvvAk
  - Detailed Tutorial: https://indico.cern.ch/event/1326201/timetable/#16-using-derived-data-in-o2phy

```
o2-analysis-lf-reso2initializer -b --configuration json://configuration_resoinitializer_pp.json --aod-writer-resfile AO2D_ResoPP  --aod-writer-keep dangling
| o2-analysis-pid-tpc -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tof -b --configuration json://configuration_resoinitializer_pp.json |
o2-analysis-pid-tof-base -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-pid-tpc-base -b --configuration json://
configuration_resoinitializer_pp.json | o2-analysis-centrality-table -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-multiplicity-table -b --
configuration json://configuration_resoinitializer_pp.json | o2-analysis-event-selection -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-ft0-
corrected-table -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-trackselection -b --configuration json://configuration_resoinitializer_pp.json |
o2-analysis-tracks-extra-converter -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-track-propagation -b --configuration json://
configuration_resoinitializer_pp.json | o2-analysis-bc-converter -b --configuration json://configuration_resoinitializer_pp.json | o2-analysis-timestamp -b --configuration
json://configuration_resoinitializer_pp.json --aod-file @input_data.txt
```

Hands-on 2: Produce resonance table

Output

```
514M Nov  8 00:23 AO2D_LHC22o_526860_apass4_2340_o2_ctf_run00526860_orbit0192456960_tf0000122707_epn171_001.root
 35M Nov  8 00:23 AO2D_ResoPP.root
```



- **Size comparison**
  - For given pp 13.6 TeV dataset
    - **514 MB → 35 MB** (**~7%** of original)
  - Case of Pb-Pb 5.36 TeV
    - **783 MB → 27 MB** (**~3.5%** of original)

  ✓ Compression rate can be affected based on the selection cut applied.
- **Inside the derived dataset:**
  - One can find the **produced tables!**

Hands-on 3: Using the derived data

- **Start from the previous output**
  - or download from here: **https://cernbox.cern.ch/s/8o0oJpMUbjoC9kM**
- **First task:** `O2Physics/Tutorials/PWGLF/Resonance/resonances_step0.cxx`
  - bash script & configuration: **https://cernbox.cern.ch/s/HufJpISlk5rX3EJ**
  - Command:
    - `$o2-analysistutorial-lf-resonances-step0 -b --aod-file AO2D_ResoGenMC_50files.root` Default
    - `$o2-analysistutorial-lf-resonances-step0 -b --aod-file AO2D_ResoGenMC_50files.root --nBins 10` Input from cli
    - `$o2-analysistutorial-lf-resonances-step0 -b --aod-file AO2D_ResoGenMC_50files.root --configuration json://configuration_resotutorial_0.json` Using configuration.json

Detail: Resonances_step0 #1

```cpp
struct resonances_tutorial {
  HistogramRegistry histos{"histos", {}, OutputObjHandlingPolicy::AnalysisObject};

  // Configurable for number of bins
  Configurable<int> nBins{"nBins", 100, "N bins in all histos"};
  // Configurable for min pT cut
  Configurable<double> cMinPtcut{"cMinPtcut", 0.15, "Track minium pt cut"};

  // Initialize the ananlysis task
  void init(o2::framework::InitContext&)
  {
    // register histograms
    histos.add("hVertexZ", "hVertexZ", HistType::kTH1F, {{nBins, -15., 15.}});
    histos.add("hEta", "Eta distribution", kTH1F, {{200, -1.0f, 1.0f}});
  }
```

Detail: Resonances_step0 #1

```cpp
struct resonances_tutorial {
  HistogramRegistry histos{"histos", {}, OutputObjHandlingPolicy::AnalysisObject};

  // Configurable for number of bins
  Configurable<int> nBins{"nBins", 100, "N bins in all histos"};
  // Configurable for min pT cut
  Configurable<double> cMinPtcut{"cMinPtcut", 0.15, "Track minium pt cut"};

  // Initialize the ananlysis task
  void init(o2::framework::InitContext&)
  {
    // register histograms
    histos.add("hVertexZ", "hVertexZ", HistType::kTH1F, {{nBins, -15., 15.}});
    histos.add("hEta", "Eta distribution", kTH1F, {{200, -1.0f, 1.0f}});
  }
```

Histogram output

Detail: Resonances_step0 #1

```cpp
struct resonances_tutorial {
  HistogramRegistry histos{"histos", {}, OutputObjHandlingPolicy::AnalysisObject};

  // Configurable for number of bins
  Configurable<int> nBins{"nBins", 100, "N bins in all histos"};
  // Configurable for min pT cut
  Configurable<double> cMinPtcut{"cMinPtcut", 0.15, "Track minium pt cut"};

  // Initialize the ananlysis task
  void init(o2::framework::InitContext&)
  {
    // register histograms
    histos.add("hVertexZ", "hVertexZ", HistType::kTH1F, {{nBins, -15., 15.}});
    histos.add("hEta", "Eta distribution", kTH1F, {{200, -1.0f, 1.0f}});
  }
```

Histogram output

Configurables

Detail: Resonances_step0 #1

```cpp
struct resonances_tutorial {
  HistogramRegistry histos{"histos", {}, OutputObjHandlingPolicy::AnalysisObject};

  // Configurable for number of bins
  Configurable<int> nBins{"nBins", 100, "N bins in all histos"};
  // Configurable for min pT cut
  Configurable<double> cMinPtcut{"cMinPtcut", 0.15, "Track minium pt cut"};

  // Initialize the ananlysis task
  void init(o2::framework::InitContext&)
  {
    // register histograms
    histos.add("hVertexZ", "hVertexZ", HistType::kTH1F, {{nBins, -15., 15.}});
    histos.add("hEta", "Eta distribution", kTH1F, {{200, -1.0f, 1.0f}});
  }
```

Histogram output

Configurables

Initializing variables

Detail: Resonances_step0 #2

```cpp
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;

  return true;
}

// Fill histograms (main function)
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1))
      continue; // track selection and PID selection
    histos.fill(HIST("hEta"), track1.eta());
  }
}

// Process the data
void process(aod::ResoCollision& collision, aod::ResoTracks const& resotracks)
{
  // Fill the event counter
  histos.fill(HIST("hVertexZ"), collision.posZ());
  fillHistograms<false, false>(collision, resotracks, resotracks); // Fill histograms, no MC, no mixing
}
```

Detail: Resonances_step0 #2

```cpp
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;

  return true;
}

// Fill histograms (main function)
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1))
      continue; // track selection and PID selection
    histos.fill(HIST("hEta"), track1.eta());
  }
}
```

```cpp
// Process the data
void process(aod::ResoCollision& collision, aod::ResoTracks const& resotracks)
{
  // Fill the event counter
  histos.fill(HIST("hVertexZ"), collision.posZ());
  fillHistograms<false, false>(collision, resotracks, resotracks); // Fill histograms, no MC, no mixing
}
```

Main process loop

Detail: Resonances_step0 #2

```cpp
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;

  return true;
}
```

```cpp
// Fill histograms (main function)
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1))
      continue; // track selection and PID selection
    histos.fill(HIST("hEta"), track1.eta());
  }
}
```

Template function for all kinds of process

```cpp
// Process the data
void process(aod::ResoCollision& collision, aod::ResoTracks const& resotracks)
{
  // Fill the event counter
  histos.fill(HIST("hVertexZ"), collision.posZ());
  fillHistograms<false, false>(collision, resotracks, resotracks); // Fill histograms, no MC, no mixing
}
```

Main process loop

Detail: Resonances_step0 #2

```cpp
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;

  return true;
}
```

Track selection

```cpp
// Fill histograms (main function)
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1))
      continue; // track selection and PID selection
    histos.fill(HIST("hEta"), track1.eta());
  }
}
```

Template function for all kinds of process

```cpp
// Process the data
void process(aod::ResoCollision& collision, aod::ResoTracks const& resotracks)
{
  // Fill the event counter
  histos.fill(HIST("hVertexZ"), collision.posZ());
  fillHistograms<false, false>(collision, resotracks, resotracks); // Fill histograms, no MC, no mixing
}
```

Main process loop

Detail: Resonances_step0 #2

```cpp
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;

  return true;
}
```

Track selection

```cpp
// Fill histograms (main function)
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1))
      continue; // track selection and PID selection
    histos.fill(HIST("hEta"), track1.eta());
  }
}
```

Template function for all kinds of process

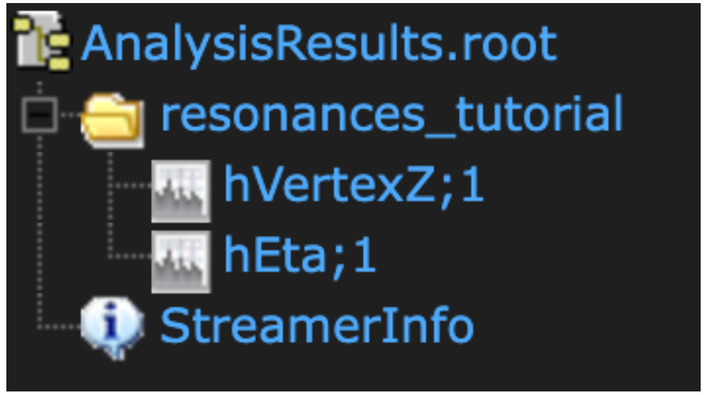
```cpp
// Process the data
void process(aod::ResoCollision& collision, aod::ResoTracks const& resotracks)
{
  // Fill the event counter
  histos.fill(HIST("hVertexZ"), collision.posZ());
  fillHistograms<false, false>(collision, resotracks, resotracks); // Fill histograms, no MC, no mixing
}
```

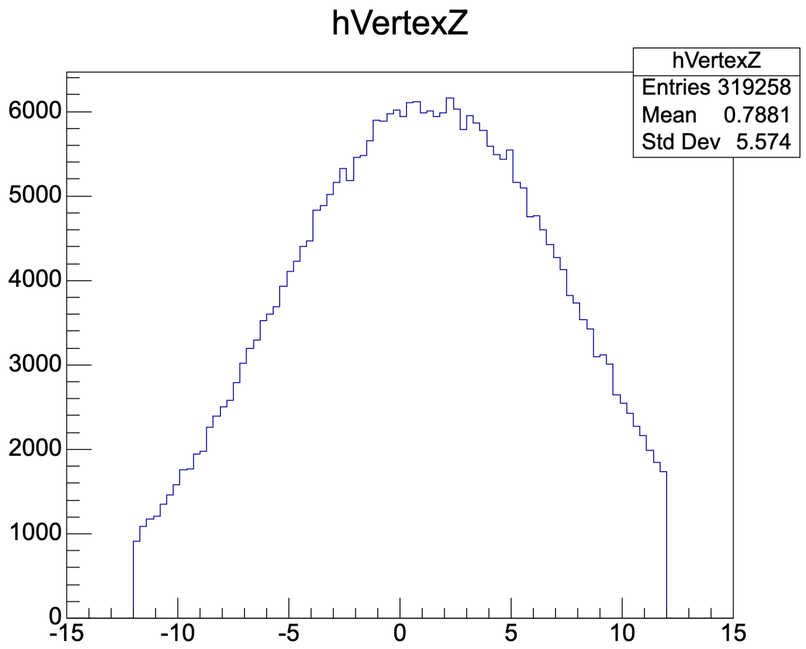
Resonance table!

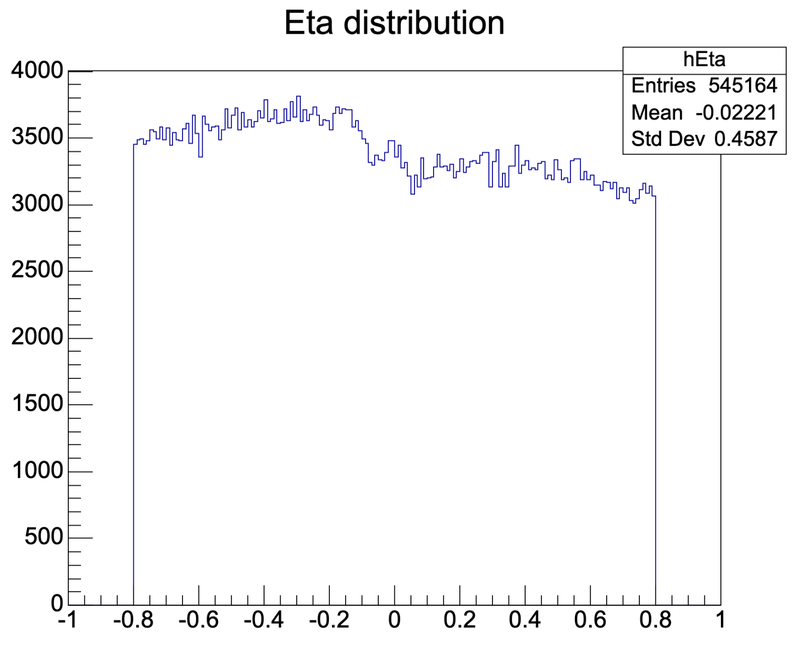
Main process loop

Hands-on 3: Using the derived data

Output







- **There will be two histograms.**
  - You can adjust the number of bins by using the configurable `nBins`

Hands-on 4: Draw Phi invariant mass histogram

- **Using the resonance derived output**
  - Download from here: **https://cernbox.cern.ch/s/8o0oJpMUbjoC9kM**
- **Modify the first task:** `O2Physics/Tutorials/PWGLF/Resonance/resonances_step0.cxx`
  - **Example:** `O2Physics/Tutorials/PWGLF/Resonance/resonances_step1.cxx`
    - bash script & configuration: **https://cernbox.cern.ch/s/HTM1See7Ua3ses1**
- **Strategy**
  - Add Track Selection
  - Add the PID Selection
  - Add the second track loop
  - Reconstruct the resonance

Detail: Resonances_step1 #1

Add track selection

```
// Track selection
// primary track condition
Configurable<bool> cfgPrimaryTrack{"cfgPrimaryTrack", true, "Primary track selection"};
Configurable<bool> cfgGlobalWoDCATrack{"cfgGlobalWoDCATrack", true, "Global track selection without DCA"};
Configurable<bool> cfgPVContributor{"cfgPVContributor", true, "PV contributor track selection"};
// DCA Selections
// DCAr to PV
Configurable<double> cMaxDCArToPVcut{"cMaxDCArToPVcut", 0.5, "Track DCAr cut to PV Maximum"};
// DCAz to PV
Configurable<double> cMaxDCAzToPVcut{"cMaxDCAzToPVcut", 2.0, "Track DCAz cut to PV Maximum"};
Configurable<double> cMinDCAzToPVcut{"cMinDCAzToPVcut", 0.0, "Track DCAz cut to PV Minimum"};
```

Track-selection filter reference: O2Physics/Common/DataModel/TrackSelectionTables.h

Detail: Resonances_step1 #1

Add track selection

```
// Track selection
// primary track condition
Configurable<bool> cfgPrimaryTrack{"cfgPrimaryTrack", true, "Primary track selection"};
Configurable<bool> cfgGlobalWoDCATrack{"cfgGlobalWoDCATrack", true, "Global track selection without DCA"};
Configurable<bool> cfgPVContributor{"cfgPVContributor", true, "PV contributor track selection"};
// DCA Selections
// DCAr to PV
Configurable<double> cMaxDCArToPVcut{"cMaxDCArToPVcut", 0.5, "Track DCAr cut to PV Maximum"};
// DCAz to PV
Configurable<double> cMaxDCAzToPVcut{"cMaxDCAzToPVcut", 2.0, "Track DCAz cut to PV Maximum"};
Configurable<double> cMinDCAzToPVcut{"cMinDCAzToPVcut", 0.0, "Track DCAz cut to PV Minimum"};
```

Track-selection helper

Track-selection filter reference: O2Physics/Common/DataModel/TrackSelectionTables.h

Detail: Resonances_step1 #1

Add track selection

```
// Track selection
// primary track condition
Configurable<bool> cfgPrimaryTrack{"cfgPrimaryTrack", true, "Primary track selection"};
Configurable<bool> cfgGlobalWoDCATrack{"cfgGlobalWoDCATrack", true, "Global track selection without DCA"};
Configurable<bool> cfgPVContributor{"cfgPVContributor", true, "PV contributor track selection"};
// DCA Selections
// DCAr to PV
Configurable<double> cMaxDCArToPVcut{"cMaxDCArToPVcut", 0.5, "Track DCAr cut to PV Maximum"};
// DCAz to PV
Configurable<double> cMaxDCAzToPVcut{"cMaxDCAzToPVcut", 2.0, "Track DCAz cut to PV Maximum"};
Configurable<double> cMinDCAzToPVcut{"cMinDCAzToPVcut", 0.0, "Track DCAz cut to PV Minimum"};
```

Track-selection helper

DCA Selections

Track-selection filter reference: O2Physics/Common/DataModel/TrackSelectionTables.h

Detail: Resonances_step1 #2

Add track selection

```
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;
  if (std::abs(track.dcaXY()) > cMaxDCArToPVcut)
    return false;
  if (std::abs(track.dcaZ()) > cMaxDCAzToPVcut)
    return false;
  if (cfgPrimaryTrack && !track.isPrimaryTrack())
    return false;
  if (cfgGlobalWoDCATrack && !track.isGlobalTrackWoDCA())
    return false;
  if (cfgPVContributor && !track.isPVContributor())
    return false;
  return true;
}
```

Detail: Resonances_step1 #2

Add track selection

```
// Track selection
template <typename TrackType>
bool trackCut(const TrackType track)
{
  // basic track cuts
  if (std::abs(track.pt()) < cMinPtcut)
    return false;
  if (std::abs(track.dcaXY()) > cMaxDCArToPVcut)
    return false;
  if (std::abs(track.dcaZ()) > cMaxDCAzToPVcut)
    return false;
  if (cfgPrimaryTrack && !track.isPrimaryTrack())
    return false;
  if (cfgGlobalWoDCATrack && !track.isGlobalTrackWoDCA())
    return false;
  if (cfgPVContributor && !track.isPVContributor())
    return false;
  return true;
}
```

Additional track-selections

Detail: Resonances_step1 #3

Add PID selection

```
// PID selection
Configurable<float> nsigmaCutTPC{"nsigmacutTPC", 3.0, "Value of the TPC Nsigma cut"};
Configurable<float> nsigmaCutCombined{"nsigmaCutCombined", 3.0, "Value of the TOF Nsigma cut"};
```

```
// PID selection TPC +TOF Veto
template <typename T>
bool selectionPID(const T& candidate)
{
  if (candidate.hasTOF() && (candidate.tofNSigmaKa() * candidate.tofNSigmaKa() + candidate.tpcNSigmaKa()
  * candidate.tpcNSigmaKa()) < (2.0 * nsigmaCutCombined * nsigmaCutCombined)) {
    return true;
  } else if (std::abs(candidate.tpcNSigmaKa()) < nsigmaCutTPC) {
    return true;
  }
  return false;
}
```

Dedicated function for PID selection

Detail: Resonances_step1 #4

Add second track loop

```
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  auto multiplicity = collision.multV0M();
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1) || !selectionPID(track1)) {
      continue; // track selection and PID selection
    }
    // QA plots
    histos.fill(HIST("hEta"), track1.eta());
    histos.fill(HIST("hDcaxy"), track1.dcaXY());
    histos.fill(HIST("hDcaz"), track1.dcaZ());
    histos.fill(HIST("hNsigmaKaonTPC"), track1.tpcNSigmaKa());
    if (track1.hasTOF()) {
      histos.fill(HIST("hNsigmaKaonTOF"), track1.tofNSigmaKa());
    }
    for (auto track2 : dTracks2) { // loop over all dTracks2
      if (!trackCut(track2) || !selectionPID(track2)) {
        continue; // track selection and PID selection
      }
      if (track2.index() <= track1.index()) {
        continue; // condition to avoid double counting of pair
      }
```

Detail: Resonances_step1 #4

Add second track loop

```
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  auto multiplicity = collision.multV0M();
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1) || !selectionPID(track1)) {
      continue; // track selection and PID selection
    }
    // QA plots
    histos.fill(HIST("hEta"), track1.eta());
    histos.fill(HIST("hDcaxy"), track1.dcaXY());
    histos.fill(HIST("hDcaz"), track1.dcaZ());
    histos.fill(HIST("hNsigmaKaonTPC"), track1.tpcNSigmaKa());
    if (track1.hasTOF()) {
      histos.fill(HIST("hNsigmaKaonTOF"), track1.tofNSigmaKa());
    }
    for (auto track2 : dTracks2) { // loop over all dTracks2
      if (!trackCut(track2) || !selectionPID(track2)) {
        continue; // track selection and PID selection
      }
      if (track2.index() <= track1.index()) {
        continue; // condition to avoid double counting of pair
      }
```

Second track loop

Detail: Resonances_step1 #4

Add second track loop

```
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  auto multiplicity = collision.multV0M();
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1) || !selectionPID(track1)) {
      continue; // track selection and PID selection
    }
    // QA plots
    histos.fill(HIST("hEta"), track1.eta());
    histos.fill(HIST("hDcaxy"), track1.dcaXY());
    histos.fill(HIST("hDcaz"), track1.dcaZ());
    histos.fill(HIST("hNsigmaKaonTPC"), track1.tpcNSigmaKa());
    if (track1.hasTOF()) {
      histos.fill(HIST("hNsigmaKaonTOF"), track1.tofNSigmaKa());
    }
    for (auto track2 : dTracks2) { // loop over all dTracks2
      if (!trackCut(track2) || !selectionPID(track2)) {
        continue; // track selection and PID selection
      }
      if (track2.index() <= track1.index()) {
        continue; // condition to avoid double counting of pair
      }
```

Track selection

Second track loop

Detail: Resonances_step1 #4

Add second track loop

```
template <bool IsMC, bool IsMix, typename CollisionType, typename TracksType>
void fillHistograms(const CollisionType& collision, const TracksType& dTracks1, const TracksType& dTracks2)
{
  auto multiplicity = collision.multV0M();
  for (auto track1 : dTracks1) { // loop over all dTracks1
    if (!trackCut(track1) || !selectionPID(track1)) {
      continue; // track selection and PID selection
    }
    // QA plots
    histos.fill(HIST("hEta"), track1.eta());
    histos.fill(HIST("hDcaxy"), track1.dcaXY());
    histos.fill(HIST("hDcaz"), track1.dcaZ());
    histos.fill(HIST("hNsigmaKaonTPC"), track1.tpcNSigmaKa());
    if (track1.hasTOF()) {
      histos.fill(HIST("hNsigmaKaonTOF"), track1.tofNSigmaKa());
    }
    for (auto track2 : dTracks2) { // loop over all dTracks2
      if (!trackCut(track2) || !selectionPID(track2)) {
        continue; // track selection and PID selection
      }
      if (track2.index() <= track1.index()) {
        continue; // condition to avoid double counting of pair
      }
```

Track selection

Second track loop

Prevent double counting

Detail: Resonances_step1 #5

Reconstruct the resonance

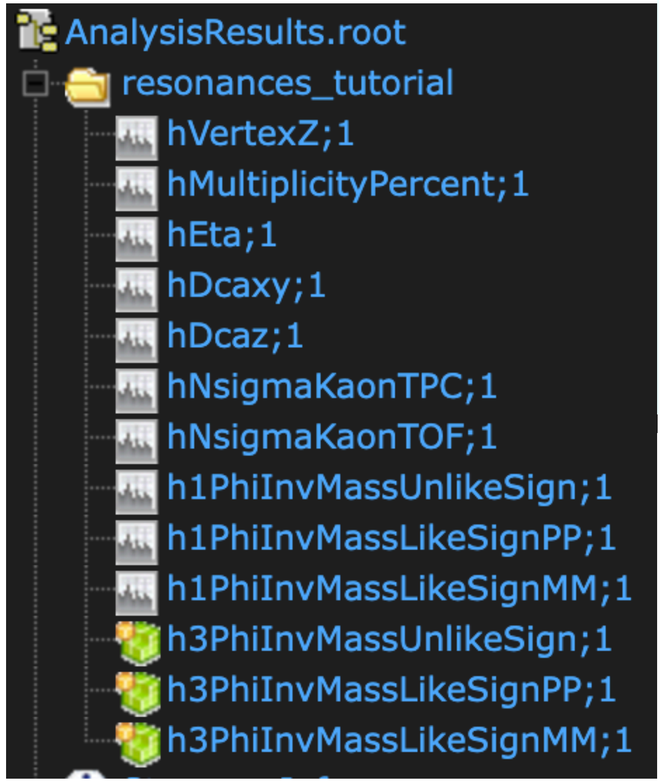
```
// Fill histograms (main function)
double rapidity, mass, pT, paircharge;
TLorentzVector daughter1, daughter2, mother;
```

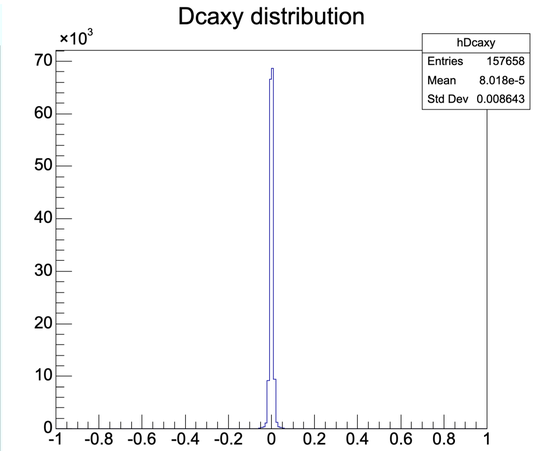
Common variables (TLorentzVector)

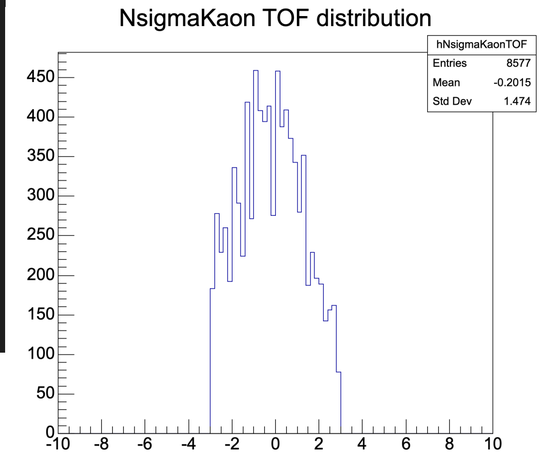
```
daughter1.SetXYZM(track1.px(), track1.py(), track1.pz(), massKa); // set the daughter1 4-momentum
daughter2.SetXYZM(track2.px(), track2.py(), track2.pz(), massKa); // set the daughter2 4-momentum
mother = daughter1 + daughter2;                                   // calculate the mother 4-momentum
mass = mother.M();
pT = mother.Pt();
rapidity = mother.Rapidity();
paircharge = track1.sign() * track2.sign();

if (std::abs(rapidity) > 0.5)
  continue; // rapidity cut

if (paircharge < 0) { // unlike sign
  histos.fill(HIST("h3PhiInvMassUnlikeSign"), multiplicity, pT, mass);
  histos.fill(HIST("h1PhiInvMassUnlikeSign"), mass);
```

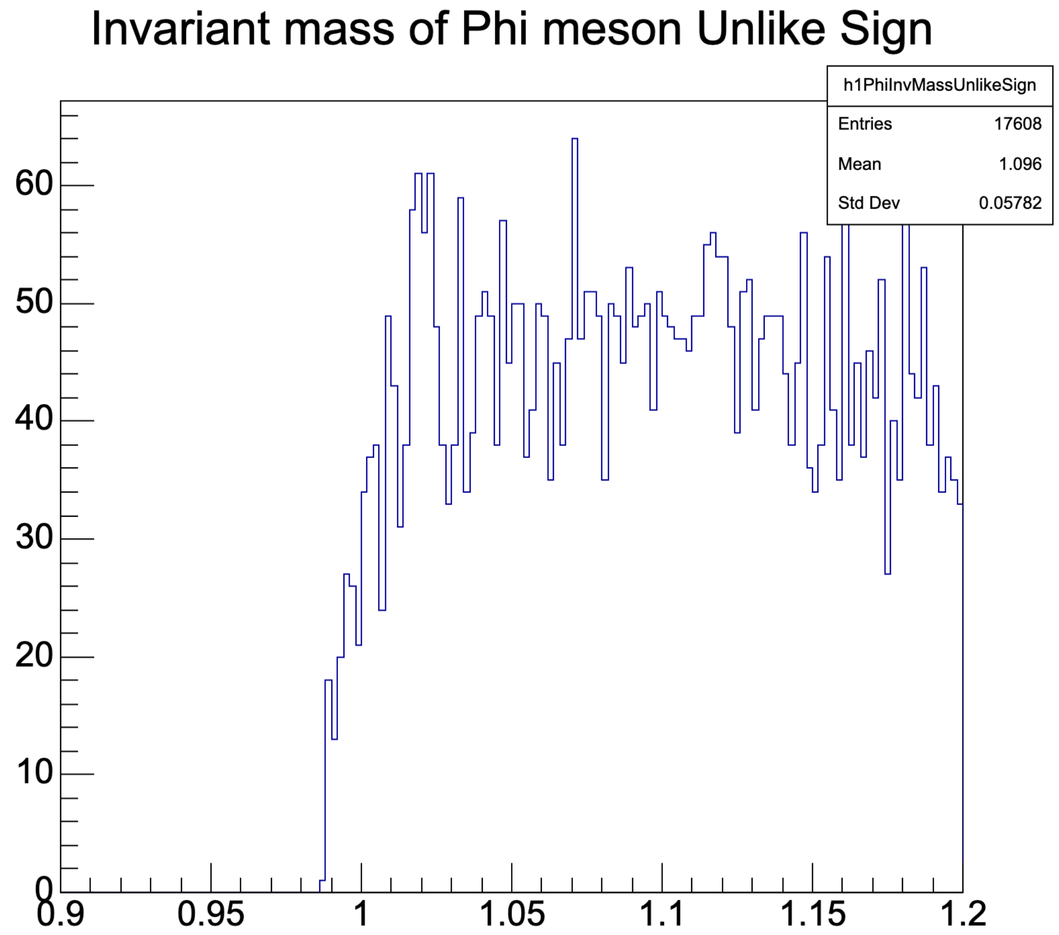
Hands-on 4: Draw Phi invariant mass histogram

Output









Hands-on 5: Event Mixing

- **Using the resonance derived output**
  - Download from here: **https://cernbox.cern.ch/s/8o0oJpMUbjoC9kM**
- **Modify the previous task:** `O2Physics/Tutorials/PWGLF/Resonance/resonances_step1.cxx`
  - **Example:** `O2Physics/Tutorials/PWGLF/Resonance/resonances_step2.cxx`
    - bash script & configuration: **https://cernbox.cern.ch/s/HTM1See7Ua3ses1**
- **Strategy**
  - Add new process for Event mixing
  - Define a new binning type and pairs
  - Fill the histogram

Detail: Resonances_step2

Simplest event-mixing

```
// Event Mixing
Configurable<int> nEvtMixing{"nEvtMixing", 5, "Number of events to mix"};
ConfigurableAxis CfgVtxBins{"CfgVtxBins", {VARIABLE_WIDTH, -10.0f, -8.f, -6.f, -4.f,
-2.f, 0.f, 2.f, 4.f, 6.f, 8.f, 10.f}, "Mixing bins - z-vertex"};
ConfigurableAxis CfgMultBins{"CfgMultBins", {VARIABLE_WIDTH, 0., 1., 5., 10., 30.,
50., 70., 100., 110.}, "Mixing bins - multiplicity"};
```

Configurable for Event-Mixing (2D pool)

```
// Processing Event Mixing
using BinningTypeVtxZT0M = ColumnBinningPolicy<aod::collision::PosZ, aod::resocollision::MultV0M>;
void processME(o2::aod::ResoCollisions& collisions, aod::ResoTracks const& resotracks)
{
  auto tracksTuple = std::make_tuple(resotracks);
  BinningTypeVtxZT0M colBinning{{CfgVtxBins, CfgMultBins}, true};
  SameKindPair<aod::ResoCollisions, aod::ResoTracks, BinningTypeVtxZT0M> pairs{colBinning, nEvtMixing, -1, collisions,
  tracksTuple, &cache}; // -1 is the number of the bin to skip

  for (auto& [collision1, tracks1, collision2, tracks2] : pairs) { // loop over all pairs
    fillHistograms<false, true>(collision1, tracks1, tracks2);     // Fill histograms, no MC, mixing
  }
};
PROCESS_SWITCH(resonances_tutorial, processME, "Process EventMixing for combinatorial background", false); // Event Mixing
```

Make a new process

Detail: Resonances_step2

Simplest event-mixing

```
// Event Mixing
Configurable<int> nEvtMixing{"nEvtMixing", 5, "Number of events to mix"};
ConfigurableAxis CfgVtxBins{"CfgVtxBins", {VARIABLE_WIDTH, -10.0f, -8.f, -6.f, -4.f,
-2.f, 0.f, 2.f, 4.f, 6.f, 8.f, 10.f}, "Mixing bins - z-vertex"};
ConfigurableAxis CfgMultBins{"CfgMultBins", {VARIABLE_WIDTH, 0., 1., 5., 10., 30.,
50., 70., 100., 110.}, "Mixing bins - multiplicity"};
```

Configurable for Event-Mixing (2D pool)

```
// Processing Event Mixing
using BinningTypeVtxZT0M = ColumnBinningPolicy<aod::collision::PosZ, aod::resocollision::MultV0M>;
void processME(o2::aod::ResoCollisions& collisions, aod::ResoTracks const& resotracks)
{
  auto tracksTuple = std::make_tuple(resotracks);
  BinningTypeVtxZT0M colBinning{{CfgVtxBins, CfgMultBins}, true};
  SameKindPair<aod::ResoCollisions, aod::ResoTracks, BinningTypeVtxZT0M> pairs{colBinning, nEvtMixing, -1, collisions,
  tracksTuple, &cache}; // -1 is the number of the bin to skip

  for (auto& [collision1, tracks1, collision2, tracks2] : pairs) { // loop over all pairs
    fillHistograms<false, true>(collision1, tracks1, tracks2);     // Fill histograms, no MC, mixing
  }
};
PROCESS_SWITCH(resonances_tutorial, processME, "Process EventMixing for combinatorial background", false); // Event Mixing
```

Make a new process

Let the function know