Event selection for UPC events in Run 3

Sasha Bylinkin
University of Bergen

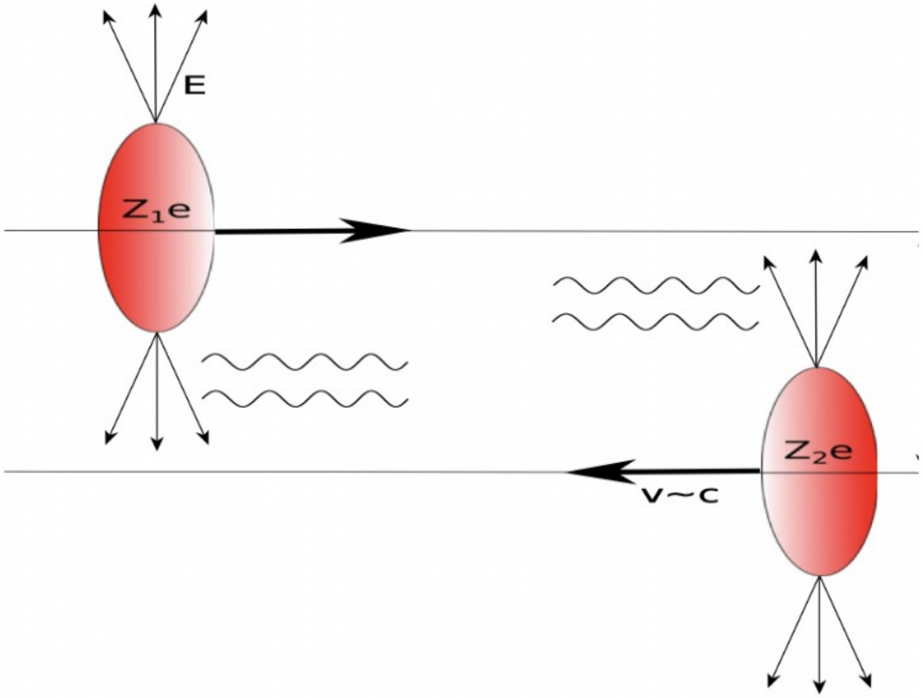
O2 Analysis Tutorial, 16th October 2024

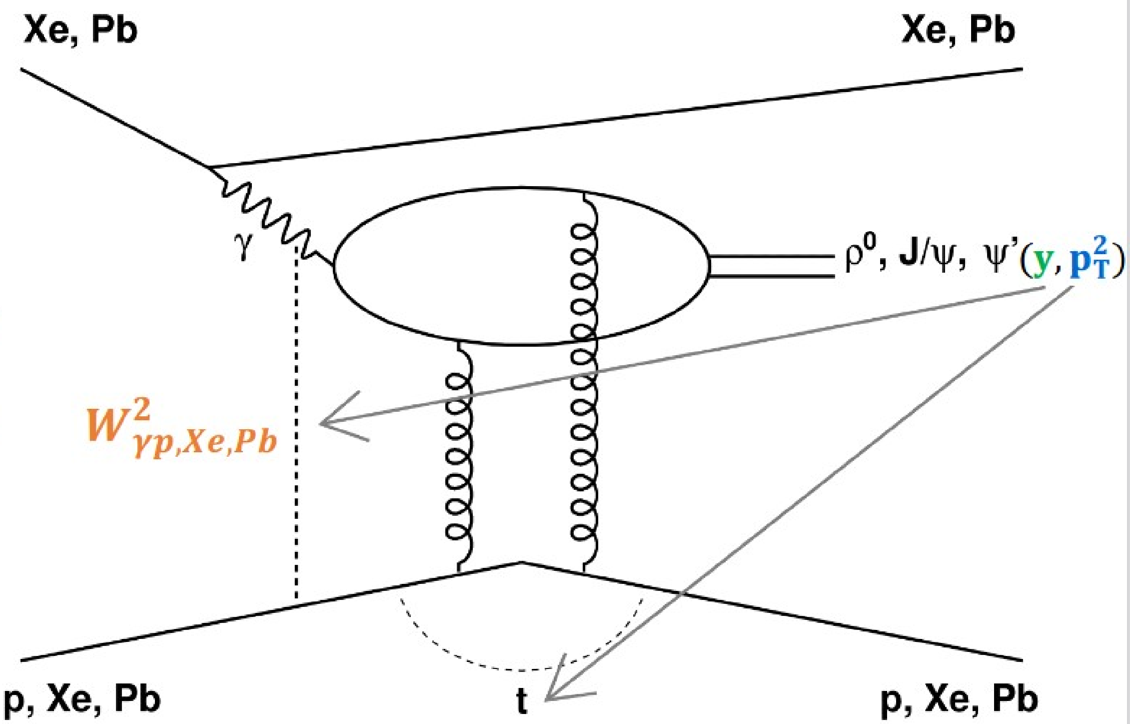
Outline

- Physics motivation
  - From exclusive to inclusive events
- Event Selection
  - Double Gap and Single Gap events
- Skimming
  - Derived data model
  - Analysis tools

Photon induced processes in heavy ion collisions

- Ultrarelativistic moving nuclei produce strong electromagnetic (EM) fields that can be treated as a **quasi-real photons flux**





In **Runs 1 & 2** ALICE extensively measured *exclusive* particle production:

→ only 1 particle produced in the central detector (Muon Arm) and nuclei do not break up

→ easy to trigger such events by controlling nuclear break up

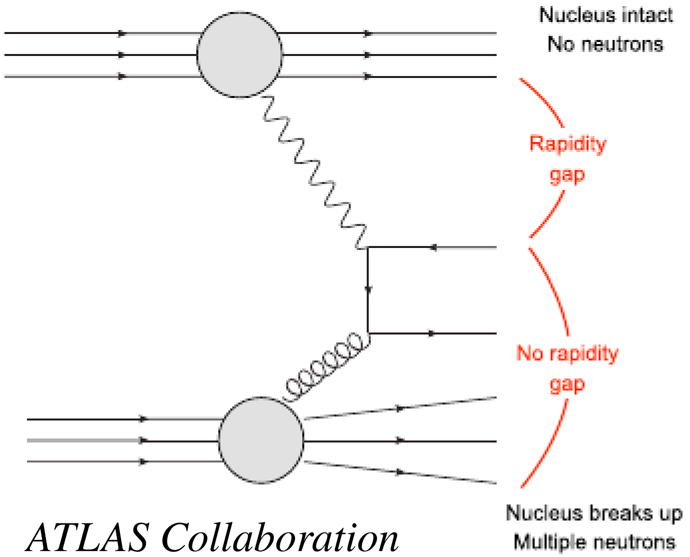
→ veto activity in the forward detectors (V0, AD @ Run 2 => FIT @ Run 3)

General photonuclear interaction

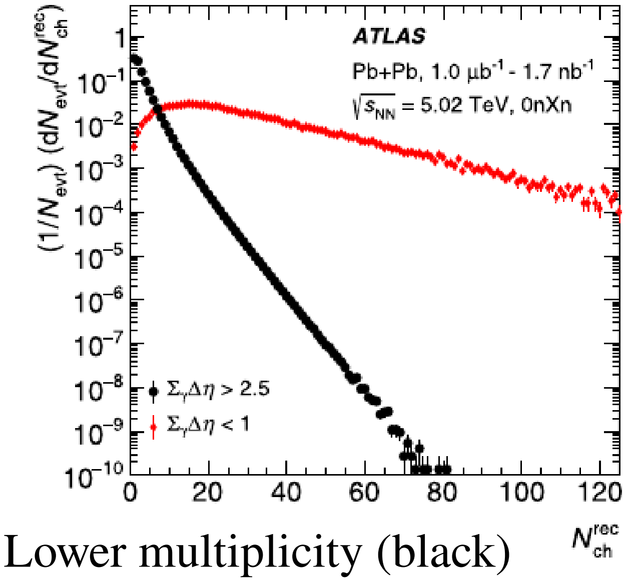
One nucleus breaks up producing many particles, but the second remains intact!

1) There is a rapidity gap, void of particles, on the side of the photon-emitting nucleus.
This is the main experimental signature.

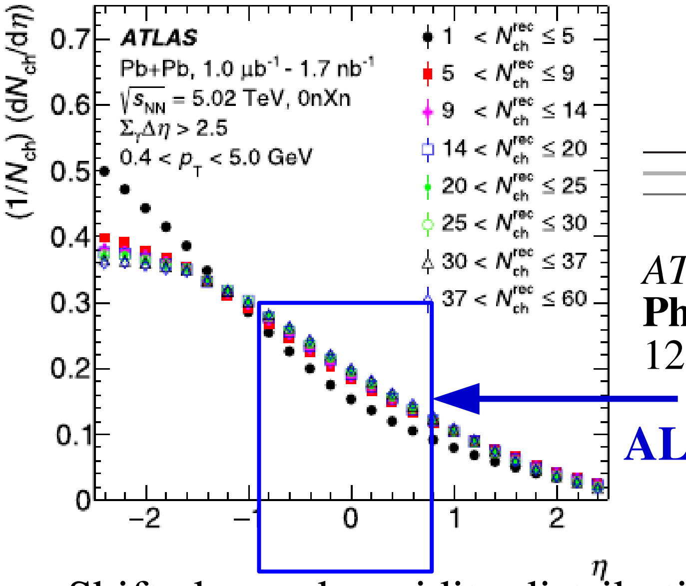
2) The particle production is not centered at mid-rapidity
but shifted to the side of the target nucleus.



Multiplicity and pseudo-rapidity distributions for γ+Pb events.



Lower multiplicity (black) than in hadronic events (red).



ALICE acceptance

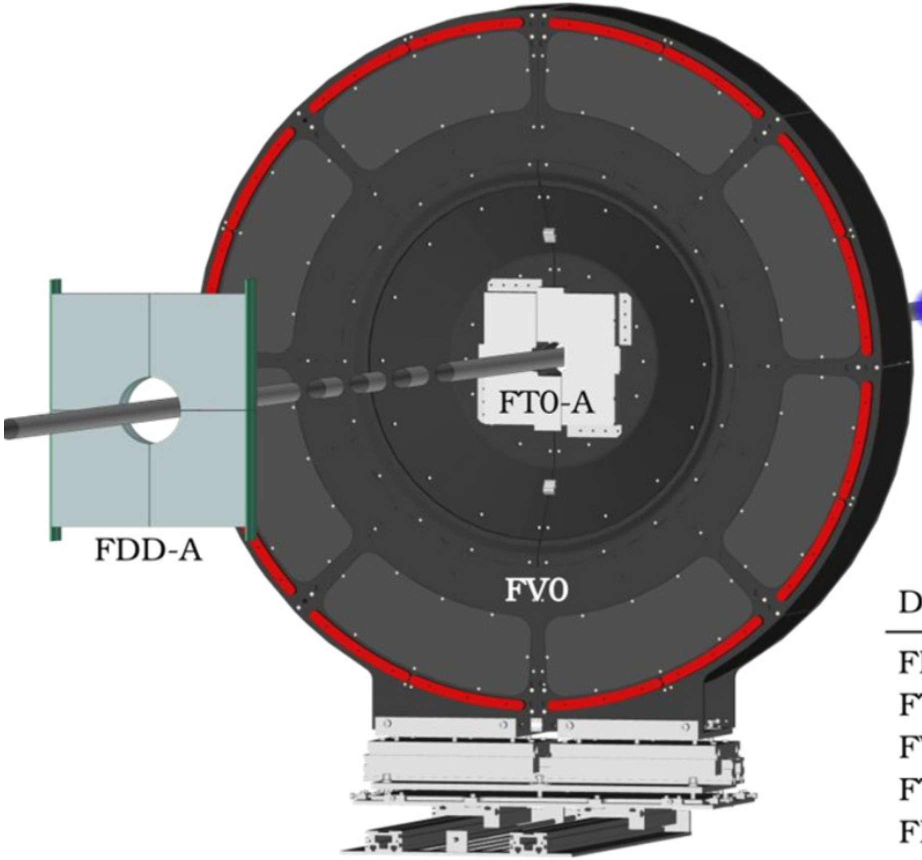
Shifted pseudorapidity distribution.

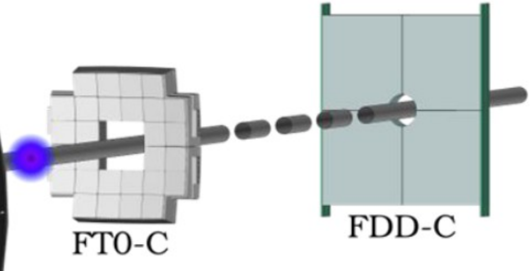
ATLAS Collaboration
Phys. Rev. C 104, 014903
12 July 2021

Selecting events: FIT Detectors

FIT = Fast Interaction Trigger: precise collision time.

FT0-A & FT0-C: Quartz Cherenkov radiators - *25ps*
FV0: Scintillator - *200ps*
FDD-A & FDD-C: Scintillator - *425ps*

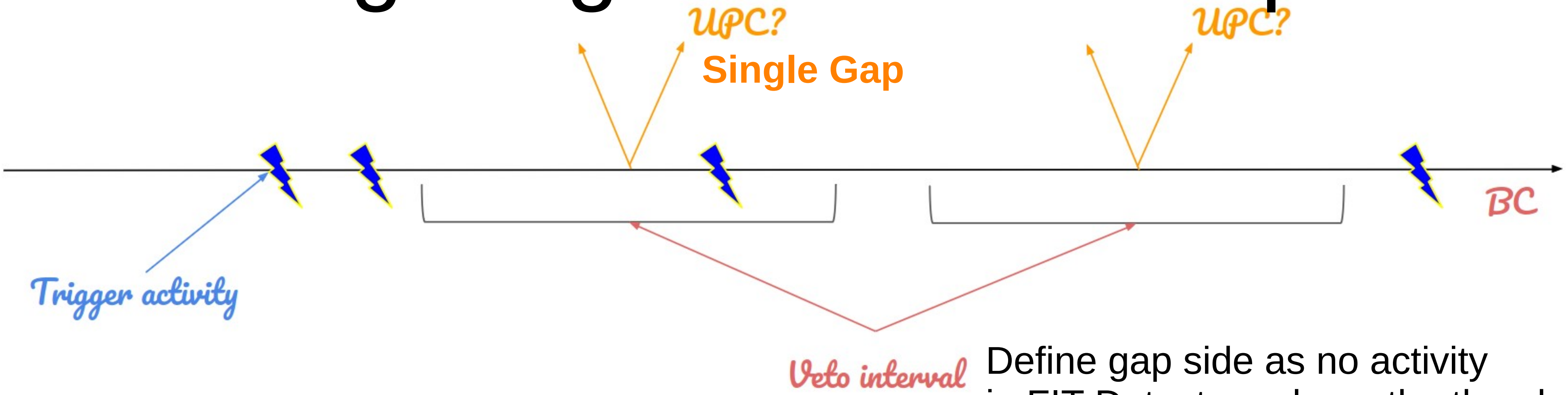




Bunch Crossing: *25ns!*

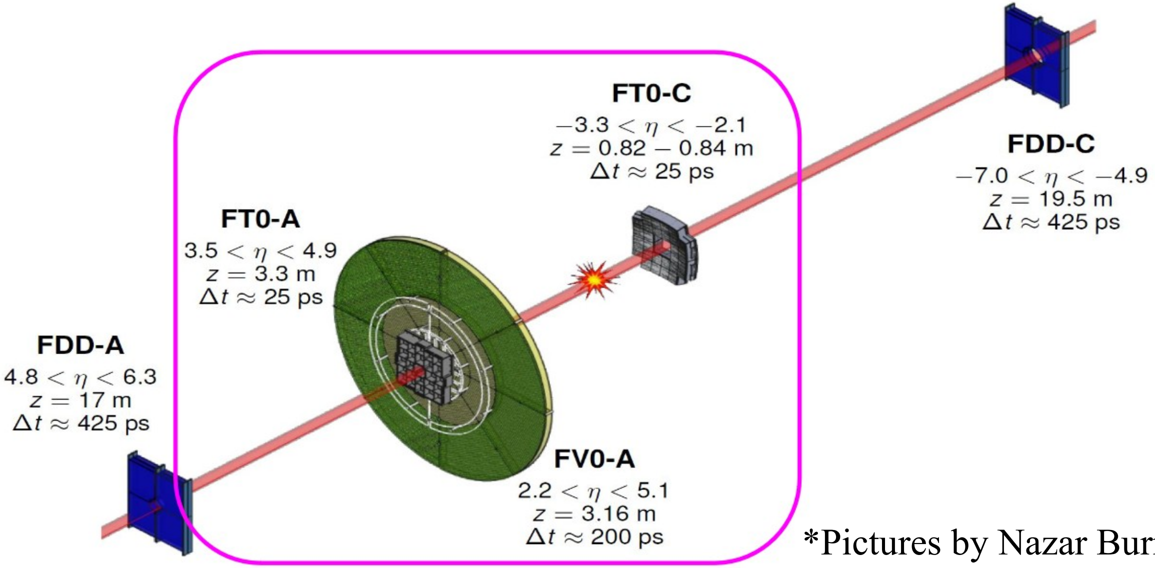
Detector	z [mm]	$\eta_{\min}$	$\eta_{\max}$
FDD-A	17000	4.8	6.3
FT0-A	3305	3.5	4.9
FV0	3160	2.2	5.0
FT0-C	-843	-3.3	-2.1
FDD-C	-19500	-7.0	-4.9

Selecting Single & Double Gap events



Define gap side as no activity
in FIT Detectors above the threshold
in the neighboring BCs: ±2

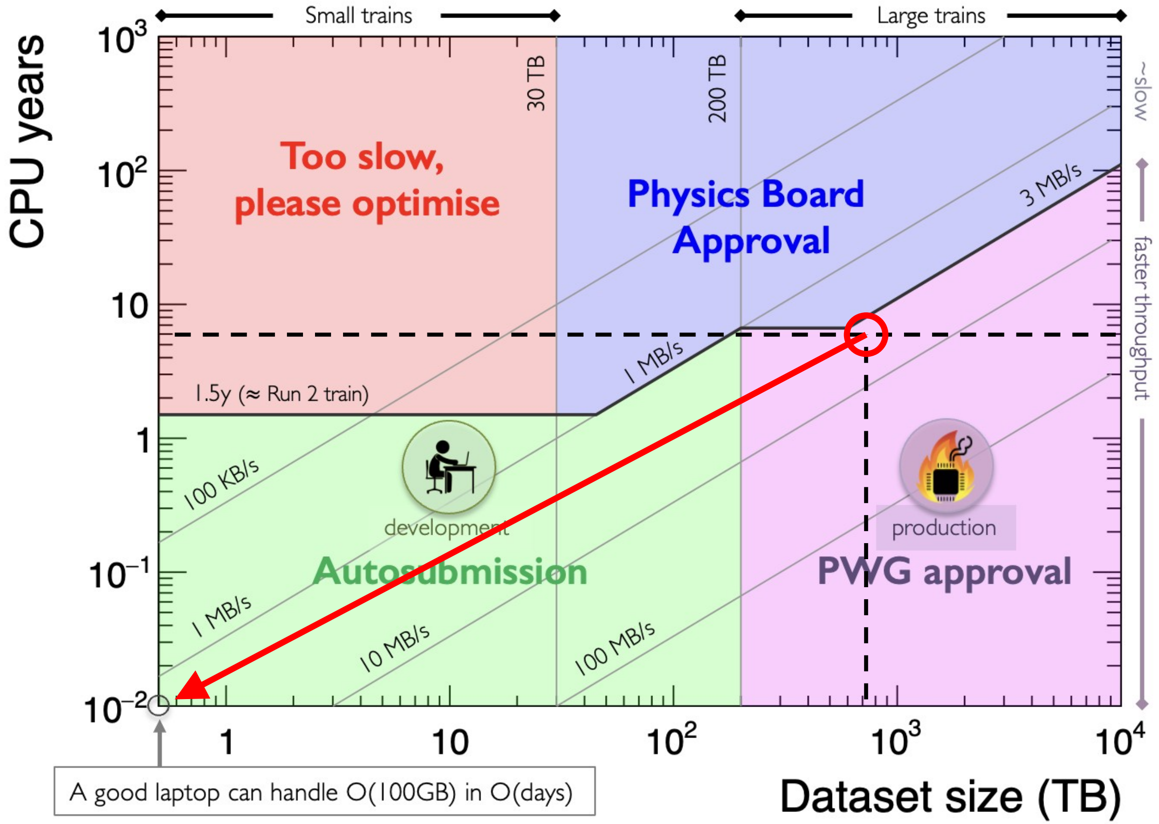
@ one of the sides (Single Gap)
or both sides of the event (Double Gap)



*Pictures by Nazar Burmasov

Skimming and derived data

- Derived dataset from pass2_upc is available (pass4 is coming soon) https://alimonitor.cern.ch/hyperloop/view-dataset/866



Full Golden UPC Sample:

Input size:	750 TB
Reduction factor:	1200
Output size:	650 GB

Derived data contains necessary information for various analyses and allows much faster execution.

Pass4 will have one common reconstruction → no _upc.
See slides 17-19 for details.

Bonus: Hyperloop Service Wagon

+/- **Service Wagons HF**

+/- **Hyperloop on-the-fly simulation core service wagons**

+/- **Service Wagons JE**

+/- **Service Wagons UD**

Experts: rolavick,abylinki

JIRA : OHSW-6

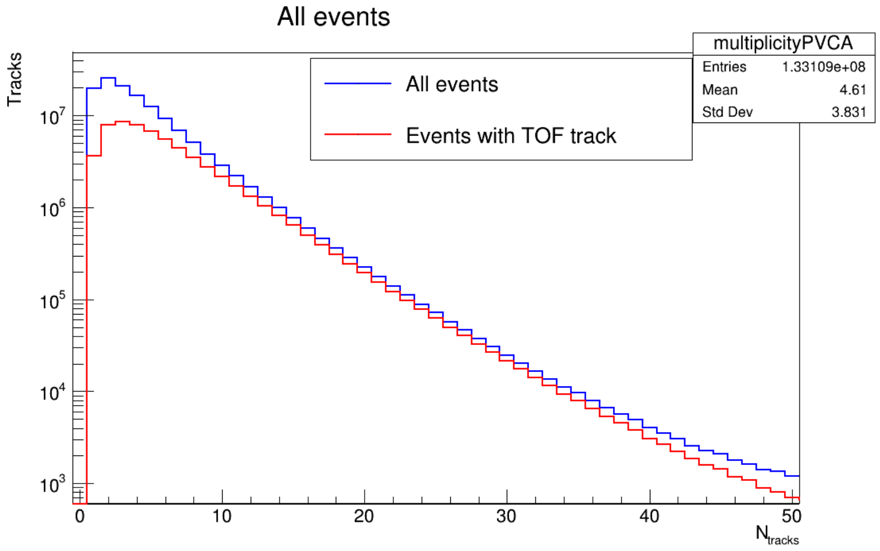
Wagon
SG_Producer
SG_Producer_MC
SG_Producer_MC_notrackloops
SG_Producer_MC_onlyreco
SG_Producer_MC_onlyreco_notrackloops

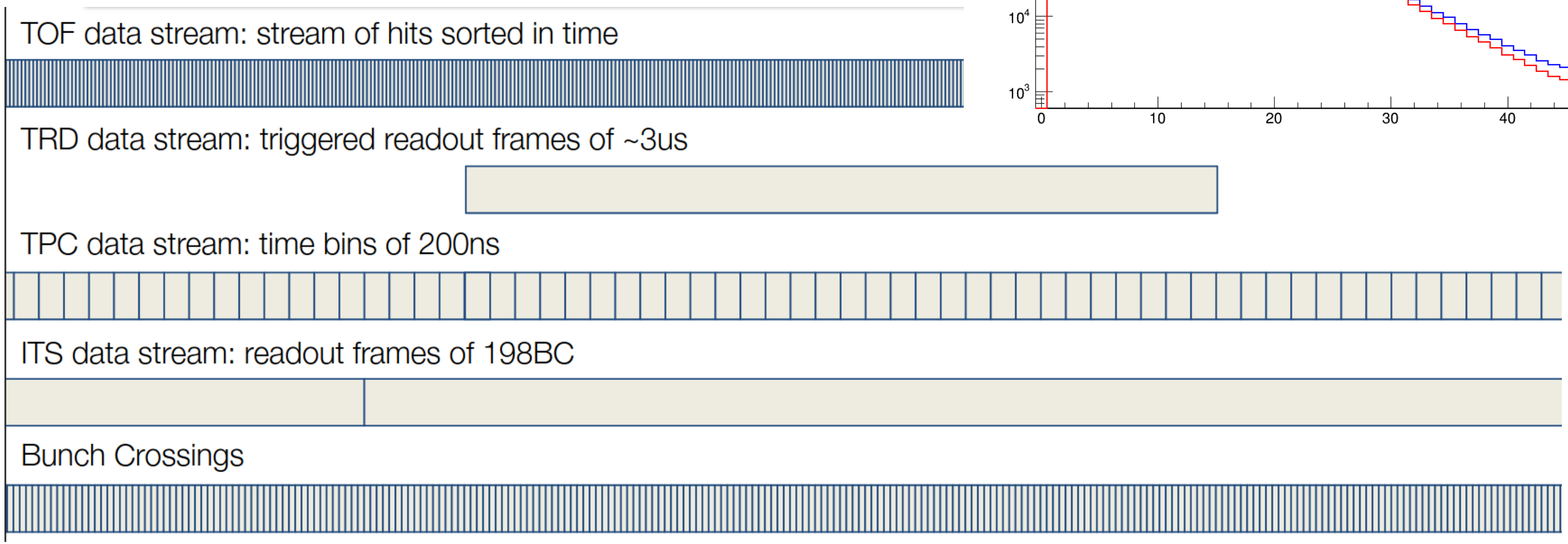
Quick Access: pvn Service Wagons UD

Service wagon doesn’t save the data (UD Tables), but provides the events selection for other PAG groups (Strangeness, LF ...)

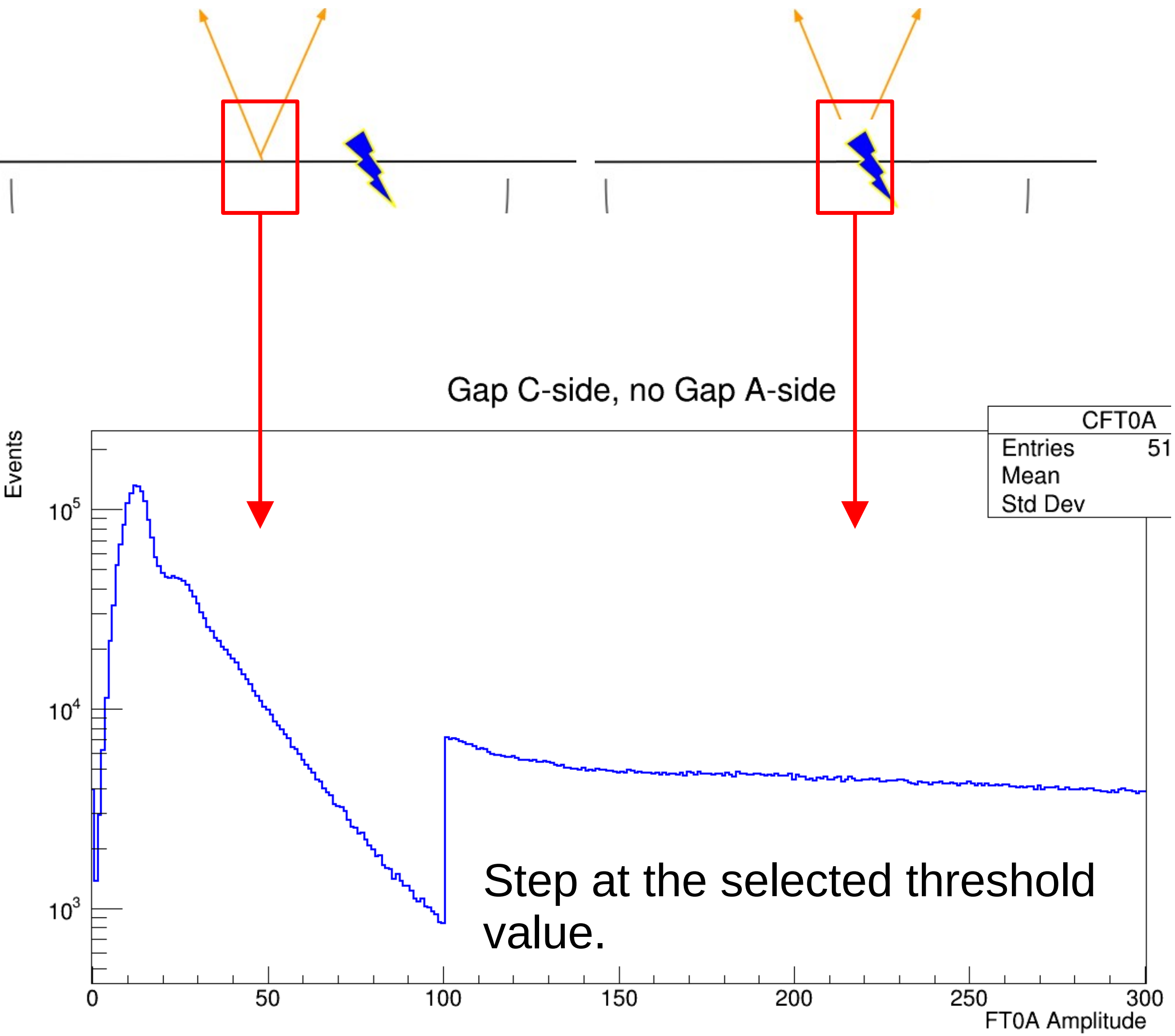
Collision BC vs FIT BC

Collision time is determined from tracks associated to a primary vertex.
UPC events are characterized by low multiplicities and often low transverse momenta tracks.
→ If we have only ITS and TPC tracks, the time precision is **200ns = 8 Bunch Crossings.**





Collision BC vs FIT BC



In ~**50%** of events the collision is associated with the wrong BC.

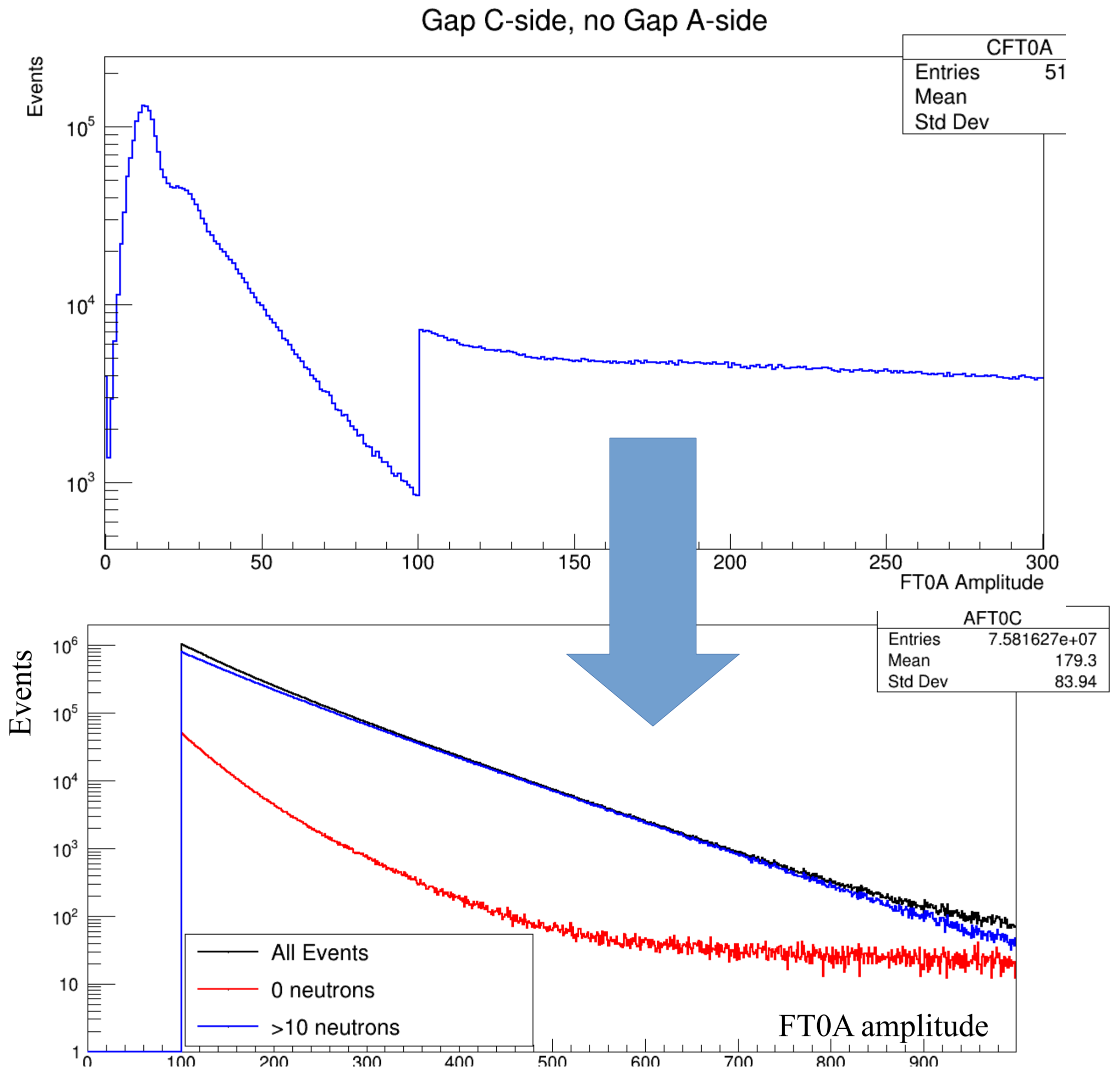
Good to see background from the e+e- pileup

***New feature in SGSelector.h*:**

```
result.bc = &newbc;
result.value = gA && gC ? 2 : (gA ? 0 : 1);
return result;
```

The selector return the closest BC with the FIT activity above the threshold, which is used to fill in FIT and ZDC collision info.

Collision BC vs FIT BC



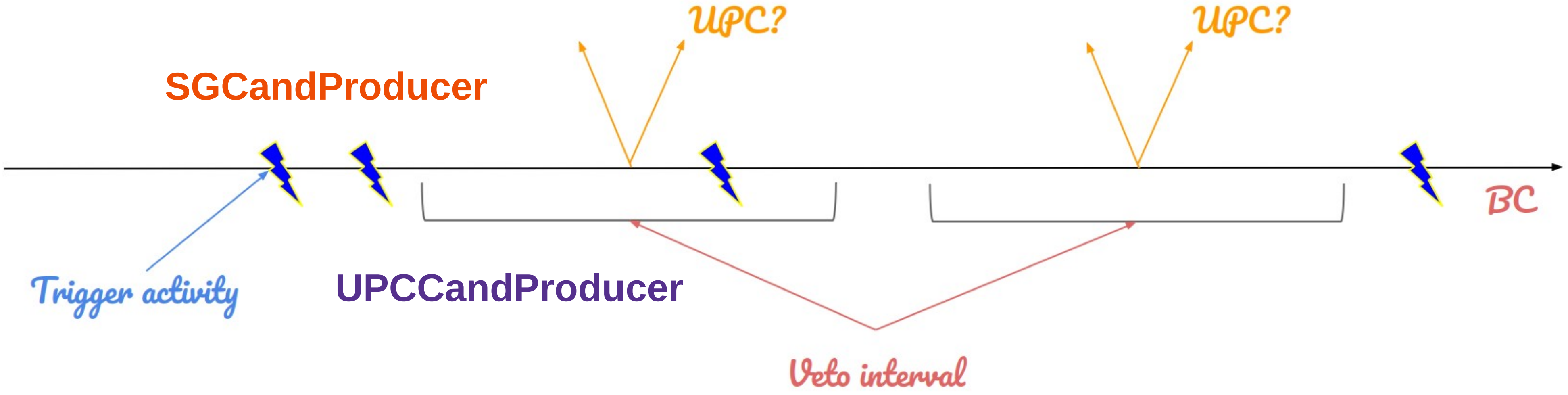
***New feature in SGSelector.h*:**

```
result.bc = &newbc;
result.value = gA && gC ? 2 : (gA ? 0 : 1);
return result;
```

The selector return the closest BC with the FIT activity above the threshold, which is used to fill in FIT and ZDC collision info.

→ If the nucleus breaks up in an event, then we save the detector's amplitudes for the bunch crossing in which it actually happened.

SGCandProducer vs UPCCandProducer



SGCandProducer: Starts with the collision information

→ Needs 2 tracks (Central Barrel) associated to a common vertex.

→ Has some uncertainty in assigning BC from the collision

UPCCandProducer: Starts with the BC information

→ Not all BCs have an associated collision

→ Particularly useful for Forward Tracks (Muon Arm) that do not create a vertex

Both producers create similar UD Tables as an output.

Running SGCandProducer locally

Needed:

- Successful O2Physics Installation
- Input file (AO2D.root)
- OutputDirector.json → Defines which derived data tables will be saved
- runsg.sh → shell script to actually run the producer
- Configuration.json → configuration file with the skimming parameters

Skimming parameters

• mNDtcoll	col. time resolution used for ΔBCs	= 1
• mMinNBCs	min. value of ΔBcs	= 2
• mITSOnlyTracks	saving ITS-only tracks	= 1
• m[Min,Max]NTracks	min./max. number of PV tracks	[2, 100]
• m[Min,Max]Pt	min./max. track pT	[0.1 100] GeV
• m[Min,Max]Eta	min./max. track η	[-0.9, 0.9]
• mFITAmpLimits	amplitude thresholds for FIT	FT0A = 150 FT0C = 50

Skimming parameters are fixed for now...

runsg.sh script

Configuration filename

Service wagons

OutputDirector name

Input Filename

albyl1023@ift-b7lb1t3: ~/alice

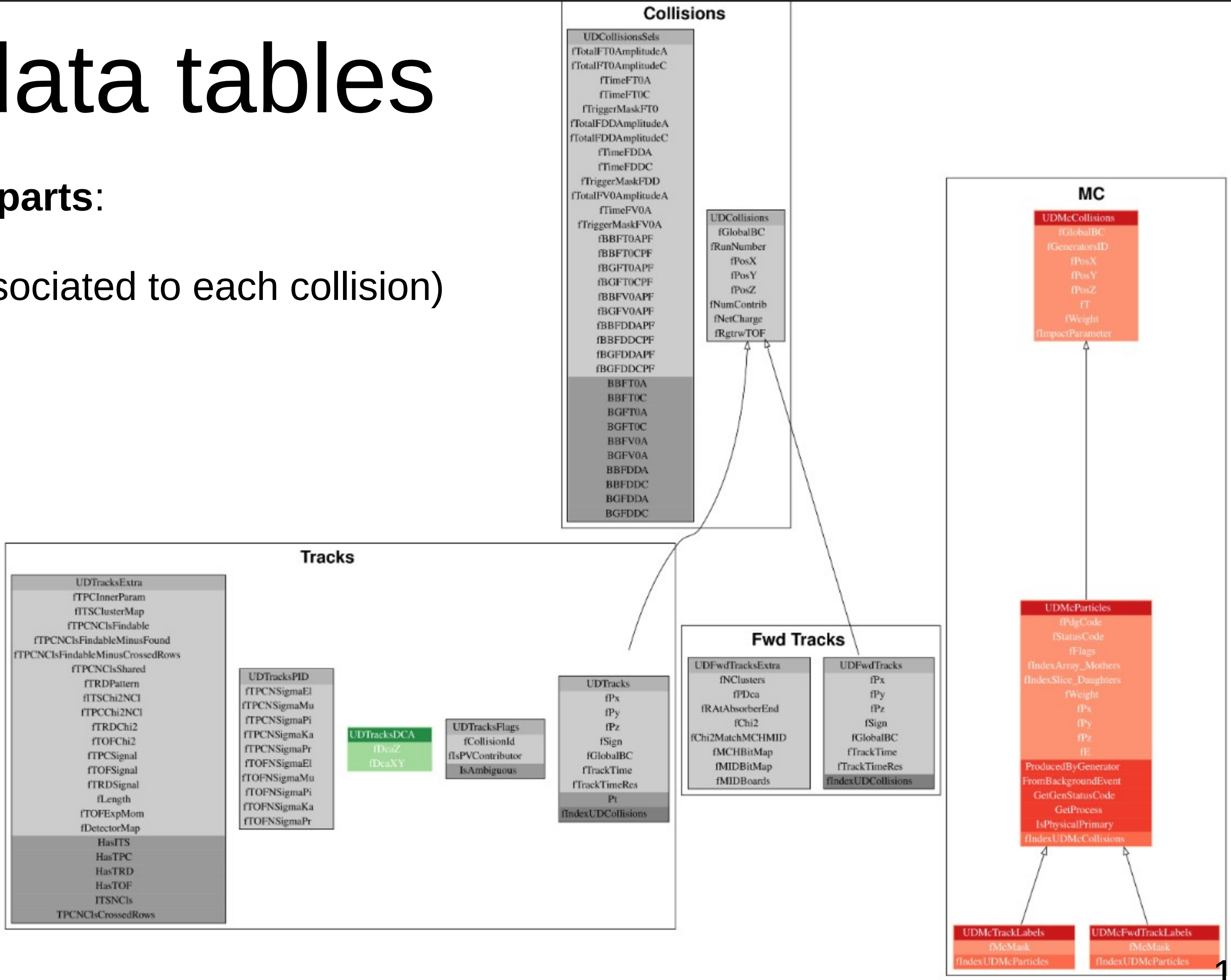
```
OPTION="--configuration json://configurationSG.json"

o2-analysis-track-propagation ${OPTION} |\
o2-analysis-timestamp ${OPTION} |\
o2-analysis-pid-tof-full ${OPTION} |\
o2-analysis-pid-tof-beta ${OPTION} |\
o2-analysis-pid-tof-base ${OPTION} |\
o2-analysis-pid-tpc-full ${OPTION} |\
o2-analysis-pid-tpc-base ${OPTION} |\
o2-analysis-ft0-corrected-table ${OPTION} |\
o2-analysis-event-selection ${OPTION} |\
o2-analysis-trackselection ${OPTION} |\
o2-analysis-multiplicity-table ${OPTION} |\
o2-analysis-ud-sgcand-producer ${OPTION} --aod-writer-json OutputDirectorSG.json --aod-file AO2D_pass4.root
```

Derived data tables

Contain the following parts:
- Collision information
- Track information (associated to each collision)
- ZDC Tables

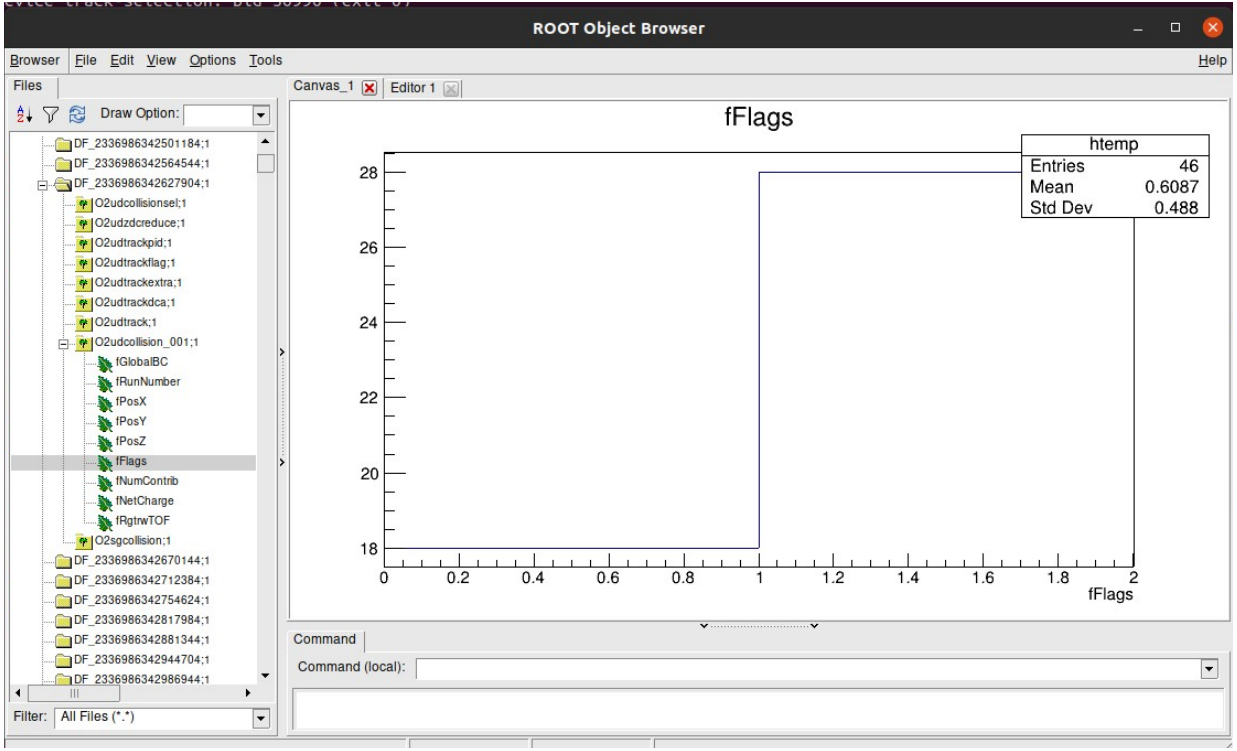
Optionally:
- Forward Tracks
- Monte Carlo Tables



Derived data tables NB!

UDCollision → UDCollision_001

in **pass2** we had a special **pass2_upc** reconstruction with special tracking parameters to increase tracking efficiency in low multiplicity events.



pass4 is just one reconstruction with UPC parameters used event-by-event based of the number of tracks.

Flag showing this is added to UDCollision table.

0 → standard setting

1 → UPC setting

UDCollisions Converter

O2Physics/PWGUD/TableProducer/Converters/UDCollisionsConverter.cxx

```
/// executable name o2-analysis-ud-collisions-converter

/// \author Sasha Bylinkin <alexandr.bylinkin@cern.ch>

#include "Framework/runDataProcessing.h"
#include "Framework/AnalysisTask.h"
#include "Framework/AnalysisDataModel.h"
#include "PWGUD/DataModel/UDTables.h"

using namespace o2;
using namespace o2::framework;

// Converts UDCollisions for version 000 to 001
struct UDCollisionsConverter {
  Produces<o2::aod::UDCollisions_001> udCollisions_001;

  void process(o2::aod::UDCollisions_000 const& collisions)
  {

    for (const auto& collision : collisions) {

      udCollisions_001(collision.globalBC(),
                       collision.runNumber(),
                       collision.posX(),
                       collision.posY(),
                       collision.posZ(),
                       0.0f, // dummy UPC reco flag, not available in version 000
                       collision.numContrib(),
                       collision.netCharge(),
                       collision.rgtrwTOF());
    }
  }
};

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

UD Collisions Converter is needed to convert old table from derived pass2_upc data into the new format.

Note the dummy reco flag.

It should be included as a dependency when you run your analysis task on the old derived data. *o2-analysis-ud-collisions-converter*.

Hyperloop:
Service Wagon UD/UD_collisions_converter

OutputDirector.json

```
{ "OutputDirector": {
  "debug_mode": true,
  "resfile": "AO2D_SG_debug1",
  "OutputDescriptors": [
    {
      "table": "AOD/SGCOLLISION/0"
    },
    {
      "table": "AOD/UDCOLLISION/1"
    },
    {
      "table": "AOD/UDTRACKEXTRA/0"
    },
    {
      "table": "AOD/UDTRACKPID/0"
    },
    {},
    {
      "table": "AOD/UDCOLLISIONSEL/0"
    },
    {
      "table": "AOD/UDTRACKDCA/0"
    },
    {
      "table": "AOD/UDTRACK/0"
    },
    {
      "table": "AOD/UDZDCREDUCE/0"
    },
    {
      "table": "AOD/UDTRACKFLAG/0"
    }
  ],
  "ntfmerge": 1
} }
```

Output filename.

Saved tables.
Note the version (1).

Hyperloop:
Include the versioned table in the derived data or you will get a warning!

Single_Gap_Producer

[ ]	AOD	UDTracksLabels	UDTRACKLABEL	0
[x]	AOD	SGCollisions	SGCOLLISION	0
[ ]	AOD	UDCollisions	UDCOLLISION	0
[x]	AOD	UDTracks	UDTRACK	0
[x]	AOD	UDTracksFlags	UDTRACKFLAG	0
[x]	AOD	UDTracksExtra	UDTRACKEXTRA	0
[x]	AOD	UDCollisionsSels	UDCOLLISIONSEL	0
[x]	AOD	UDTracksPID	UDTRACKPID	0
[x]	AOD	UDTracksDCA	UDTRACKDCA	0
[ ]	AOD	UDZdcs	UDZDC	0
[x]	AOD	UDZdcsReduced	UDZDCREDUCE	0
[ ]	AOD	UDFwdTracksExtra_001	UDFWDTRACKEXTRA	1
[ ]	AOD	UDMcCollisions	UDMCCOLLISIONS	0
[ ]	AOD	UDMcCollsLabels	UDMCCOLLSLABEL	0
[ ]	AOD	UDMcParticles	UDMCPARTICLES	0
[ ]	AOD	UDMcTrackLabels	UDMCTRACKLABEL	0
[x]	AOD	UDCollisions_001	UDCOLLISION	1

Save

Analysis Tools

- TrueGapFunction in https://github.com/AliceO2Group/O2Physics/blob/master/PWGUD/Core/SGSelector.h
- TrackSelector

  https://github.com/AliceO2Group/O2Physics/blob/master/PWGUD/Core/SGTrackSelector.h

TrueGap function

- SGSelector.trueGap(CC& collision, float fv0, float ft0a, float ft0c, float zdc_cut)
- Use on derived data to refine Gap selections
  - One can further restrict FIT thresholds and also add ZDC selections

    → Allows to distinguish further between 0n0n, 0nXn, Xn0n, XnXn classes, for example

TrueGap function

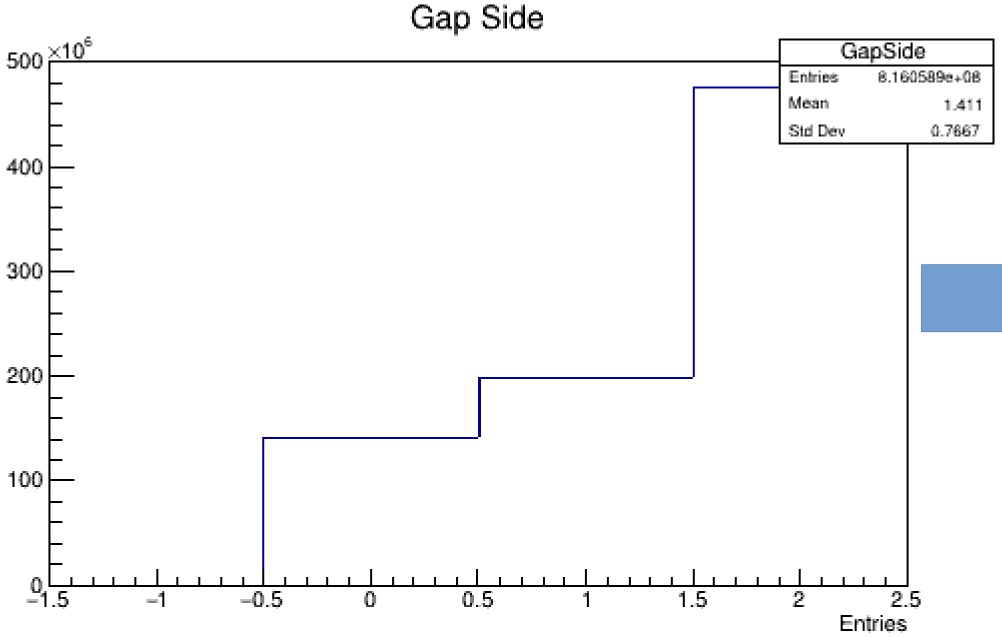
```
template <typename CC>
int trueGap(CC& collision, float fv0, float ft0a, float ft0c, float zdc_cut)
{
  float fit_cut[3] = {fv0, ft0a, ft0c};
  int gap = collision.gapSide();
  int true_gap = gap;
  float FV0A, FT0A, FT0C, ZNA, ZNC;
  FV0A = collision.totalFV0AmplitudeA();
  FT0A = collision.totalFT0AmplitudeA();
  FT0C = collision.totalFT0AmplitudeC();
  ZNA = collision.energyCommonZNA();
  ZNC = collision.energyCommonZNC();
  if (gap == 0) {
    if (FV0A > fit_cut[0] || FT0A > fit_cut[1] || ZNA > zdc_cut)
      true_gap = -1;
  } else if (gap == 1) {
    if (FT0C > fit_cut[2] || ZNC > zdc_cut)
      true_gap = -1;
  } else if (gap == 2) {
    if ((FV0A > fit_cut[0] || FT0A > fit_cut[1] || ZNA > zdc_cut) && (FT0C > fit_cut[2] || ZNC > zdc_cut))
      true_gap = -1;
    else if ((FV0A > fit_cut[0] || FT0A > fit_cut[1] || ZNA > zdc_cut) && (FT0C <= fit_cut[2] && ZNC <= zdc_cut))
      true_gap = 1;
    else if ((FV0A <= fit_cut[0] && FT0A <= fit_cut[1] && ZNA <= zdc_cut) && (FT0C > fit_cut[2] || ZNC > zdc_cut))
      true_gap = 0;
    else if (FV0A <= fit_cut[0] && FT0A <= fit_cut[1] && ZNA <= zdc_cut && FT0C <= fit_cut[2] && ZNC <= zdc_cut)
      true_gap = 2;
    else
      std::cout << "Something wrong with DG" << std::endl;
  }
  return true_gap;
}
```

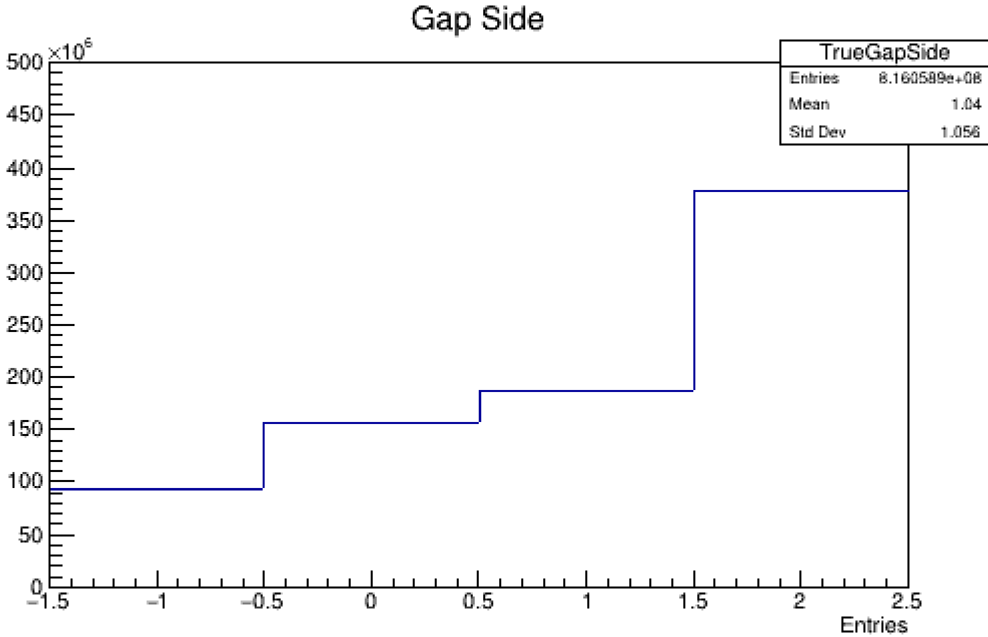
Compares the saved FIT and ZDC info vs new thresholds (should be more strict to have any effect)

And returns the integer value defining the new gap side
*see next slide.

TrueGap function

int truegapSide = sgSelector.trueGap(collision, FIT_cut[0], FIT_cut[1], FIT_cut[2], ZDC_cut);

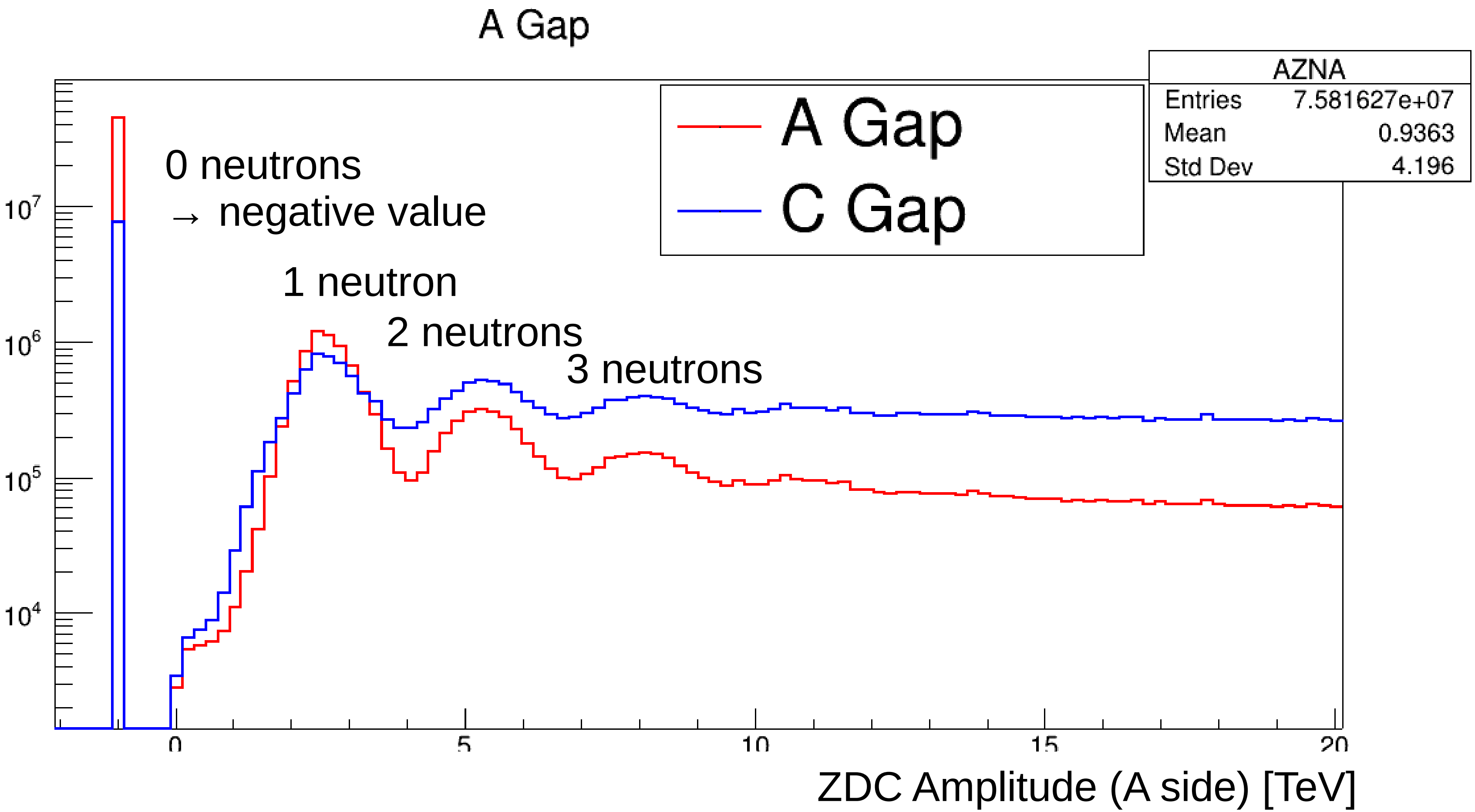




0 – A-side Gap
1 – C-side Gap
2 – Double Gap
-1 – No Gap

Changing the classification of events when analyzing derived data.

ZDC Amplitudes



No pedestal in ZDCs @Run 3

TrueGap function: ZDC Classes

int truegapSide**1** = sgSelector.trueGap(collision, FIT_cut[0], FIT_cut[1], FIT_cut[2], **1000**);
→ Refine selection with new FIT thresholds. **No ZDC cut**
int truegapSide**2** = sgSelector.trueGap(collision, FIT_cut[0], FIT_cut[1], FIT_cut[2], **0.1**);
→ Call trueGap function the second time now with ZDC cut. **Any ZDC signal**

TruegapSide1 == 2: UPC event.

truegapSide2 == 0: 0nXn	0 – A-side Gap
truegapSide2 == 1: Xn0n	1 – C-side Gap
truegapSide2 == 2: 0n0n	2 – Double Gap
truegapSide2 == -1: XnXn	-1 – No Gap

TrueGap function doesn't remove events from the data sample, so could be called multiple times from your analysis task.
→ One can call with several ZDC Cuts: i.e. 4, 7, 9 – to further distinguish 1n, 2n, 3n classes.

Track Selector

Task to simplify and standardize the track selection over all analyses.

Track selection parameters are set as configurables in the analysis task:

```
Configurable<float> PV_cut{"PV_cut", 0.0, "Use Only PV tracks"};
Configurable<float> dcaZ_cut{"dcaZ_cut", 2.0, "dcaZ cut"};
Configurable<float> dcaXY_cut{"dcaXY_cut", 2.0, "dcaXY cut (0 for Pt-function)"};
Configurable<float> tpcChi2_cut{"tpcChi2_cut", 4, "Max tpcChi2NCl"};
Configurable<float> tpcNClsFindable_cut{"tpcNClsFindable_cut", 70, "Min tpcNClsFindable"};
Configurable<float> itsChi2_cut{"itsChi2_cut", 36, "Max itsChi2NCl"};
Configurable<float> eta_cut{"eta_cut", 0.9, "Track Pseudorapidity"};
Configurable<float> pt_cut{"pt_cut", 0.1, "Track Pt"};
```

Then the pointer to the track is passed in the analysis task:

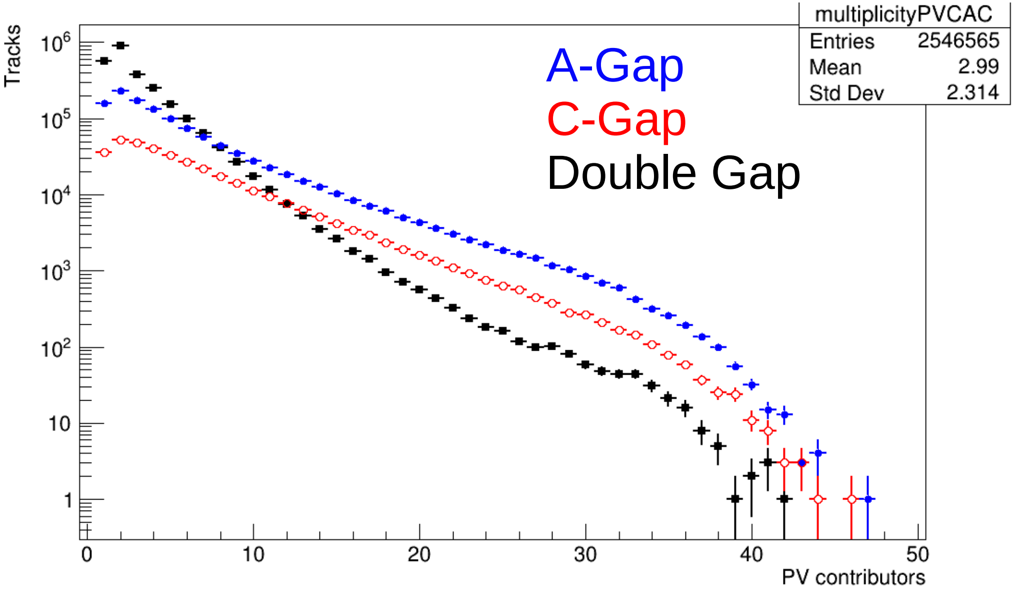
```
std::vector<float> parameters = {PV_cut, dcaZ_cut, dcaXY_cut, tpcChi2_cut,
                                 tpcNClsFindable_cut, itsChi2_cut, eta_cut, pt_cut};
```

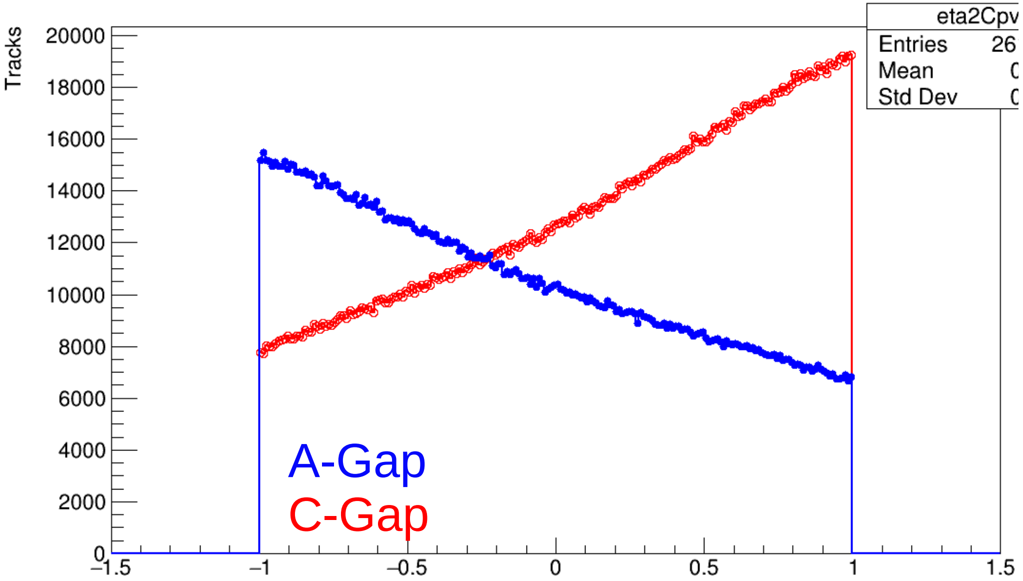
bool isGoodTrack *= trackselector(t0, parameters);*

Track pointer **Array of parameter values**

Backup

Multiplicity and Rapidity distributions

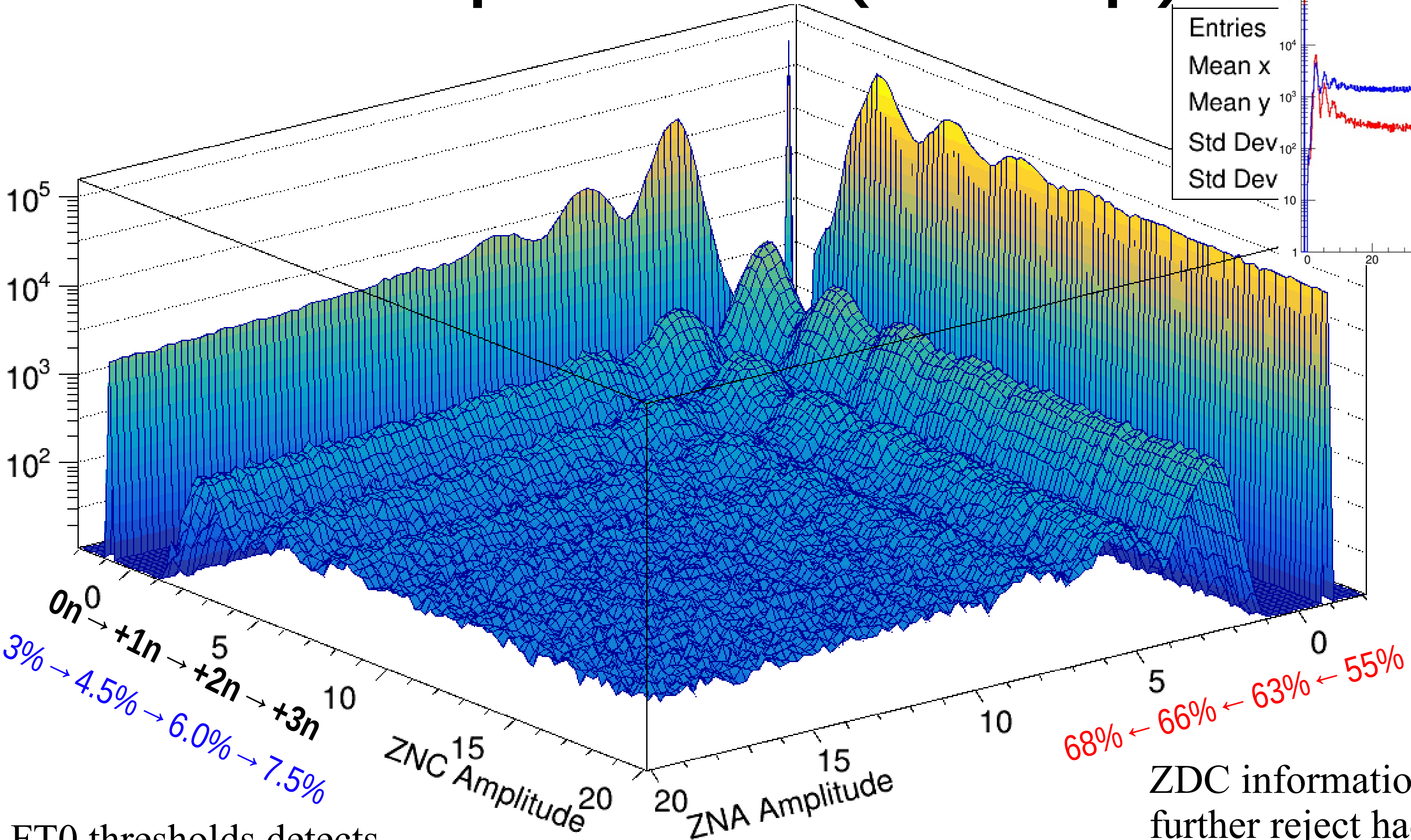


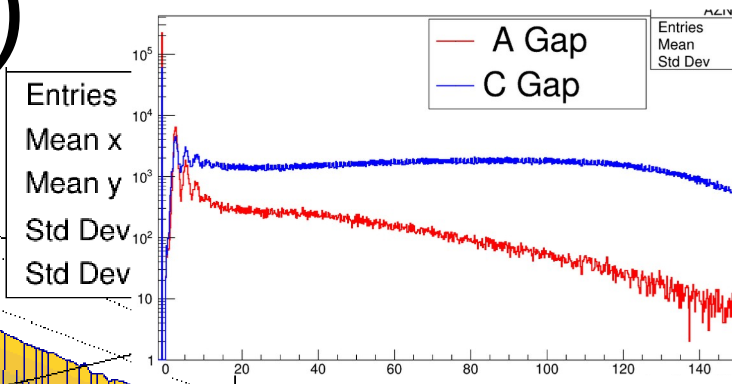


Multiplicities go to higher values with a visible difference between Single Gap and Double Gap events.

Nice asymmetric distributions!

ZDC Amplitudes (A-Gap)





FT0 thresholds detects active events rather well.

ZDC information can be used to further reject hadronic events.
On derived data!