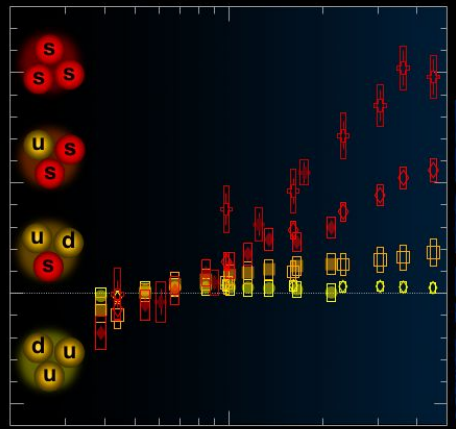
Weak decays and secondary vertices

O2 Analysis Tutorial 11/11/2025



Romain Schotter

Austrian Academy of Sciences, Austria

romain.schotter@cern.ch

Weak decay topologies: V0s and cascades

The strange hadrons reconstruction relies on two weak decay topologies:

V ^{0} : neutral particle decaying into a pair of charged particles (V-shaped decay)

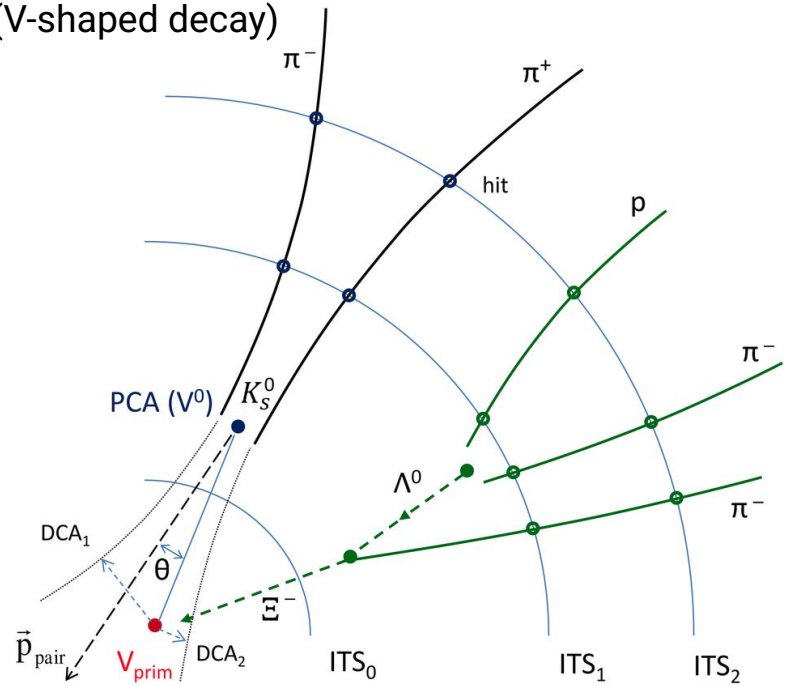
$K^0_S \rightarrow \pi^+\pi^-$ c τ = 2.6844 cm B.R. 69.2%

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

Cascade : charged particle decaying into a V ^{0} + charged particle (bachelor)

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

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



The strangeness analysis workflow

Asynchronous reconstruction (O^2)

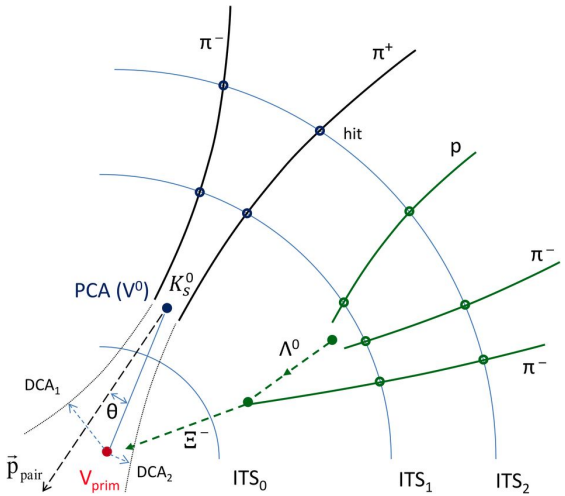
Secondary vertex reconstruction: SVertexer.cxx
Initial loose cut parameters: SVertexerParams.h

The strangeness analysis workflow

Asynchronous reconstruction (O^2)

Secondary vertex reconstruction: SVertexer.cxx
Initial loose cut parameters: SVertexerParams.h

- Core of the secondary vertex reconstruction performed by the SVertexer.cxx is the DCAfitter
- DCAfitter = central tool to determine the Distance of Closest Approach (DCA) between tracks
- Some examples:
  - strangenessBuilderHelper: V0s (K^0_S, Λ, γ), cascades (Ξ, Ω)
  - hyperRecoTask: ${}^3_{\Lambda}H$
  - candidateCreator2Prong: D^0
  - candidateCreatorB0: B^0
  - tableReader: J/ψ, ψ(2S) at midrapidity and forward rapidity



The strangeness analysis workflow

Asynchronous reconstruction (O^2)

AO2D.root

Secondary vertex reconstruction: SVertexer.cxx
Initial loose cut parameters: SVertexerParams.h

V0 table: aod::V0s
cascade table: aod::Cascades

The strangeness analysis workflow

Asynchronous reconstruction (O^2)

AO2D.root

Secondary vertex reconstruction: SVertexer.cxx
Initial loose cut parameters: SVertexerParams.h

V0 table: aod::V0s
cascade table: aod::Cascades

Run 3 V0 table (version 002)
Header file: Framework/Core/include/Framework/AnalysisDataModel.h
Is used in:

- o2::aod::V0s = o2::aod::V0s_002

Name		Getter	Type	Comment
o2::soa::Index	GI	globalIndex	int64_t	
o2::aod::v0::CollisionId	I	collisionId	int32	Collision index
o2::aod::v0::PosTrackId	I	posTrackId	int	Positive track
o2::aod::v0::NegTrackId	I	negTrackId	int	Negative track
o2::aod::v0::V0Type		v0Type	uint8_t	custom bitmap for various selections (see below)
o2::aod::v0::IsStandardV0	D	isStandardV0	bool	is standard V0
o2::aod::v0::IsPhotonV0	D	isPhotonV0	bool	is TPC-only V0 for which the photon-mass-hypothesis was good
o2::aod::v0::IsCollinearV0	D	isCollinearV0	bool	is V0 for which the photon-mass-hypothesis was good and was fitted collinearly

Run 3 cascade table
Header file: Framework/Core/include/Framework/AnalysisDataModel.h
Is used in:

- o2::aod::Cascades = o2::aod::Cascades_001

Name		Getter	Type	Comment
o2::soa::Index	GI	globalIndex	int64_t	
o2::aod::cascade::CollisionId	I	collisionId	int32	Collision index
o2::aod::cascade::V0Id	I	v0Id	int32	V0 index
o2::aod::cascade::BachelorId	I	bachelorId	int	Bachelor track index

The strangeness analysis workflow

Asynchronous reconstruction (O^2)

AO2D.root

Secondary vertex reconstruction: SVertexer.cxx
Initial loose cut parameters: SVertexerParams.h

V0 table: aod::V0s
cascade table: aod::Cascades

ONLY IDs are stored at these tables
NO topological/kinematic variables

Run 3 V0 table (version 002)
Header file: Framework/Core/include/Framework/AnalysisDataModel.h
Is used in:

- o2::aod::V0s = o2::aod::V0s_002

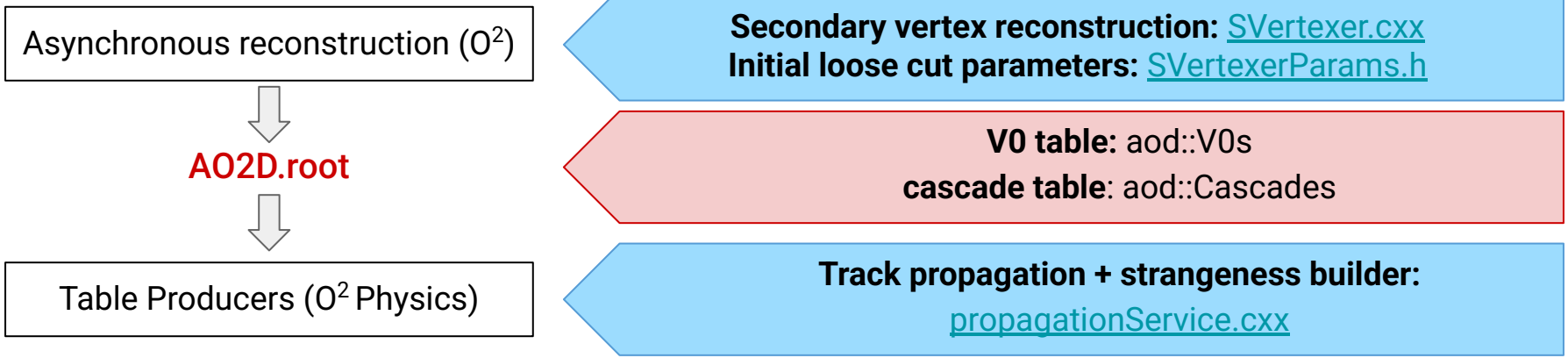
Name		Getter	Type	Comment
o2::soa::Index	GI	globalIndex	int64_t	
o2::aod::v0::CollisionId	I	collisionId	int32	Collision index
o2::aod::v0::PosTrackId	I	posTrackId	int	Positive track
o2::aod::v0::NegTrackId	I	negTrackId	int	Negative track
o2::aod::v0::V0Type		v0Type	uint8_t	custom bitmap for various selections (see below)
o2::aod::v0::IsStandardV0	D	isStandardV0	bool	is standard V0
o2::aod::v0::IsPhotonV0	D	isPhotonV0	bool	is TPC-only V0 for which the photon-mass-hypothesis was good
o2::aod::v0::IsCollinearV0	D	isCollinearV0	bool	is V0 for which the photon-mass-hypothesis was good and was fitted collinearly

Run 3 cascade table
Header file: Framework/Core/include/Framework/AnalysisDataModel.h
Is used in:

- o2::aod::Cascades = o2::aod::Cascades_001

Name		Getter	Type	Comment
o2::soa::Index	GI	globalIndex	int64_t	
o2::aod::cascade::CollisionId	I	collisionId	int32	Collision index
o2::aod::cascade::V0Id	I	v0Id	int32	V0 index
o2::aod::cascade::BachelorId	I	bachelorId	int	Bachelor track index

The strangeness analysis workflow



The strangeness analysis workflow

- V0/cascade tables produced in the `propagationService.cxx` using the `strangnessBuilderHelper.h`

- **Main task responsible for**
  - propagating the tracks to the DCA to the PV
  - assembling V0/cascade candidates and its analysis-related properties

- Input configuration: **topological selections**
  + optional pre-selections (select V0s/casc. based on invariant mass and TPC Nσ selections)

- Output:
  - **V0Datas**: regular V0 analysis table
  - **CascDatas**: regular cascade analysis table
  - **TraCascDatas**: tracked cascade analysis table
  - **KFCascDatas**: KF cascade analysis table

Analysis Level Tables: aod::V0Datas

#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
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
	...	...
	v0data::Px	px
	v0data::Py	py
	v0data::Pz	pz

Analysis Level Tables: aod::V0Datas

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

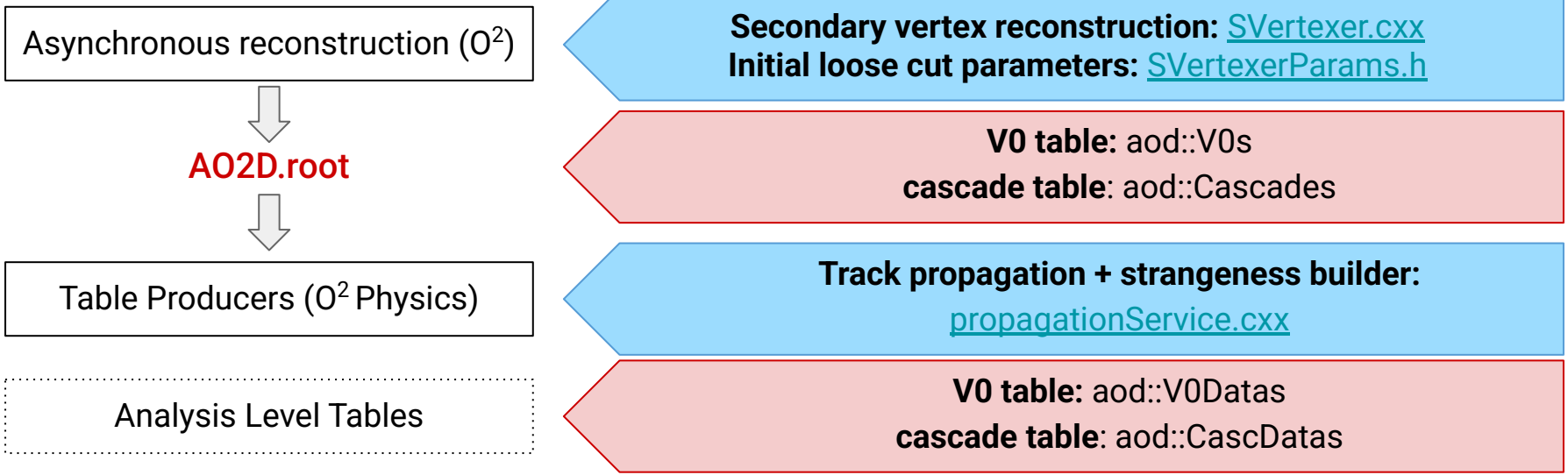
Derived (dynamic)	
v0data::Pt	pt
v0data::V0Radius	v0radius
v0data::V0CosPA	v0cospa
v0data::DCAV0ToPV	dcav0topv
v0data::MLambda	mLambda
v0data::MAntiLambda	mAntiLambda
v0data::MK0Short	mK0Short
v0data::YK0Short	yK0Short
v0data::YLambda	yLambda
v0data::Eta	eta
v0data::Phi	phi
...	...
v0data::Px	px
v0data::Py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

Analysis Level Tables: aod::CascDatas

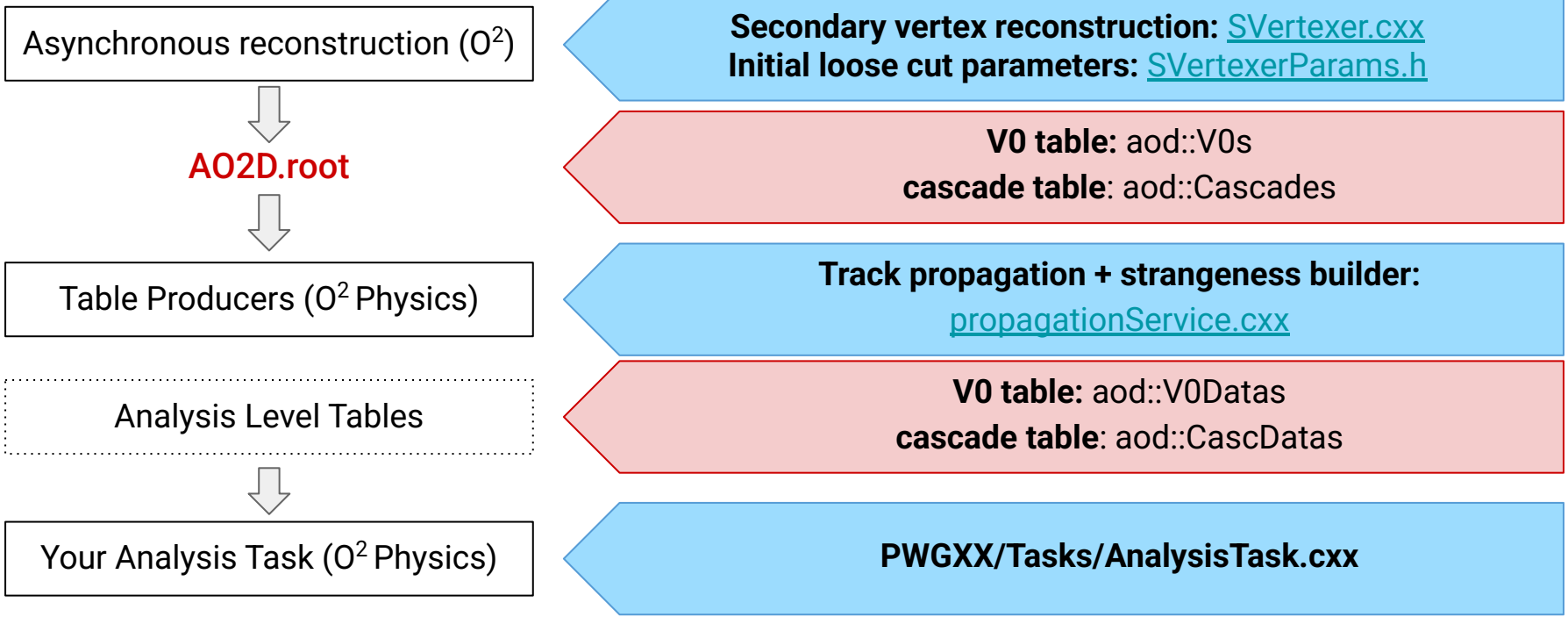
#include "PWGLF/DataModel/LFStrangenessTables.h" CascDatas = soa::Join<**CascIndices**, CascBBs, **CascCores**>

	Name	Getter
Bound	cascade::V0Id	v0Id
	cascade::BachelorId	bachelorId
	cascade::CollisionId	collisionId
Calculated	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
Decay vertex position	cascdata::X	x
	cascdata::Y	y
	cascdata::Z	z
Topological variables	cascdata::DCAV0Daughters	dcav0daughters
	cascdata::DCACascDaughters	dcascascdaughters
	cascdata::DCAPosToPV	dcapostopv
Derived (dynamic)	cascdata::DCANegToPV	dcanegtopv
	cascdata::DCABachToPV	dcabachtopv
	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::Pt	pt
	cascdata::P	p
	cascdata::PxLambda	pxlambda
	...	...

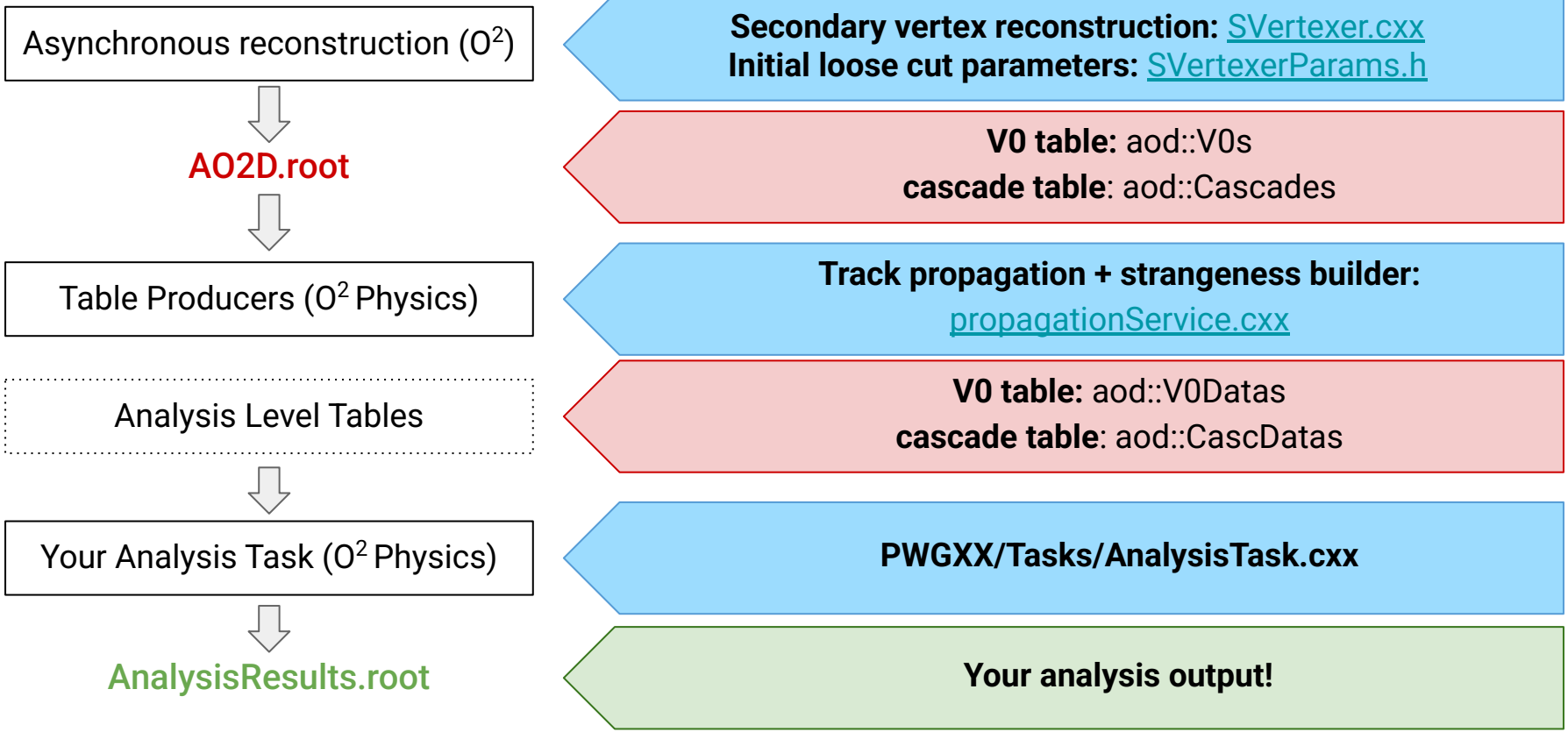
The strangeness analysis workflow



The strangeness analysis workflow



The strangeness analysis workflow



How to access the strangeness tables?

Include strangeness header to access to the strangeness data model

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

Header needed
`LFStrangenessTables.h`

How to access the strangeness tables?

Include strangeness header to access to the strangeness data model

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

Header needed
`LFStrangenessTables.h`

Subscribe to the strangeness tables

```
void process(aod::V0Datas const& v0s)
{...}
```

Contains kinematic and topological information about the **V0s**

```
void process(aod::CascDatas const& cascades)
{...}
```

Contains kinematic and topological information about the **cascades**

How to access the strangeness tables?

Include strangeness header to access to the strangeness data model

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

Header needed
`LFStrangenessTables.h`

Subscribe to the strangeness tables

```
void process(aod::V0Datas const& v0s)
{
  for (const auto& v0 : v0s) {
    v0.mK0Short();
    v0.mLambda();
    v0.mAntiLambda();
  }
}
```

```
void process(aod::CascDatas const& cascades)
{
  for (const auto& cascade : cascades) {
    cascade.mXi();
    cascade.mOmega();
  }
}
```

Loop over all V0s/cascades

Access to the table content

Access to the invariant mass calculated under different mass hypothesis

How to access track properties?

- V0/cascade tables reference the track table allowing access information about daughter tracks → if you try to check the PID of daughter tracks with reference to that table the code will fail!

```
float nsigma_bach = cascade.bachelor.tpcNSigmaPi();
float nsigma_pos = cascade.posTrack.tpcNSigmaPr();
float nsigma_neg = cascade.negTrack.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, aod::pidTPCPr>

void process(aod::CascDatas const& cascades,
             DaughterTracks const&)
 {
   for (const auto& cascade : cascades) {
     float nsigma_bach = cascade.bachelor_as<DaughterTracks>().tpcNSigmaPi();
     float nsigma_pos = cascade.posTrack_as<DaughterTracks>().tpcNSigmaPr();
     float nsigma_neg = cascade.negTrack_as<DaughterTracks>().tpcNSigmaPi();
   }
 }
```

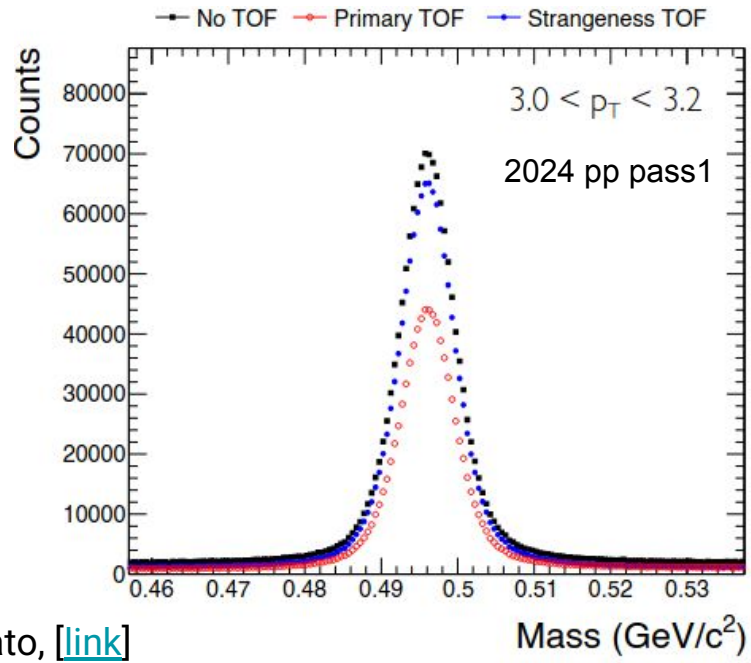
Works!

How to access TOF information?

- Primary TOF PID information can be accessed in the same way as the TPC PID

- There also exists strangeness-based TOF PID: strangenesstofpid.cxx, **optimized for secondary particles**
  - **Collision association** may be incorrect, biasing the event time (t_0)
    → recalculated with V0/cascade collision t_0
  - **Travel time before decay** not accounted for
    → correct for travel before decay

- In high interaction rate environments,
  → primary TOF leads large to signal loss
  → while strangeness-based TOF preserves more than 95% of signal



D. Chinellato, [link]

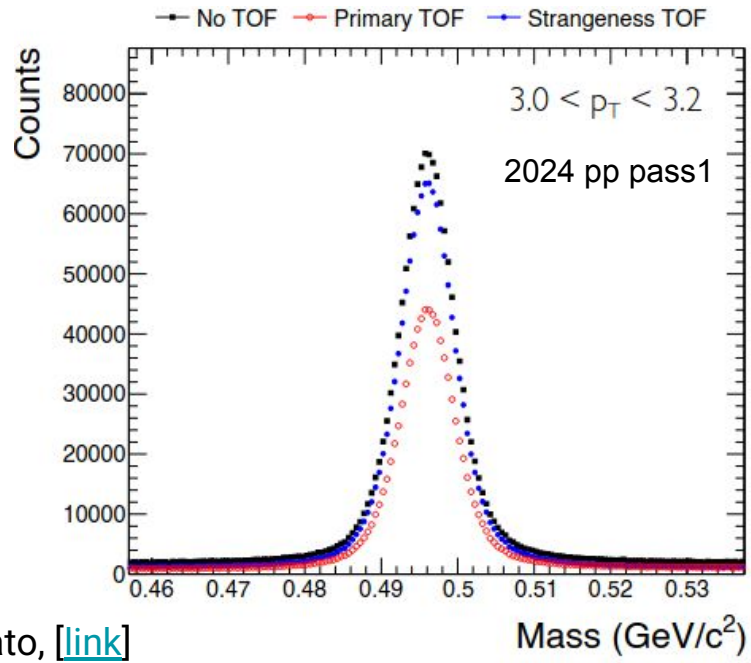
How to access TOF information?

- Primary TOF PID information can be accessed in the same way as the TPC PID

- There also exists strangeness-based TOF PID: strangenesstofpid.cxx, **optimized for secondary particles**
  - **Collision association** may be incorrect, biasing the event time (t_0)
    → recalculated with V0/cascade collision t_0
  - **Travel time before decay** not accounted for
    → correct for travel before decay

- In high interaction rate environments,
  → primary TOF leads large to signal loss
  → while strangeness-based TOF preserves more than 95% of signal

- How to use it?



D. Chinellato, [link]

How to access TOF information?

Include strangeness PID header to access to the strangeness data model

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

Header needed
`LFStrangenessPIDTables.h`

Subscribe to the strangeness PID tables

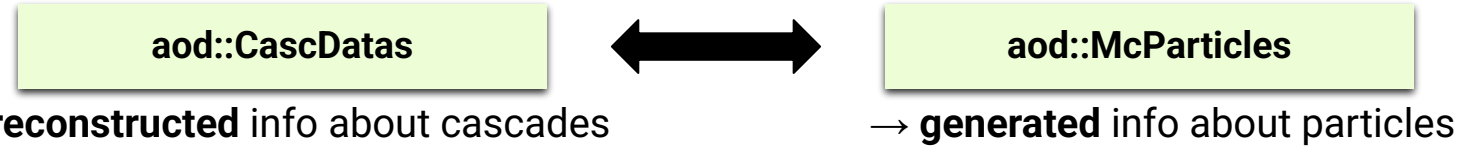
```
void process(soa::Join<aod::CascDatas,aod::CascTOFNSigmas,aod::CascTOFPIDs> const& cascades)
 {
   for (const auto& cascade : cascades) {
     if (cascade.bachelorHasTOF() && std::abs(cascade.tofNSigmaXiPi()) > NSigmaTOF) {
       continue;
     }
     if (cascade.positiveHasTOF() && std::abs(cascade.tofNSigmaXiLaPr()) > NSigmaTOF) {
       continue;
     }
     if (cascade.negativeHasTOF() && std::abs(cascade.tofNSigmaXiLaPi()) > NSigmaTOF) {
       continue;
     }
   }
 }
```

Subscribe to **aod::CascTOFNSigmas** and **aod::CascTOFPIDs** tables

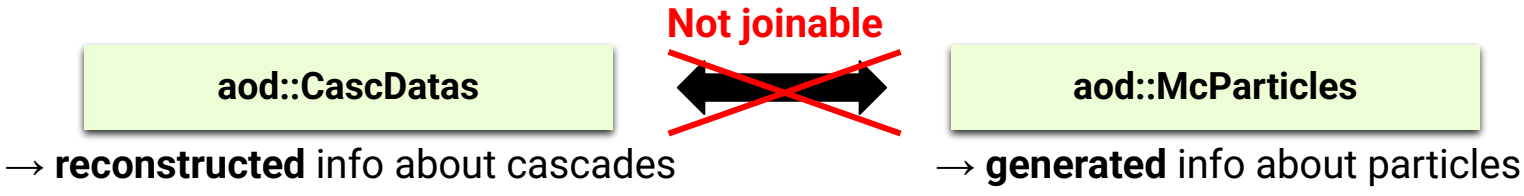
```
if (!cascade.tofXiCompatibility(NSigmaTOF)) {
  continue;
}
```

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

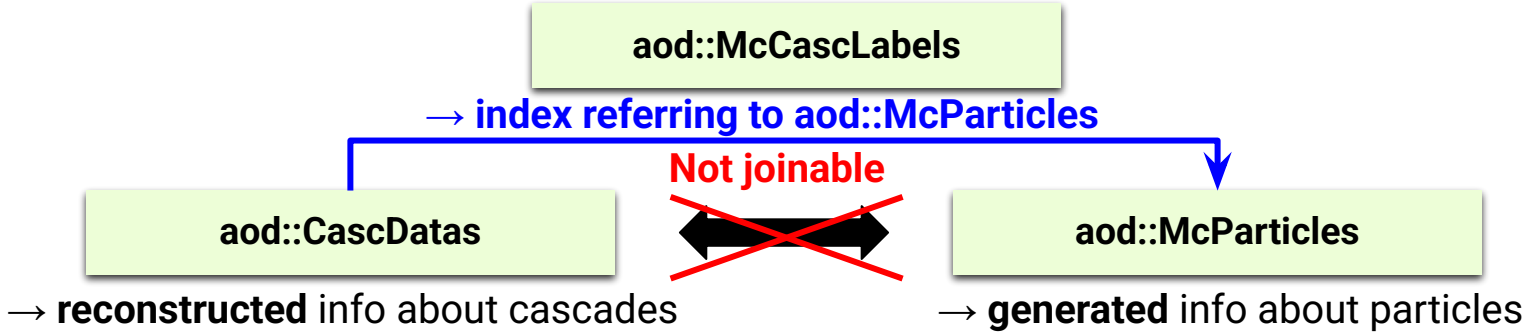
How to access MC information?



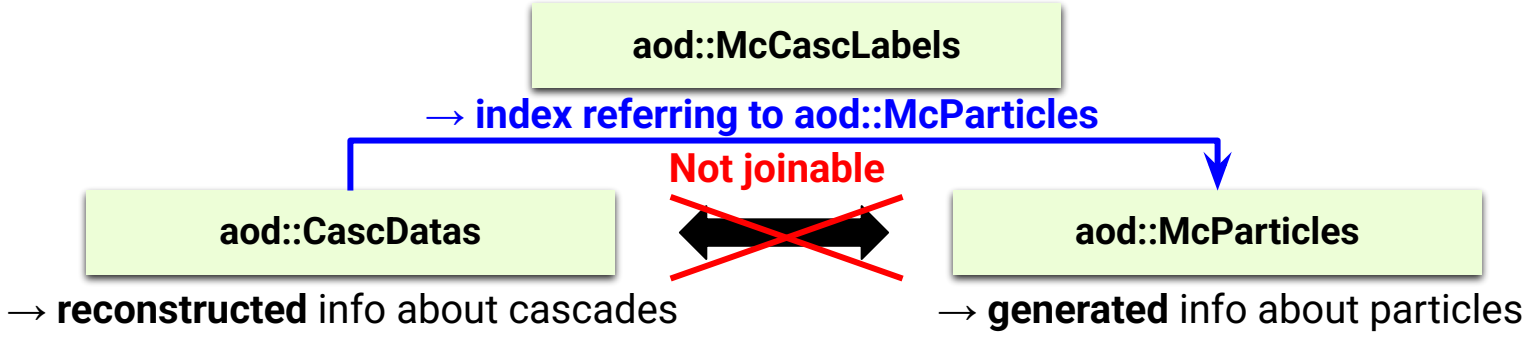
How to access MC information?



How to access MC information?



How to access MC information?




```
void process(soa::Join<aod::CascDatas, aod::McCascLabels> const& cascades,
             aod::McParticles const&)
{
  for (const auto& cascade : cascades) {
    if (!cascade.has_mcParticle()) {
      continue;
    }
    auto cascmcParticle = cascade.mcParticle();
    cascmcParticle.pdgCode();
  }
}
```

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

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

How to setup the strangeness workflow?

The strangeness analysis workflow:

```
o2-analysis-ft0-corrected-table -b --configuration json://configuration.json |
o2-analysis-trackselection -b --configuration json://configuration.json |
o2-analysis-pid-tpc-service -b --configuration json://configuration.json |
o2-analysis-pid-tof-merge -b --configuration json://configuration.json |
o2-analysis-lf-strangenesstofpid -b --configuration json://configuration.json |
o2-analysis-event-selection-service -b --configuration json://configuration.json
o2-analysis-multcenttable -b --configuration json://configuration.json |
o2-analysis-propagationservice -b --configuration json://configuration.json --aod-file @input_data.txt
```

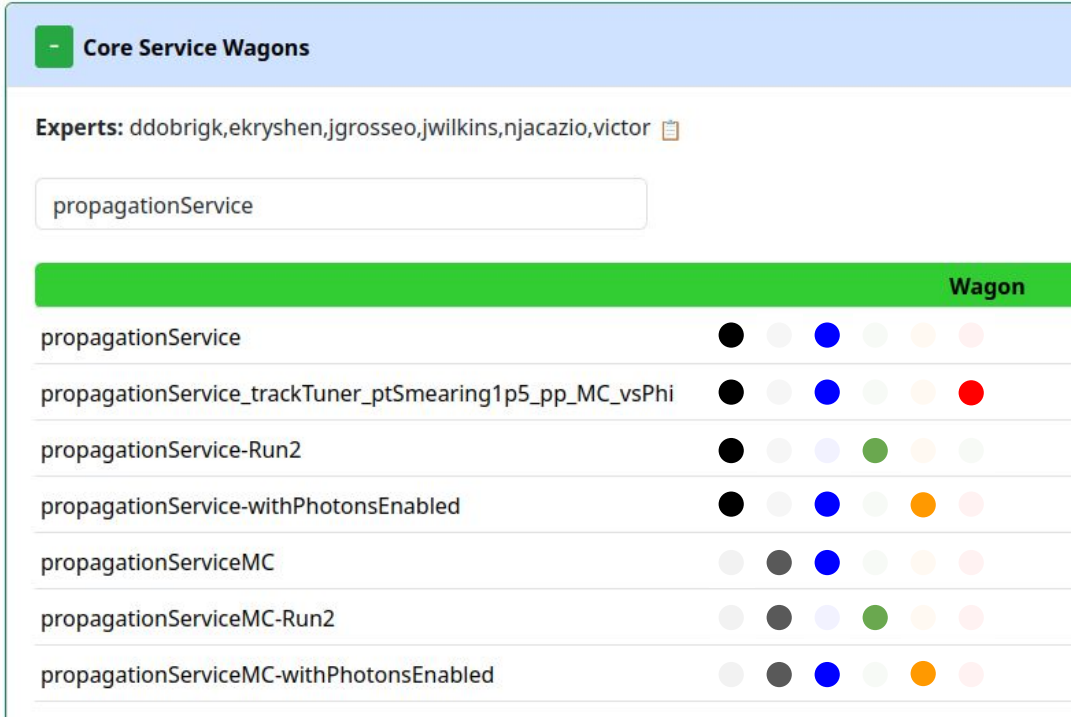
(Optional) Dependencies

Strangeness table producer task

Produces the following tables:

- **V0Datas**: regular V0 analysis table
- **CascDatas**: regular cascade analysis table
- **TraCascDatas**: tracked cascade analysis table
- **KFCascDatas**: KF cascade analysis table

Strangeness builder wagons in Hyperloop



Core services in Hyperloop:

- **Data**
- **Monte Carlo**
- **Run 3**
- **Run 2**
- **With V0s from TPC-only tracks** (for photon analysis)
- **Using the track tuner**

Include only one of them as a dependency to your task

Use as a reference configuration
→ adapt it according to your needs

Strangeness derived data

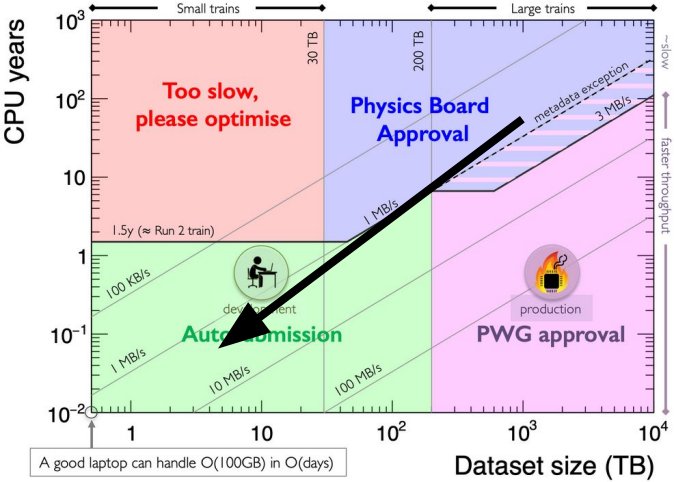
- **Derived data** are (*derived*) AO2D.root files produced with a task that creates given tables by processing other (*original*) AO2D.root files

- 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)

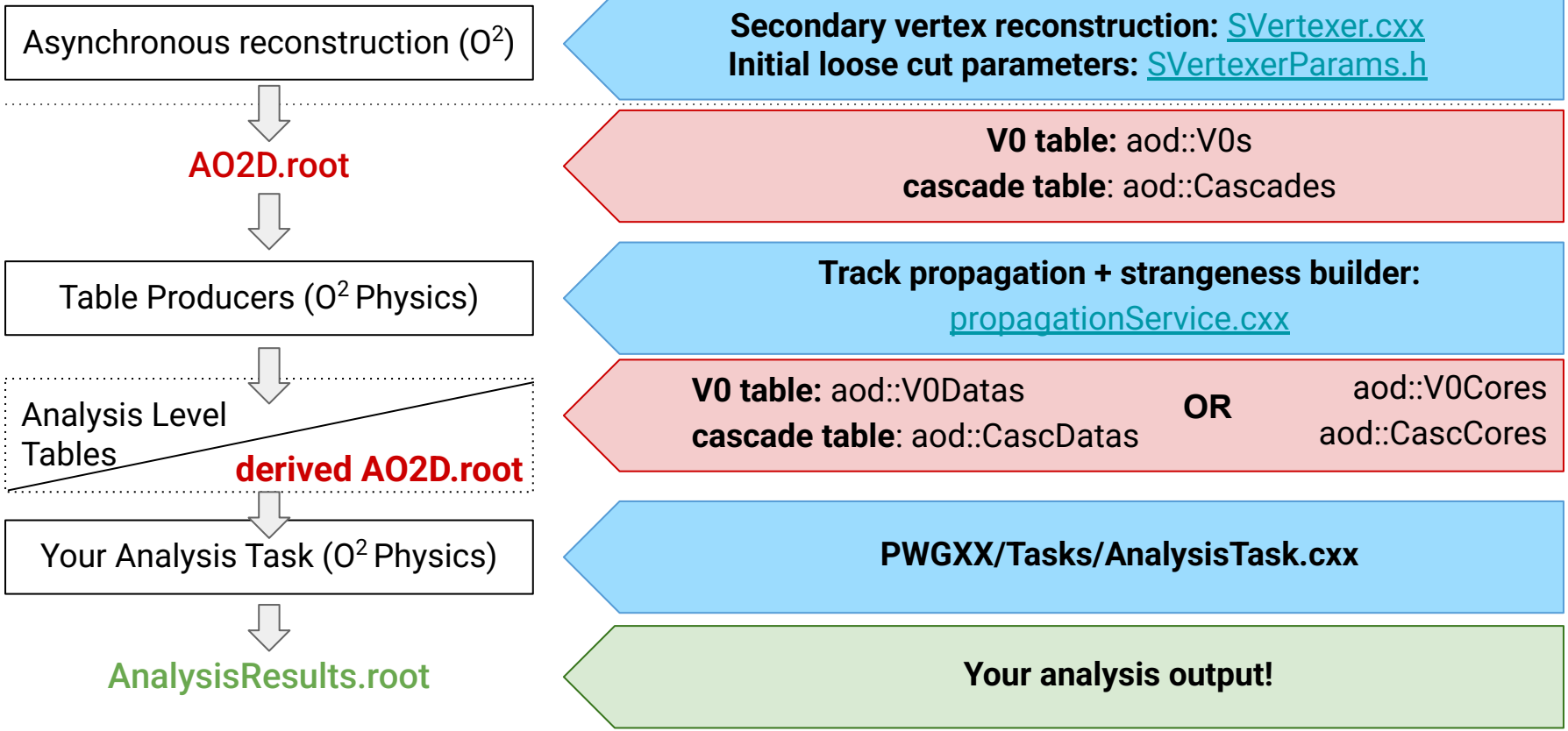
- Two types of derived data:

  - **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



The complete strangeness analysis workflow



How to access the strangeness derived tables?

Include strangeness header to access to the strangeness data model

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

Header needed
`LFStrangenessTables.h`

Subscribe and access the strangeness table content

```
void process(aod::V0Datas const& v0s)
{
  for (const auto& v0 : v0s) {
    v0.mK0Short();
    v0.mLambda();
    v0.mAntiLambda();
  }
}
```

```
void process(aod::V0Cores const& v0s)
{
  for (const auto& v0 : v0s) {
    v0.mK0Short();
    v0.mLambda();
    v0.mAntiLambda();
  }
}
```

```
void process(aod::CascDatas const& cascades)
{
  for (const auto& cascade : cascades) {
    cascade.mXi();
    cascade.mOmega();
  }
}
```

```
void process(aod::CascCores const& cascades)
{
  for (const auto& cascade : cascades) {
    cascade.mXi();
    cascade.mOmega();
  }
}
```

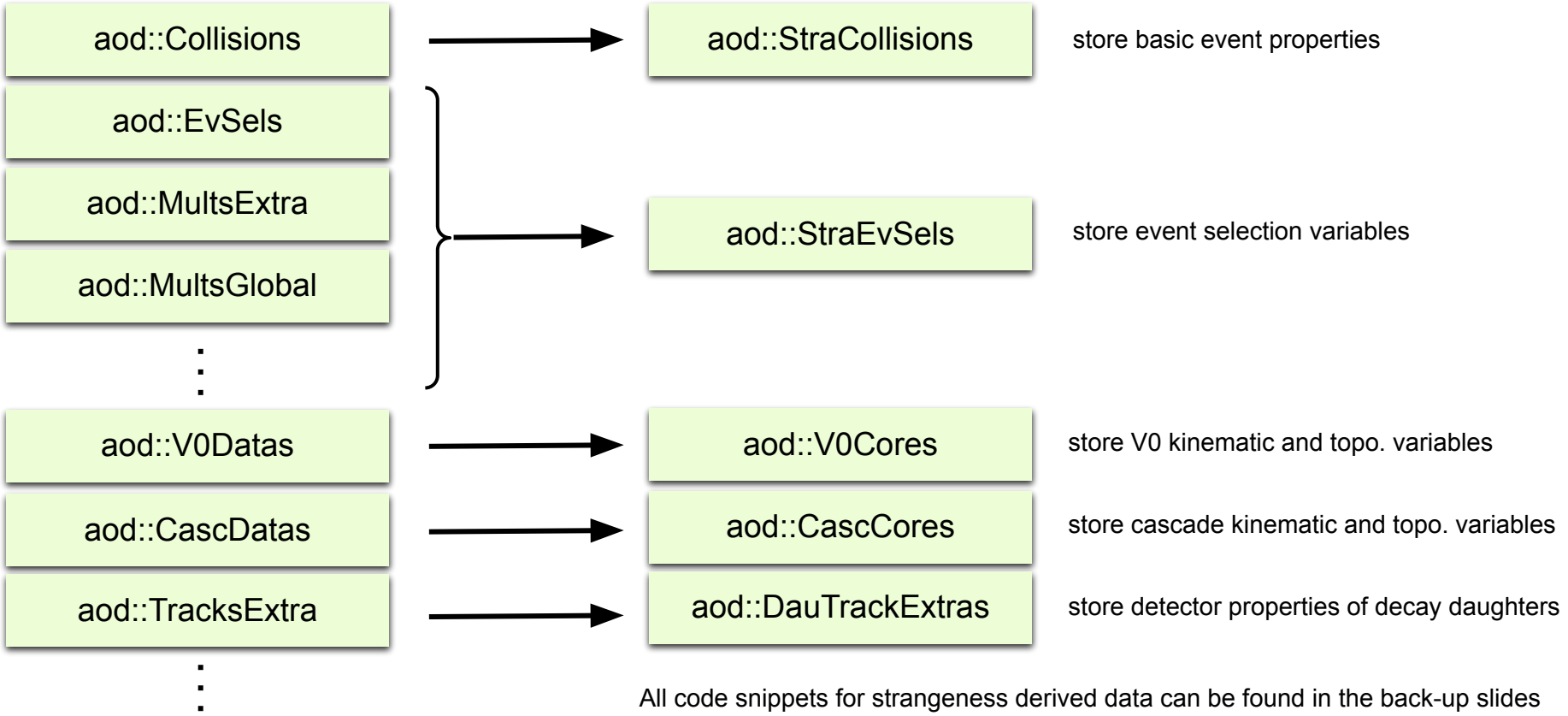
Original data

Derived data

Only difference: table subscription
+some getters…

Original data Vs strangeness derived data model

The strangeness derived data model follows a similar structure as the original data



All code snippets for strangeness derived data can be found in the back-up slides

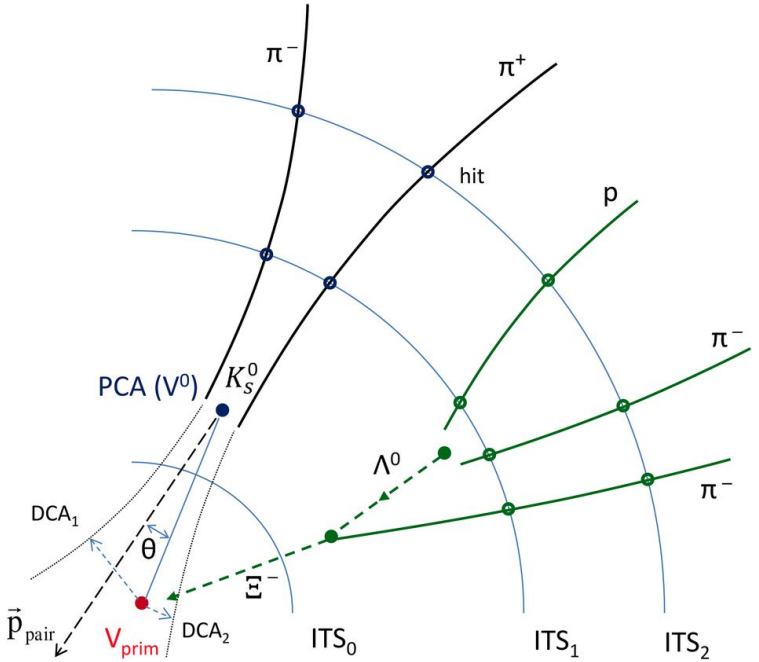
Inner working of the DCAfitter

- Depending on your analysis, there might not exist a builder
  → need to write your own builder based on the DCAfitter = central tool to determine the DCA between tracks

- **Input:** track parametrisation with the associated covariance matrix (o2::track::TrackParCov)

- **DCAfitter:**
  → Finds the spatial crossing points of the track helices
  → Propagate the tracks close to their crossing point
  → Iteratively minimizes their distance

- **Output:**
  → DCA between the tracks
  → Point of Closest Approach (PCA)
  → track parameters at PCA



How to set up the DCAfitter?

```
#include "DCAFitter/DCAFitterN.h"
```

Header needed
DCAFitterN.h

```
Service<o2::ccdb::BasicCCDBManager> ccdb;
o2::base::MatLayerCylSet* lut;
o2::vertexing::DCAFitterN<2> fitter;
```

Add the DCAfitter as data member of your task

```
void init(InitContext const&) {
   fitter.setPropagateToPCA(true);
   fitter.setMaxR(200.);
   fitter.setMinParamChange(1e-3);
   fitter.setMinRelChi2Change(0.9);
   fitter.setMaxDZIni(1e9);
   fitter.setMaxDXYIni(4.0f);
   fitter.setMaxChi2(1e9);
   fitter.setUseAbsDCA(true);
   fitter.setWeightedFinalPCA(false);
   fitter.setMatCorrType(o2::base::Propagator::MatCorrType::USEMatCorrLUT);
   fitter.setBz(-999.9f); // mag field has to be set later
}
```

Provide the DCAfitter configuration

3 material correction types:

- USEMatCorrNONE: no corrections
- USEMatCorrTGeo: from TGeo
- USEMatCorrLUT: from Look-up Tables

See back-up slides on how to load material LUT and magnetic field

How to execute the DCAfitter?

```
auto posTrackCov = getTrackParCov(posTrack);
auto negTrackCov = getTrackParCov(negTrack);

int nCand = 0;
try {
   nCand = fitter.process(posTrackCov, negTrackCov);
} catch (...) {
   continue;
}
if (nCand == 0) {
   continue;
}

fitter.propagateTracksToVertex();
if (!fitter.isPropagateTracksToVertexDone()) {
   continue;
}
```

Extract aod::track::TrackParCov from tracks from `trackUtilities.h`

Execute the DCAfitter for two considered tracks

Propagate tracks to the decay vertex position

How to extract info from the DCAfitter?

```
auto& posPropTrack = fitter.getTrack(0);
auto& negPropTrack = fitter.getTrack(1);

std::array<float, 3> momPos;
std::array<float, 3> momNeg;

posPropTrack.getPxPyPzGlo(momPos);
negPropTrack.getPxPyPzGlo(momNeg);

auto mK0Short = RecoDecay::m(
       std::array{momPos, momNeg},
       std::array{o2::constants::physics::MassPionCharged,
                  o2::constants::physics::MassPionCharged});
```

Extract momentum components of each daughter tracks at the decay vertex

Calculate invariant mass

```
auto dca = std::sqrt(fitter.getChi2AtPCACandidate());
```

Extract the DCA between decay daughters

```
const auto& vtx = fitter.getPCACandidate();
```

Get decay vertex coordinates

Summary

In this lecture you learned:

- how to **subscribe** to the strangeness tables in your analysis
- **access to** topological and kinematic variables, PID and MC **information**
- **write your own secondary vertex finder**

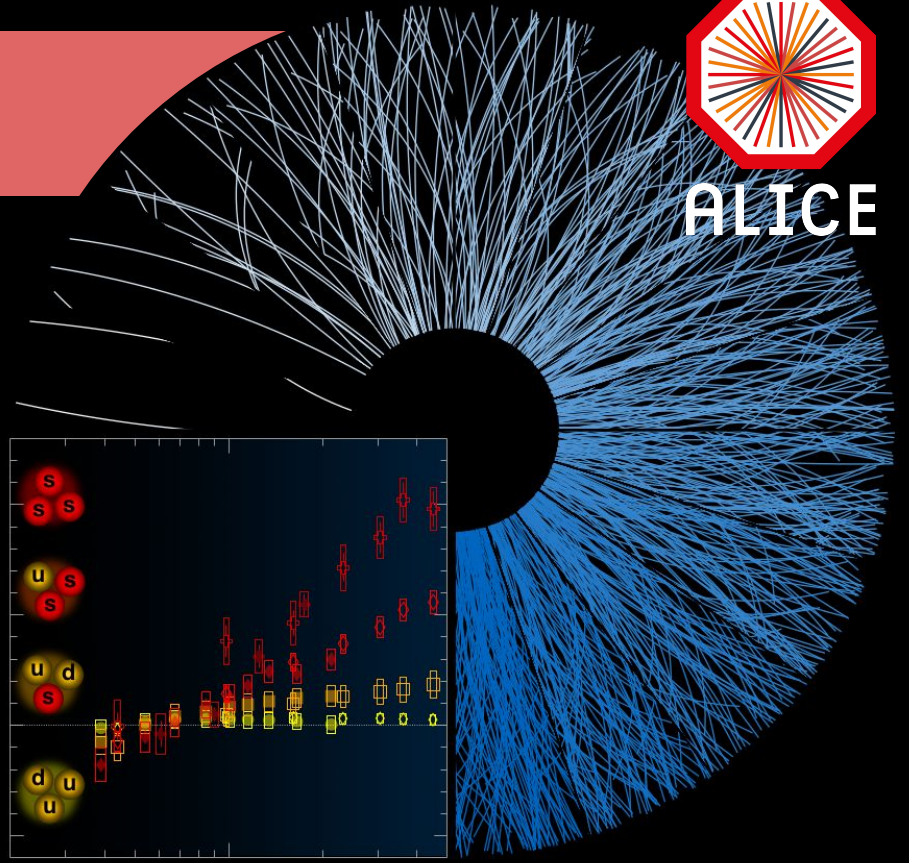
The hands-on session focuses on an analysis in OO collisions using derived data

Code is already present in your **O2Physics**

→ please update the folder before the HANDS-ON session

See you at the hands-on session of Wednesday!

Back-up slides



ALICE

How to access track properties? (derived data)

- V0/cascade tables reference the track table allowing access information about daughter tracks → if you try to check the PID of daughter tracks with reference to that table the code will fail!

```
float nsigma_bach = cascade.bachTrackExtra.tpcNSigmaPi();
float nsigma_pos = cascade.posTrackExtra.tpcNSigmaPr();
float nsigma_neg = cascade.negTrackExtra.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::DauTracksExtras, aod::DauTrackPIDs>

void process(soa::Join<aod::CascCores, aod::CascExtras> const& cascades,
             DaughterTracks const&)
 {
   for (const auto& cascade : cascades) {
     float nsigma_bach = cascade.bachTrackExtra_as<DaughterTracks>().tpcNSigmaPi();
     float nsigma_pos = cascade.posTrackExtra_as<DaughterTracks>().tpcNSigmaPr();
     float nsigma_neg = cascade.negTrackExtra_as<DaughterTracks>().tpcNSigmaPi();
   }
 }
```

Works!

How to access TOF information? (derived data)

Include strangeness PID header to access to the strangeness data model

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

Header needed
LFStrangenessPIDTables.h

Subscribe to the strangeness PID tables

```
void process(soa::Join<aod::CascCores,aod::CascTOFNSigmas,aod::CascTOFPIDs> const& cascades)
 {
   for (const auto& cascade : cascades) {
     if (cascade.bachelorHasTOF() && std::abs(cascade.tofNSigmaXiPi()) > NSigmaTOF) {
       continue;
     }
     if (cascade.positiveHasTOF() && std::abs(cascade.tofNSigmaXiLaPr()) > NSigmaTOF) {
       continue;
     }
     if (cascade.negativeHasTOF() && std::abs(cascade.tofNSigmaXiLaPi()) > NSigmaTOF) {
       continue;
     }
   }
 }
```

Subscribe to **aod::CascTOFNSigmas** and **aod::CascTOFPIDs** tables

```
if (!cascade.tofXiCompatibility(NSigmaTOF)) {
   continue;
}
```

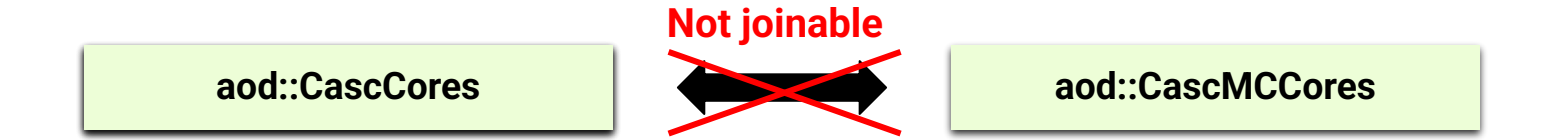
Apply TOF PID selections **if** the track has a hit in TOF

How to access MC information? (Derived data)

→ **reconstructed** info about cascades

→ **generated** info about cascades

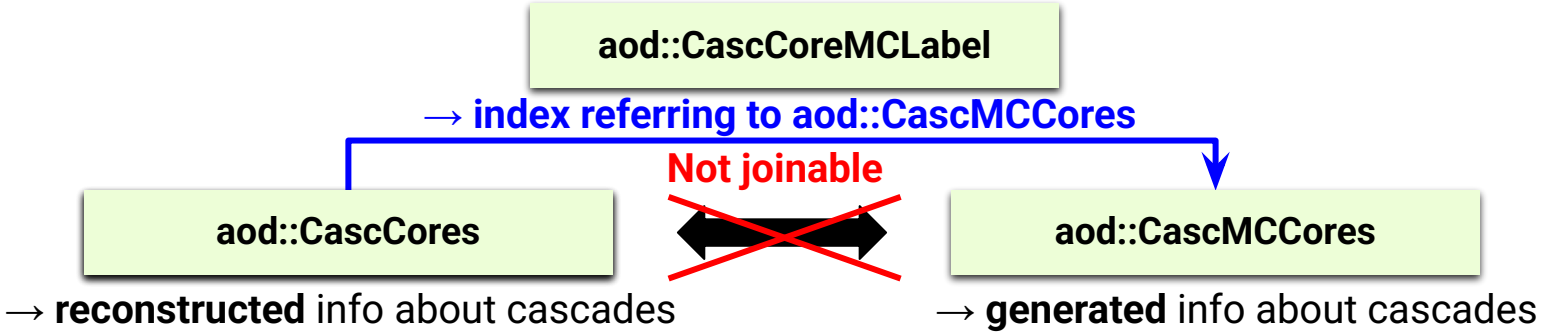
How to access MC information? (Derived data)



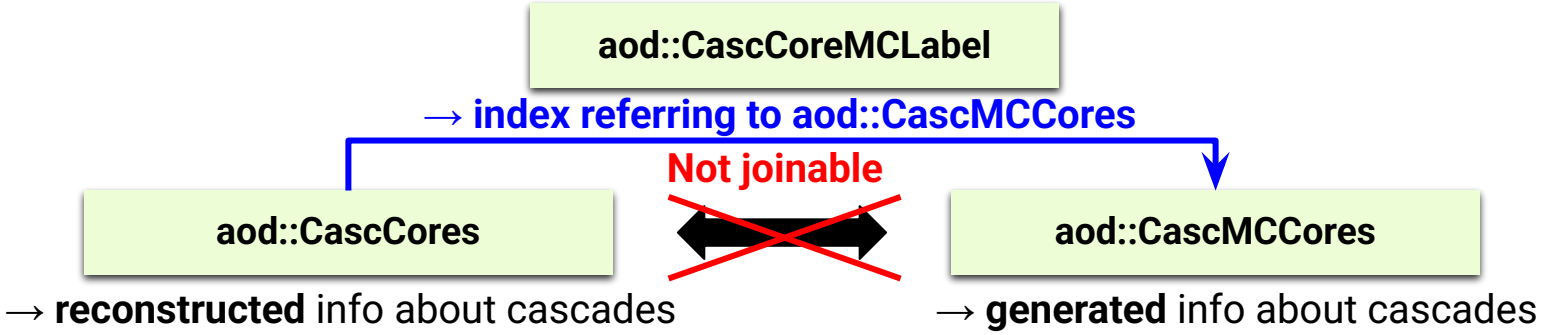
→ **reconstructed** info about cascades

→ **generated** info about cascades

How to access MC information? (Derived data)



How to access MC information? (Derived data)




```
void process(soa::Join<aod::CascDatas, aod::CascCoreMCLabels> const& cascades,
             aod::CascMcCores const&)
{
  for (const auto& cascade : cascades) {
    if (!cascade.has_cascMCCore()) {
      continue;
    }
    auto cascmcParticle = cascade.cascMCCore_as<aod::CascMCCores>();
    cascmcParticle.pdgCode();
  }
}
```

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

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

Advanced feature: how to load magnetic field

```
#include "DCAFitter/DCAFitterN.h"
```

Header needed
DCAFitterN.h

```
Service<o2::ccdb::BasicCCDBManager> ccdb;
o2::vertexing::DCAFitterN<2> fitter;
```

Add the DCAfitter as data member of your task

```
o2::parameters::GRPObject* grpo = ccdb->getForRun<o2::parameters::GRPObject>("GLO/GRP/GRP",
collision.runNumber());
o2::base::Propagator::initFieldFromGRP(grpo);
// Fetch magnetic field from ccdb for current collision
magField = grpo->getNominalL3Field();
fitter.setBz(magField);
```

Load magnetic field

Advanced feature: how to load material LUT?

```
#include "DCAFitter/DCAFitterN.h"
```

Header needed
DCAFitterN.h

```
Service<o2::ccdb::BasicCCDBManager> ccdb;
o2::base::MatLayerCylSet* lut;
o2::vertexing::DCAFitterN<2> fitter;
```

Add the DCAfitter as data member of your task

```
if (!lut) {
   LOG(info) << "Loading material look-up table for timestamp: " << collision.runNumber();
   lut = o2::base::MatLayerCylSet::rectifyPtrFromFile(ccdb->getForRun<o2::base::MatLayerCylSet>("GLO/Param/MatLUT",
collision.runNumber()));
} else {
   LOG(info) << "Material look-up table already in place. Not reloading.";
}
LOG(info) << "Setting global propagator material propagation LUT";
o2::base::Propagator::Instance()->setMatLUT(lut);
```

Load material look-up tables

Advanced feature: 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-trackselection  ${OPTION} |
o2-analysis-ft0-corrected-table  ${OPTION} |
o2-analysis-multcenttable  ${OPTION} |
o2-analysis-event-selection-service  ${OPTION} |
o2-analysis-pid-tpc-service  ${OPTION} |
o2-analysis-pid-tof-merge  ${OPTION} |
o2-analysis-propagationservice  ${OPTION} |
o2-analysis-lf-strangederivedbuilder  ${OPTION} --aod-file @input_data.txt --aod-writer-json OutputDirector.json >"$LOGFILE"
2>&1

# report status
rc=$?
if [ $rc -eq 0 ]; then
   echo "No problems!"
   mkdir -p "${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}/results/${STEP}/${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