Updates of a new analysis framework on SHOE

Giacomo Ubaldi
Dr. Roberto Zarrella

FOOT Software Meeting

18/06/2024

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

After Loop( )

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

After Loop( )

1

Create a map of **selection cuts**
for event and tracks (**reco** level)

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

After Loop( )

1 Create a map of **selection cuts** for event and tracks (**reco** level)

2 Create a map of **variable values** needed for cross section (**reco** level)

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

After Loop( )

1 Create a map of **selection cuts** for event and tracks (**reco** level)

2 Create a map of **variable values** needed for cross section (**reco** level)

3 Create a map of **selection cuts** and **variable values** (**MC truth** level)

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

After Loop( )

1 Create a map of **selection cuts** for event and tracks (**reco** level)

2 Create a map of **variable values** needed for cross section (**reco** level)

3 Create a map of **selection cuts** and **variable values** (**MC truth** level)

↓

4 Fill a **Flat Tree** with the previous maps for every event

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

1 Create a map of **selection cuts** for event and tracks (**reco** level)

2 Create a map of **variable values** needed for cross section (**reco** level)

3 Create a map of **selection cuts** and **variable values** (**MC truth** level)

4 Fill a **Flat Tree** with the previous maps for every event

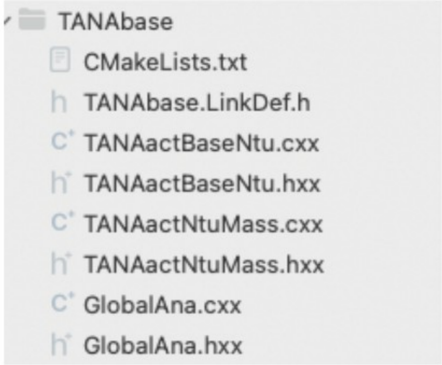
After Loop( )

4 Filter, process data to obtain a final **cross section**

New Classes

❑ Analysis folder:

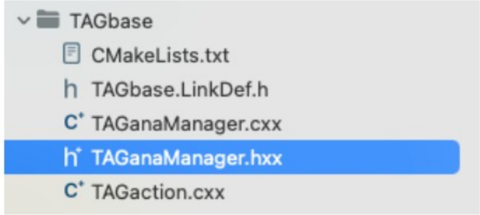
- Folder added:



TANAbase
CMakeLists.txt
TANAbase.LinkDef.h
TANAactBaseNtu.cxx
TANAactBaseNtu.hxx
TANAactNtuMass.cxx
TANAactNtuMass.hxx
GlobalAna.cxx
GlobalAna.hxx

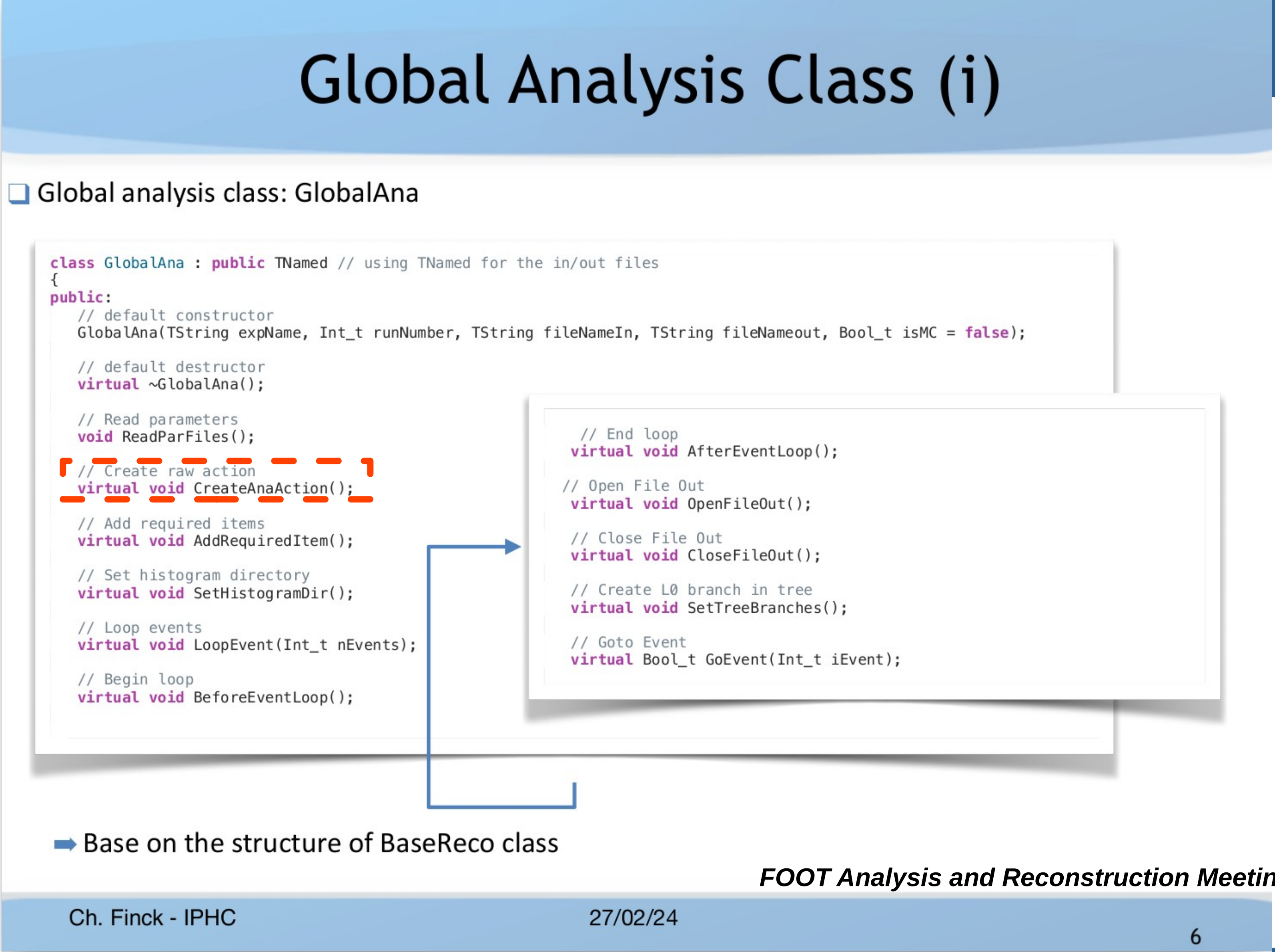
❑ Library folder:

- New analysis manager class:



TAGbase
CMakeLists.txt
TAGbase.LinkDef.h
TAGanaManager.cxx
TAGanaManager.hxx
TAGaction.cxx

- Base class for analysis: TANAactBaseNtu
- Example of analysis class: TANAactNtuMass
- Master class managing analysis: GlobalAna



Global Analysis Class (i)

❑ Global analysis class: GlobalAna

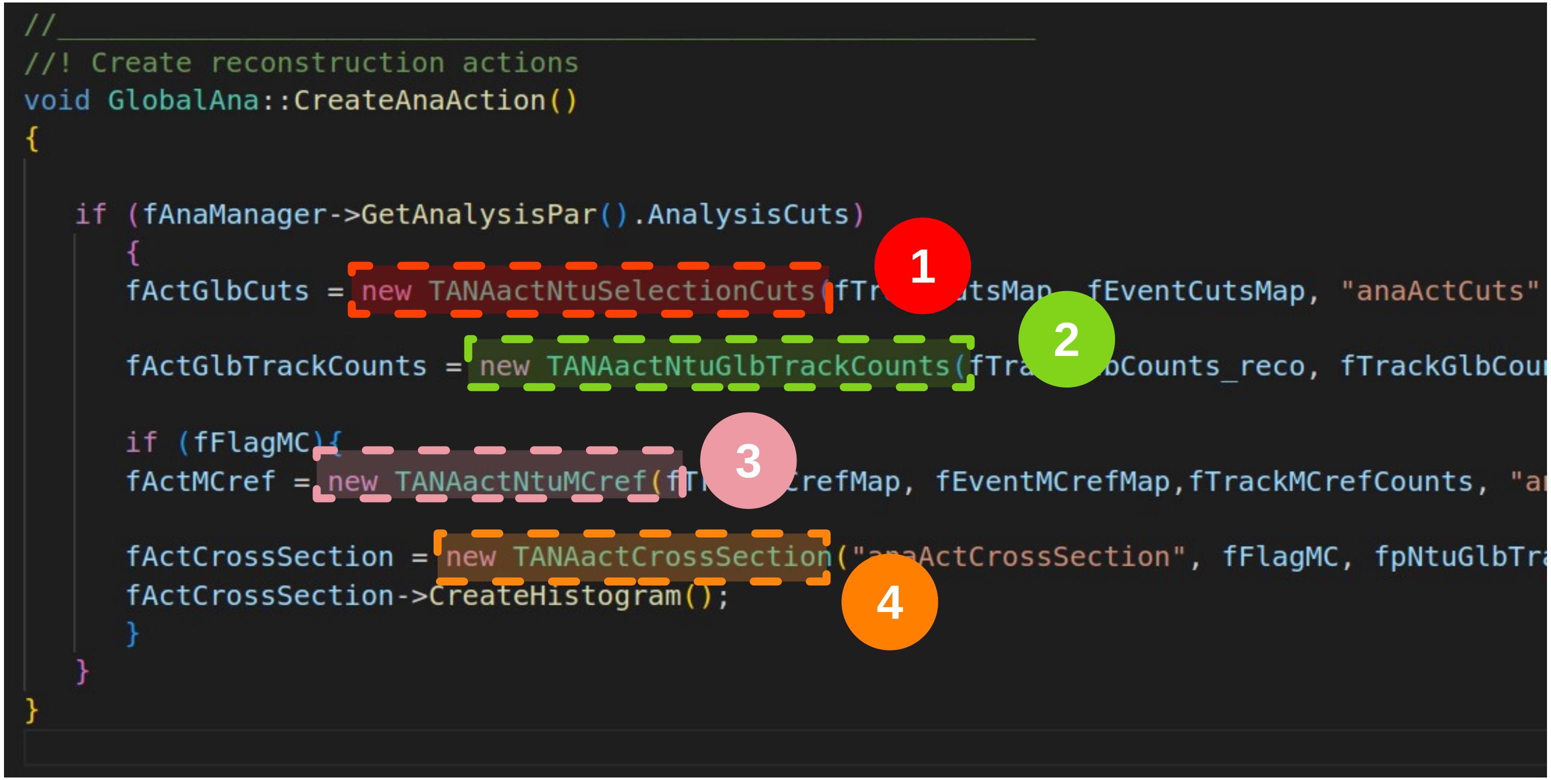
```
class GlobalAna : public TNamed // using TNamed for the in/out files
{
public:
   // default constructor
   GlobalAna(TString expName, Int_t runNumber, TString fileNameIn, TString fileNameout, Bool_t isMC = false);

   // default destructor
   virtual ~GlobalAna();

   // Read parameters
   void ReadParFiles();

   // Create raw action
   virtual void CreateAnaAction();

   // Add required items
   virtual void AddRequiredItem();

   // Set histogram directory
   virtual void SetHistogramDir();

   // Loop events
   virtual void LoopEvent(Int_t nEvents);

   // Begin loop
   virtual void BeforeEventLoop();
```

```
    // End loop
   virtual void AfterEventLoop();

  // Open File Out
   virtual void OpenFileOut();

   // Close File Out
   virtual void CloseFileOut();

   // Create L0 branch in tree
   virtual void SetTreeBranches();

   // Goto Event
   virtual Bool_t GoEvent(Int_t iEvent);
```

➡ Base on the structure of BaseReco class

Ch. Finck - IPHC 27/02/24 6

void GlobalAna::CreateAnaAction( )



```
//________________________________________________________________
//! Create reconstruction actions
void GlobalAna::CreateAnaAction()
{

   if (fAnaManager->GetAnalysisPar().AnalysisCuts)
      {
      fActGlbCuts = new TANAactNtuSelectionCuts(fTr        tsMap, fEventCutsMap, "anaActCuts"

      fActGlbTrackCounts = new TANAactNtuGlbTrackCounts(fTra        bCounts_reco, fTrackGlbCou

      if (fFlagMC){
      fActMCref = new TANAactNtuMCref(fT        crefMap, fEventMCrefMap,fTrackMCrefCounts, "a

      fActCrossSection = new TANAactCrossSection("a      aActCrossSection", fFlagMC, fpNtuGlbTra
      fActCrossSection->CreateHistogram();
      }
   }
}
```

Selection Cuts

1

TANAactNtuSelectionCuts

In Loop( ): 🔄

For every reconstructed track, two cut maps are associated:

- *fEventCutsMap* for element-wise cuts
- *fTrackCutsMap* for track-wise cuts

Selection Cuts

1

TANAactNtuSelectionCuts

In Loop( ): 🔄

For every reconstructed track, two cut maps are associated:

- *fEventCutsMap* for element-wise cuts
- *fTrackCutsMap* for track-wise cuts

- for every cut, a key of the map is generated
- an int value of **0,1** (or others if exception) is associated to each key

Selection Cuts

1

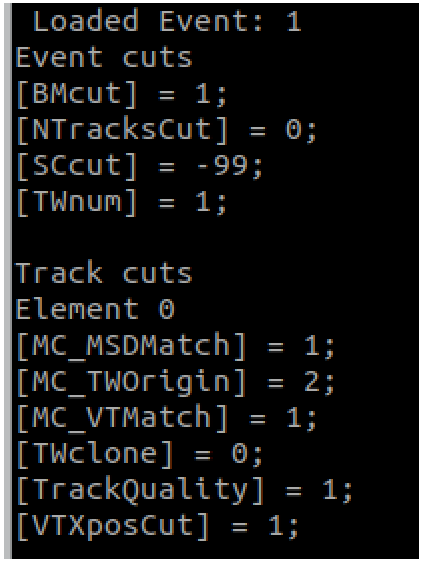
TANAactNtuSelectionCuts

In Loop( ):

For every reconstructed track, two cut maps are associated:

- *fEventCutsMap* for element-wise cuts
- *fTrackCutsMap* for track-wise cuts

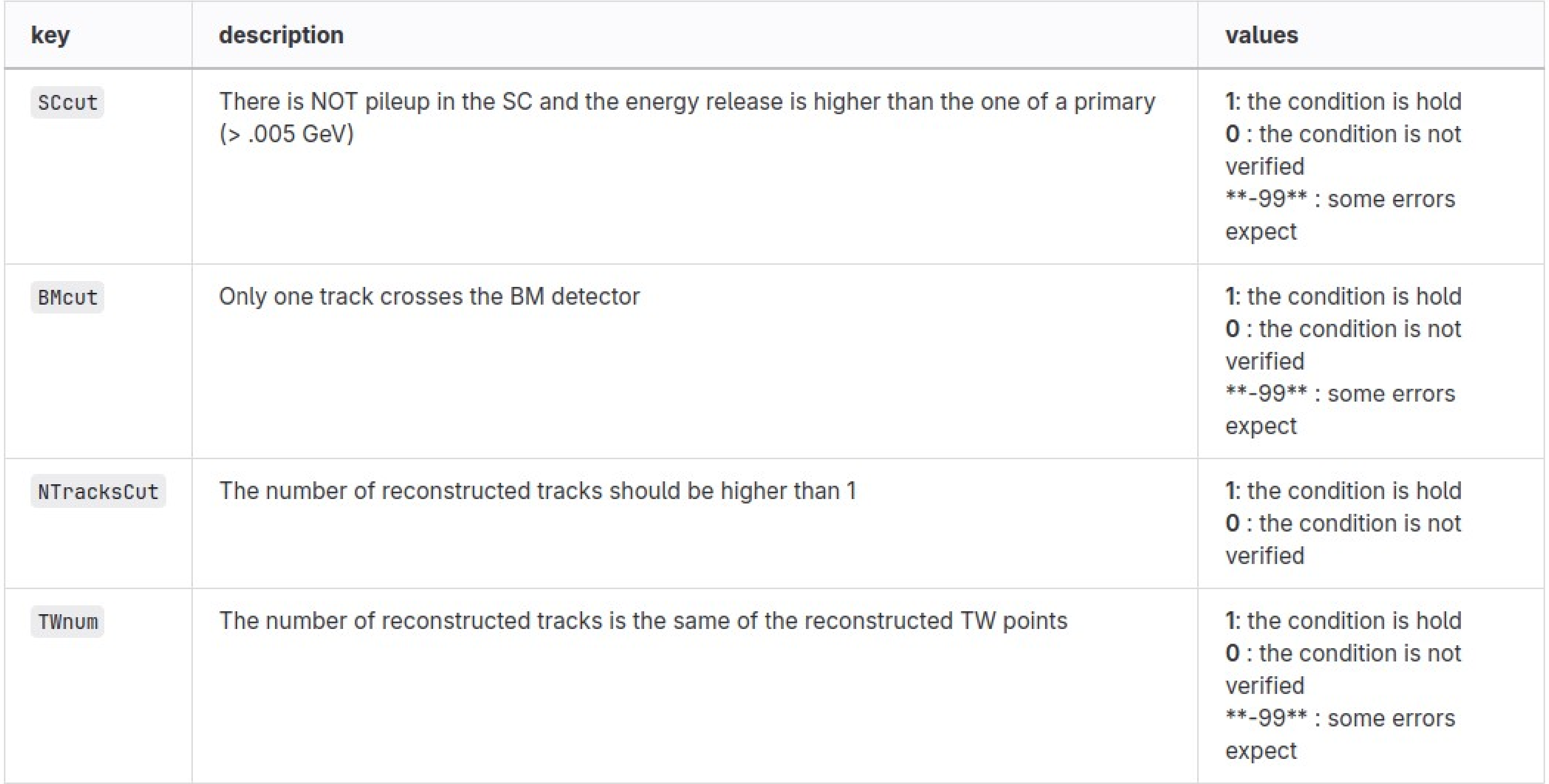
- for every cut, a key of the map is generated
- an int value of **0,1** (or others if exception) is associated to each key



```
 Loaded Event: 1
Event cuts
[BMcut] = 1;
[NTracksCut] = 0;
[SCcut] = -99;
[TWnum] = 1;

Track cuts
Element 0
[MC_MSDMatch] = 1;
[MC_TWOrigin] = 2;
[MC_VTMatch] = 1;
[TWclone] = 0;
[TrackQuality] = 1;
[VTXposCut] = 1;
```

Selection Cuts, events

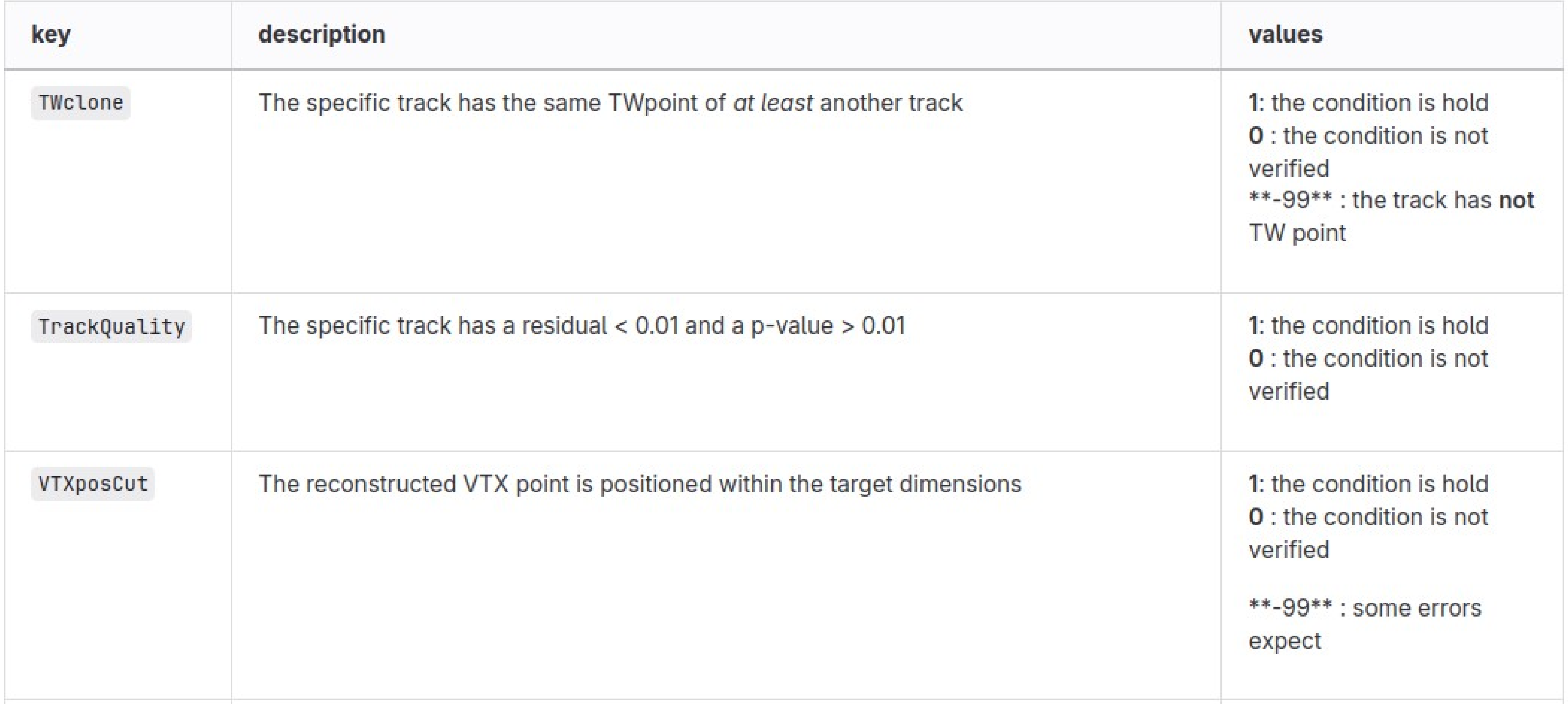
https://baltig.infn.it/asarti/shoe/-/wikis/Analysis-Cuts



key	description	values
`SCcut`	There is NOT pileup in the SC and the energy release is higher than the one of a primary (> .005 GeV)	**1**: the condition is hold **0** : the condition is not verified **-99** : some errors expect
`BMcut`	Only one track crosses the BM detector	**1**: the condition is hold **0** : the condition is not verified **-99** : some errors expect
`NTracksCut`	The number of reconstructed tracks should be higher than 1	**1**: the condition is hold **0** : the condition is not verified
`TWnum`	The number of reconstructed tracks is the same of the reconstructed TW points	**1**: the condition is hold **0** : the condition is not verified **-99** : some errors expect

Selection Cuts, tracks

https://baltig.infn.it/asarti/shoe/-/wikis/Analysis-Cuts



key	description	values
`TWclone`	The specific track has the same TWpoint of *at least* another track	**1**: the condition is hold **0** : the condition is not verified **-99** : the track has **not** TW point
`TrackQuality`	The specific track has a residual < 0.01 and a p-value > 0.01	**1**: the condition is hold **0** : the condition is not verified
`VTXposCut`	The reconstructed VTX point is positioned within the target dimensions	**1**: the condition is hold **0** : the condition is not verified **-99** : some errors expect

Reco quantities

2

TANAactNtuGlbTrackCounts

In Loop( ): 🔄

The idea is to create a map for every track, in which the main variables (the one for cross sections) are inserted

- *fTrackGlbCounts_reco* for **reconstructed** values of variables
- *fTrackGlbCounts_MC* for **true** values of variables

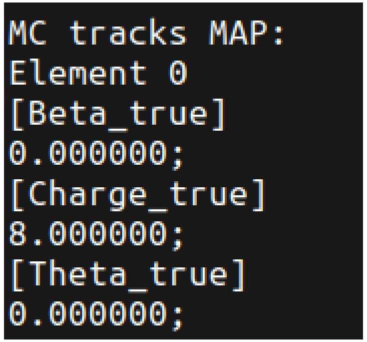
Reco quantities

2 TANAactNtuGlbTrackCounts

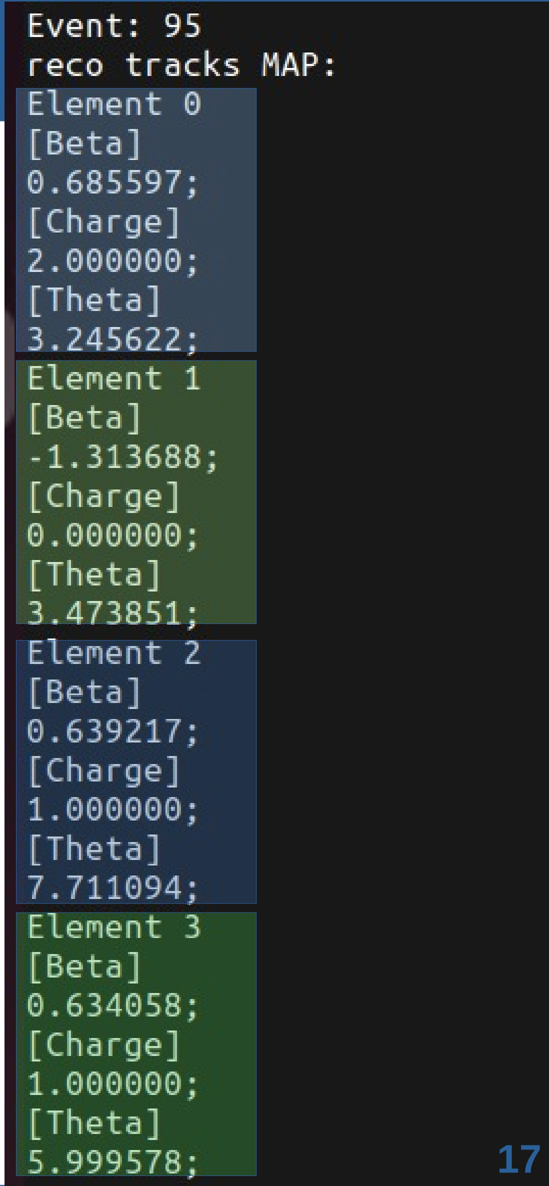
In Loop( ):

The idea is to create a map for every track, in which the main variables (the one for cross sections) are inserted

- *fTrackGlbCounts_reco* for **reconstructed** values of variables
- *fTrackGlbCounts_MC* for **true** values of variables



```
MC tracks MAP:
Element 0
[Beta_true]
0.000000;
[Charge_true]
8.000000;
[Theta_true]
0.000000;
```



```
Event: 95
reco tracks MAP:
Element 0
[Beta]
0.685597;
[Charge]
2.000000;
[Theta]
3.245622;
Element 1
[Beta]
-1.313688;
[Charge]
0.000000;
[Theta]
3.473851;
Element 2
[Beta]
0.639217;
[Charge]
1.000000;
[Theta]
7.711094;
Element 3
[Beta]
0.634058;
[Charge]
1.000000;
[Theta]
5.999578;
```

MC reference Cuts & quantities

3

TANAactNtuMCref

In Loop( ):

For every TAMCParticles, three cut maps are associated:

- *fEventMCrefMap* for **element**-wise MC cuts
- *fTrackMCrefMap* for **track**-wise MC cuts
- *fTrackMCrefCounts* for **true values** of variables

- for every cut, a key of the map is generated
- an int value of **0,1** (or others if exception) is associated to each key

MC reference Cuts & quantities

3

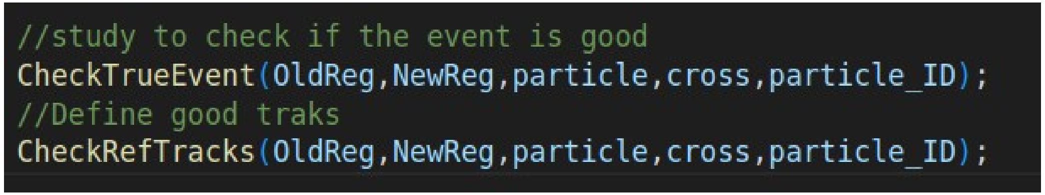
TANAactNtuMCref

In Loop( ):

For every TAMCParticles, three cut maps are associated:

- *fEventMCrefMap* for **element**-wise MC cuts
- *fTrackMCrefMap* for **track**-wise MC cuts
- *fTrackMCrefCounts* for **true values** of variables

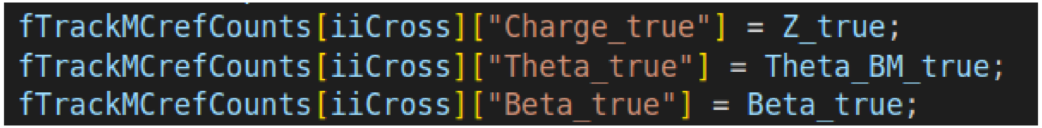
- for every cut, a key of the map is generated
- an int value of **0,1** (or others if exception) is associated to each key



```
//study to check if the event is good
CheckTrueEvent(OldReg,NewReg,particle,cross,particle_ID);
//Define good traks
CheckRefTracks(OldReg,NewReg,particle,cross,particle_ID);
```

// primary beams
   if it crosses the TG entering

// the fragment exits the TG and reaches the TW



```
fTrackMCrefCounts[iiCross]["Charge_true"] = Z_true;
fTrackMCrefCounts[iiCross]["Theta_true"] = Theta_BM_true;
fTrackMCrefCounts[iiCross]["Beta_true"] = Beta_true;
```

Toward Cross Section Measurements, 1

4

TANAactCrossSection

In Loop( ): 🔄

Two **TTree** are filled with the retrieved quantities for **all the tracks**:

- *aTree* for **reco**-wise tracks
- *aTreeMC* for **MC_truth**-wise MC cuts

Toward Cross Section Measurements, 1

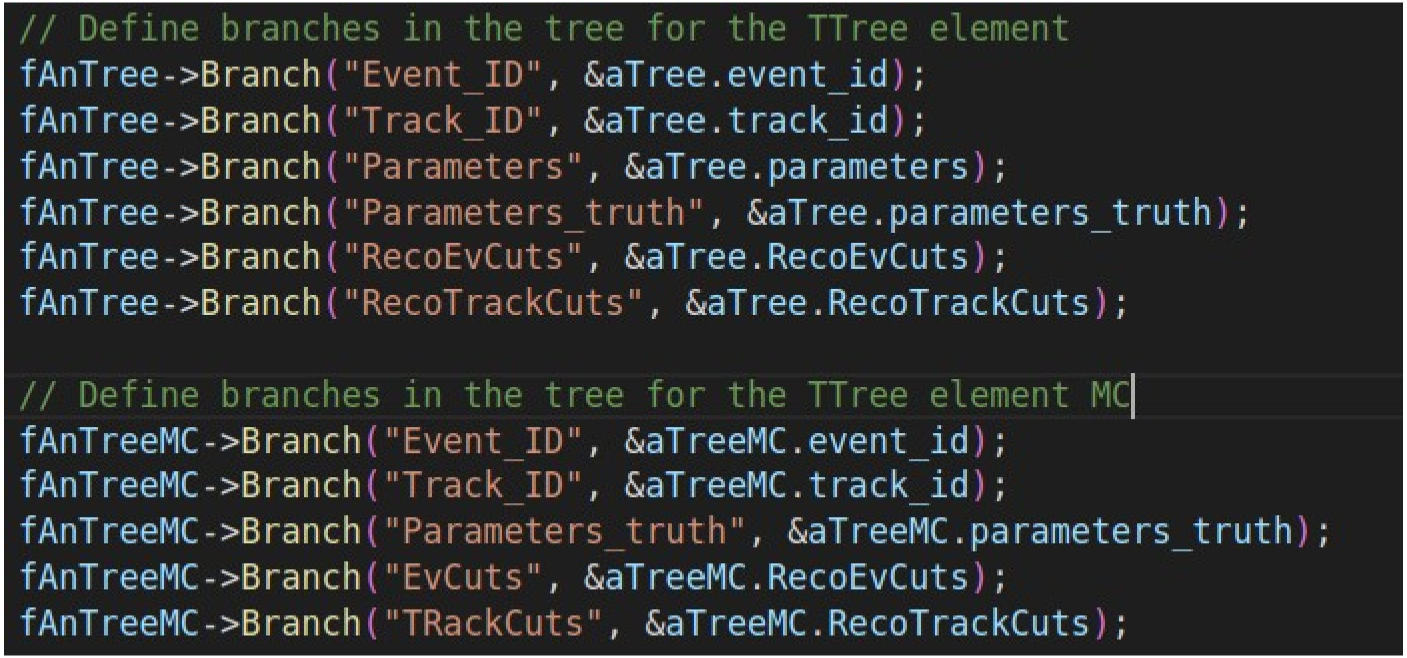
4

TANAactCrossSection

In Loop( ):

Two **TTree** are filled with the retrieved quantities for **all the tracks**:

- *aTree* for **reco**-wise tracks
- *aTreeMC* for **MC_truth**-wise MC cuts



```
// Define branches in the tree for the TTree element
fAnTree->Branch("Event_ID", &aTree.event_id);
fAnTree->Branch("Track_ID", &aTree.track_id);
fAnTree->Branch("Parameters", &aTree.parameters);
fAnTree->Branch("Parameters_truth", &aTree.parameters_truth);
fAnTree->Branch("RecoEvCuts", &aTree.RecoEvCuts);
fAnTree->Branch("RecoTrackCuts", &aTree.RecoTrackCuts);

// Define branches in the tree for the TTree element MC
fAnTreeMC->Branch("Event_ID", &aTreeMC.event_id);
fAnTreeMC->Branch("Track_ID", &aTreeMC.track_id);
fAnTreeMC->Branch("Parameters_truth", &aTreeMC.parameters_truth);
fAnTreeMC->Branch("EvCuts", &aTreeMC.RecoEvCuts);
fAnTreeMC->Branch("TRackCuts", &aTreeMC.RecoTrackCuts);
```

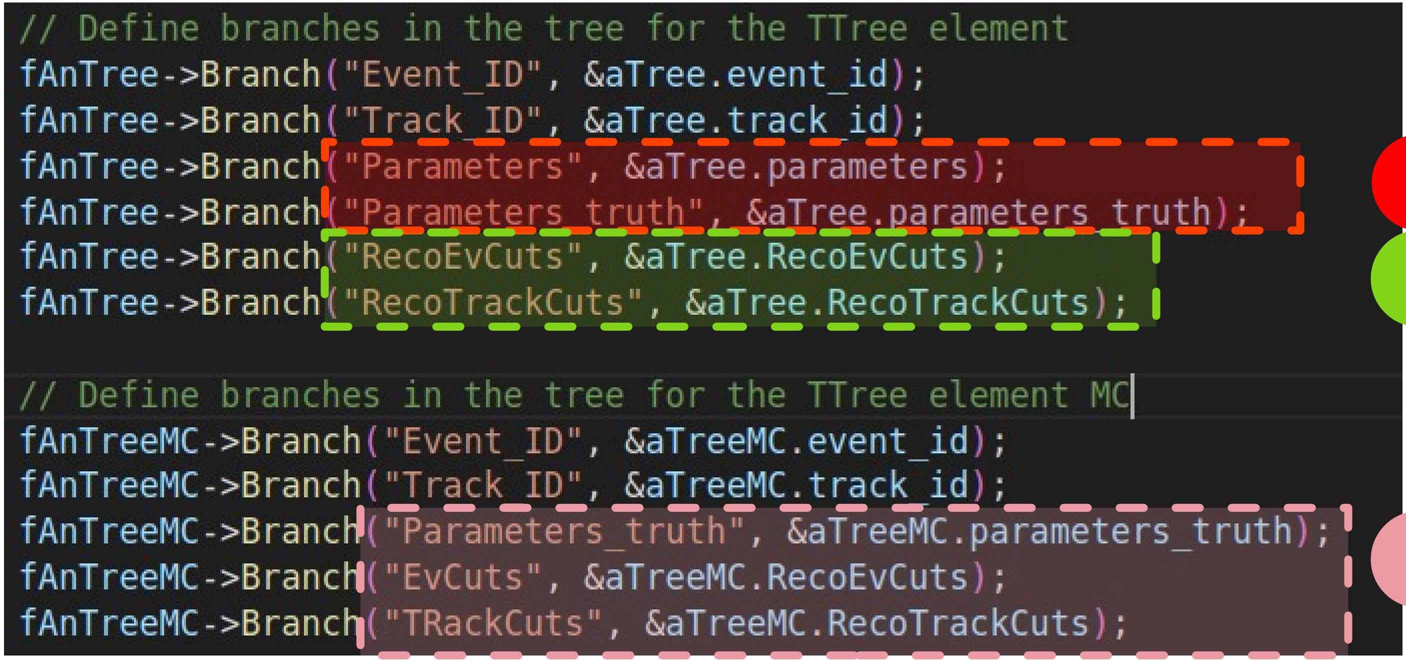
Toward Cross Section Measurements, 1

4 **TANAactCrossSection**

In Loop( ):

Two **TTree** are filled with the retrieved quantities for **all the tracks**:

- *aTree* for **reco**-wise tracks
- *aTreeMC* for **MC_truth**-wise MC cuts



```
// Define branches in the tree for the TTree element
fAnTree->Branch("Event_ID", &aTree.event_id);
fAnTree->Branch("Track_ID", &aTree.track_id);
fAnTree->Branch("Parameters", &aTree.parameters);
fAnTree->Branch("Parameters_truth", &aTree.parameters_truth);
fAnTree->Branch("RecoEvCuts", &aTree.RecoEvCuts);
fAnTree->Branch("RecoTrackCuts", &aTree.RecoTrackCuts);

// Define branches in the tree for the TTree element MC
fAnTreeMC->Branch("Event_ID", &aTreeMC.event_id);
fAnTreeMC->Branch("Track_ID", &aTreeMC.track_id);
fAnTreeMC->Branch("Parameters_truth", &aTreeMC.parameters_truth);
fAnTreeMC->Branch("EvCuts", &aTreeMC.RecoEvCuts);
fAnTreeMC->Branch("TRackCuts", &aTreeMC.RecoTrackCuts);
```

Maps from

1 TANAactNtuSelectionCuts

2 TANAactNtuGlbTrackCounts

3 TANAactNtuMCref

Analysis workflow

- updates on brach: Ubaldi_temp
- **Action** based structure

Event Loop( )

1 Create a map of **selection cuts** for event and tracks (**reco** level) ✓

2 Create a map of **variable values** needed for cross section (**reco** level) ✓

3 Create a map of **selection cuts** and **variable values** (**MC truth** level) ✓

↓

4 Fill a **Flat Tree** with the previous maps for every event ✓

→

After Loop( )

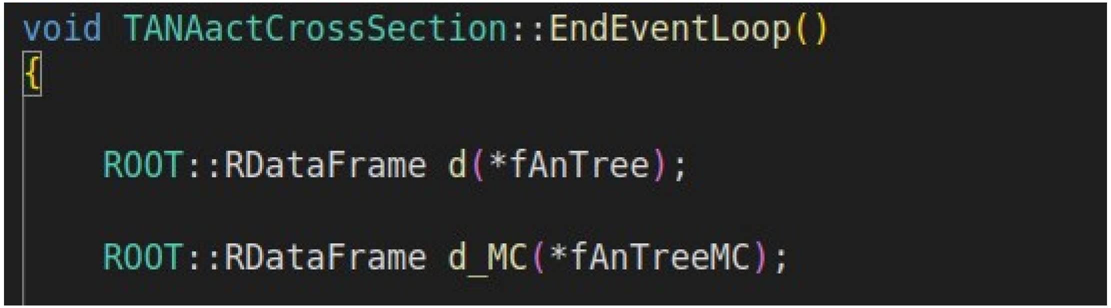
4 Filter, process data to obtain a final **cross section**

Toward Cross Section Measurements, 2

4

TANAactCrossSection

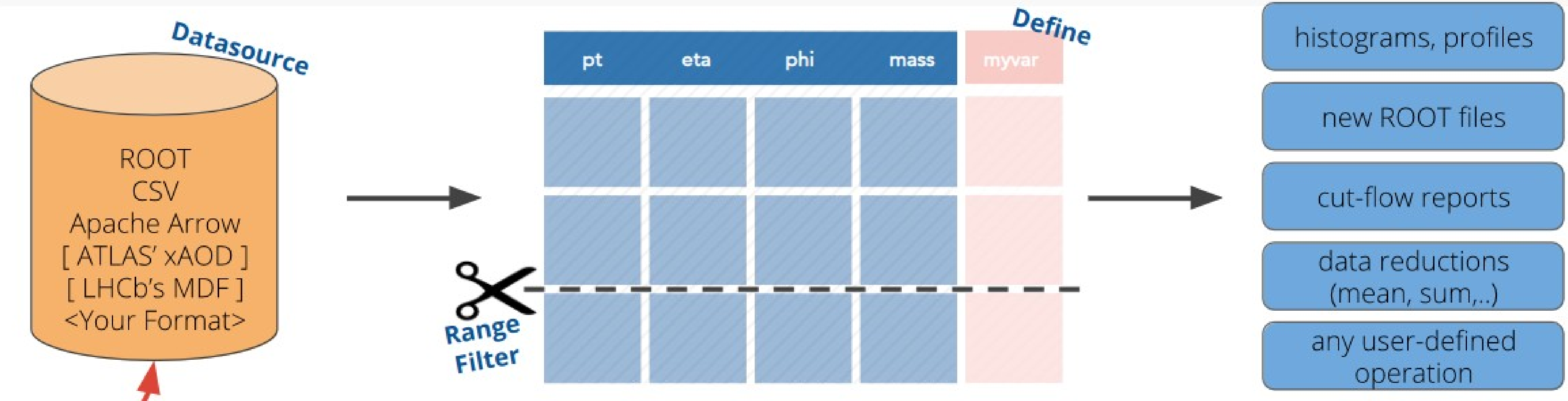
After Loop( ):
Let’s handle the TTree using the new ROOT class **ROOT::RDataFrame**



```
void TANAactCrossSection::EndEventLoop()
{

    ROOT::RDataFrame d(*fAnTree);

    ROOT::RDataFrame d_MC(*fAnTreeMC);
```

… but what is RDataFrame?

ROOT Declarative Analysis: RDataFrame



Datasource

ROOT
CSV
Apache Arrow
[ATLAS' xAOD]
[LHCb's MDF]
<Your Format>

pt | eta | phi | mass | myvar

Define

Range
Filter

- histograms, profiles
- new ROOT files
- cut-flow reports
- data reductions (mean, sum,..)
- any user-defined operation

Customisation point,
public interface!

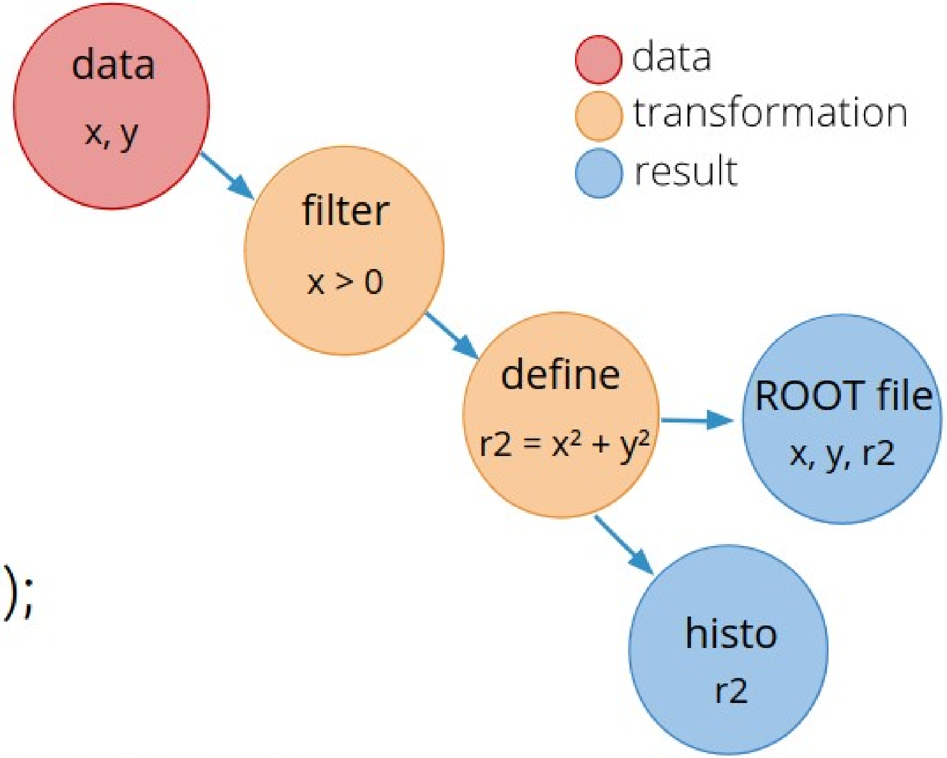
Goals:

- ➔ Be the **fastest** way to manipulate HEP data
- ➔ Be the **go-to ROOT analysis interface** from laptop to cluster
- ➔ Consistent interfaces in **Python and C++**
- ➔ Top notch documentation and examples

https://indico.cern.ch/event/759388/contributions/3356304/attachments/1815599/2968077/RDF__HSF_JeffersonLab.pdf

Analyses as computation graphs

```
ROOT::RDataFrame df(dataset);
auto df2 = df.Filter("x > 0")
             .Define("r2", "x*x + y*y");
auto rHist = df2.Histo1D("r2");
df2.Snapshot("newtree", "newfile.root");
```

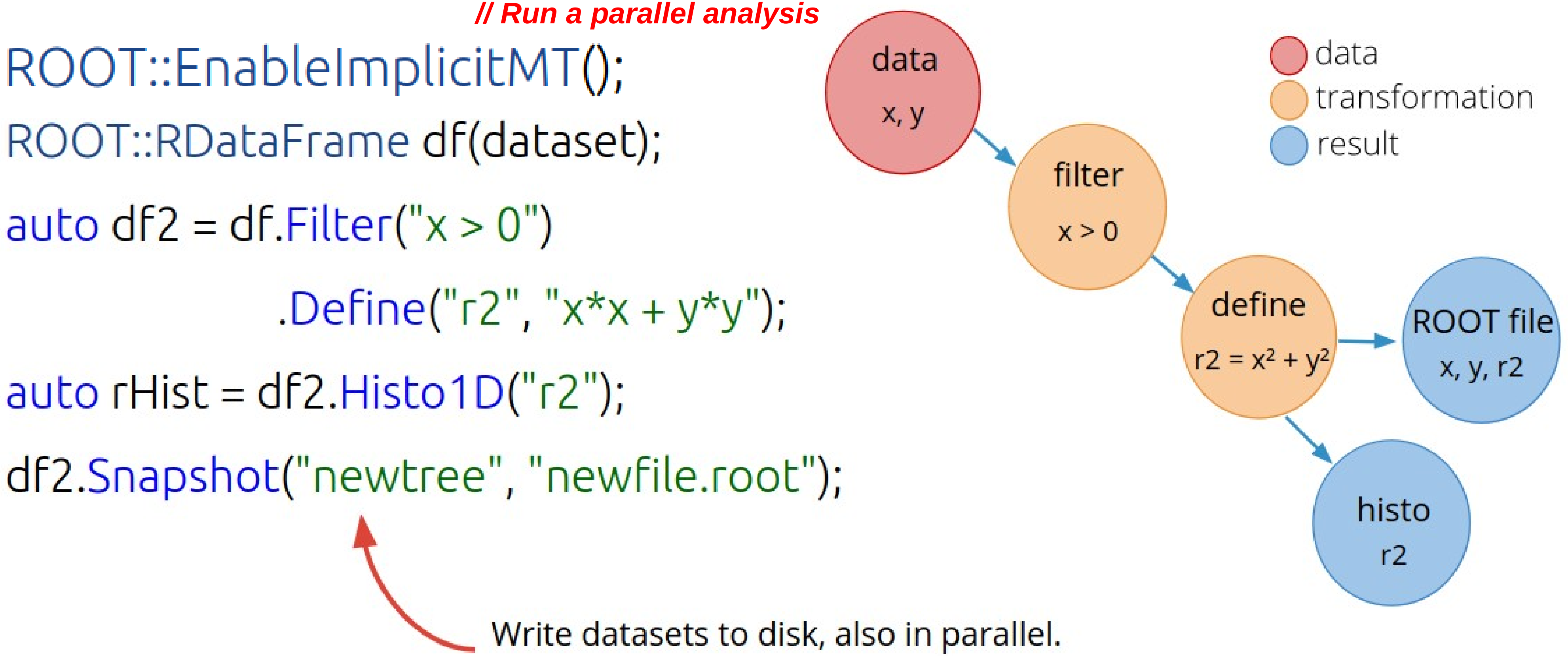


Write datasets to disk, also in parallel.

https://indico.cern.ch/event/759388/contributions/3356304/attachments/1815599/2968077/RDF__HSF_JeffersonLab.pdf

Analyses as computation graphs

// Run a parallel analysis



```
ROOT::EnableImplicitMT();
ROOT::RDataFrame df(dataset);
auto df2 = df.Filter("x > 0")
             .Define("r2", "x*x + y*y");
auto rHist = df2.Histo1D("r2");
df2.Snapshot("newtree", "newfile.root");
```

Write datasets to disk, also in parallel.

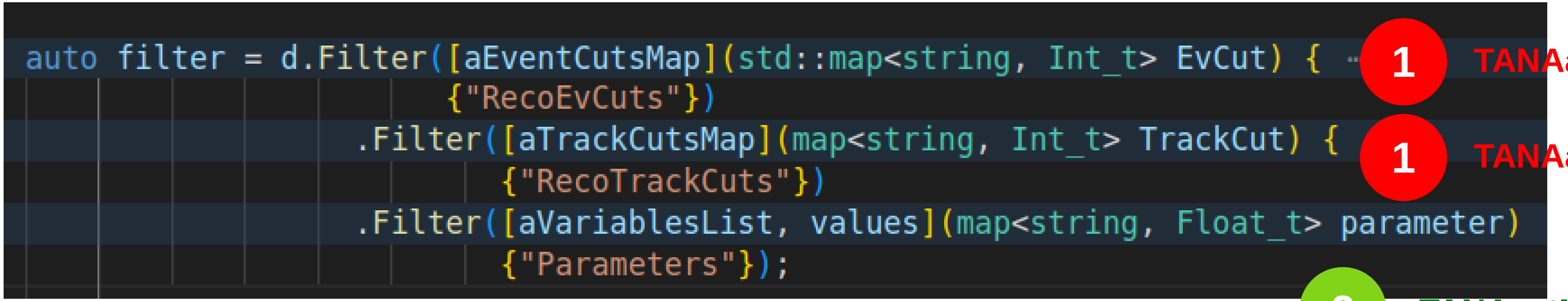
https://indico.cern.ch/event/759388/contributions/3356304/attachments/1815599/2968077/RDF__HSF_JeffersonLab.pdf

Yields via RDataFrame, 1

4 **TANAactCrossSection**

After Loop( ):

Create histograms of (differential) **yields** of a specific variable, according to cuts

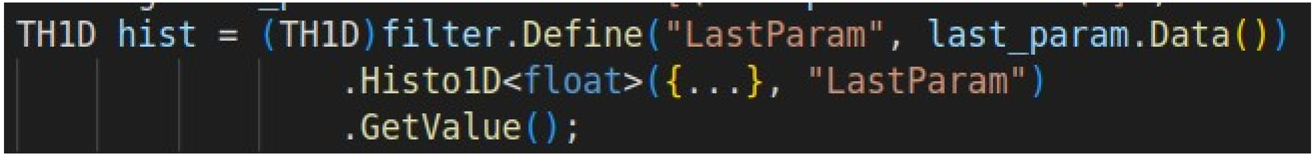


```
auto filter = d.Filter([aEventCutsMap](std::map<string, Int_t> EvCut) { ⋯
                  {"RecoEvCuts"})
              .Filter([aTrackCutsMap](map<string, Int_t> TrackCut) {
                    {"RecoTrackCuts"})
              .Filter([aVariablesList, values](map<string, Float_t> parameter)
                    {"Parameters"});
```

1 TANAactNtuSelectionCuts

1 TANAactNtuSelectionCuts

2 TANAactNtuGlbTrackCounts



```
TH1D hist = (TH1D)filter.Define("LastParam", last_param.Data())
                .Histo1D<float>({...}, "LastParam")
                .GetValue();
```

→ **Create histogram as output**

Yields via RDataFrame, 2

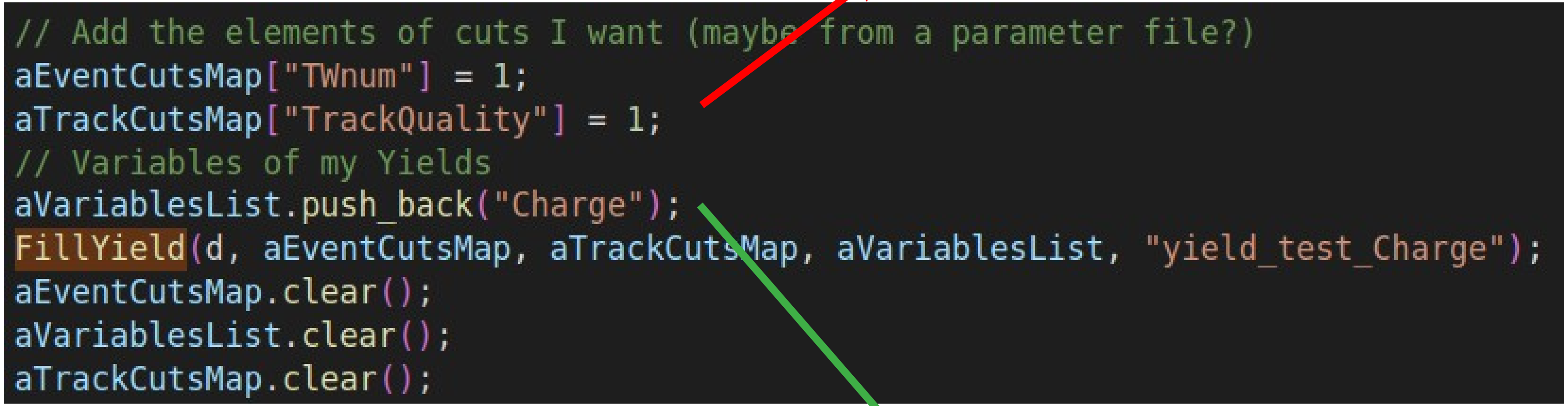
IMPORTANT

4 **TANAactCrossSection**

After Loop( ):

Create histograms of (differential) **yields** of a specific variable, according to cuts

same keys from 1 **TANAactNtuSelectionCuts**



```
// Add the elements of cuts I want (maybe from a parameter file?)
aEventCutsMap["TWnum"] = 1;
aTrackCutsMap["TrackQuality"] = 1;
// Variables of my Yields
aVariablesList.push_back("Charge");
FillYield(d, aEventCutsMap, aTrackCutsMap, aVariablesList, "yield_test_Charge");
aEventCutsMap.clear();
aVariablesList.clear();
aTrackCutsMap.clear();
```

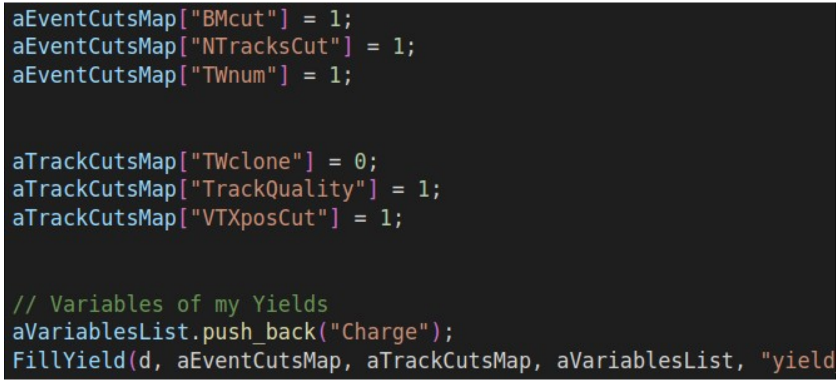
same keys from 2 **TANAactNtuGlbTrackCounts**

Examples

IMPORTANT

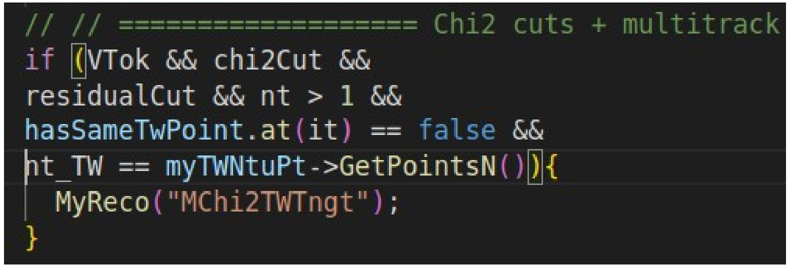
Analysis comparison, reco level

new analysis

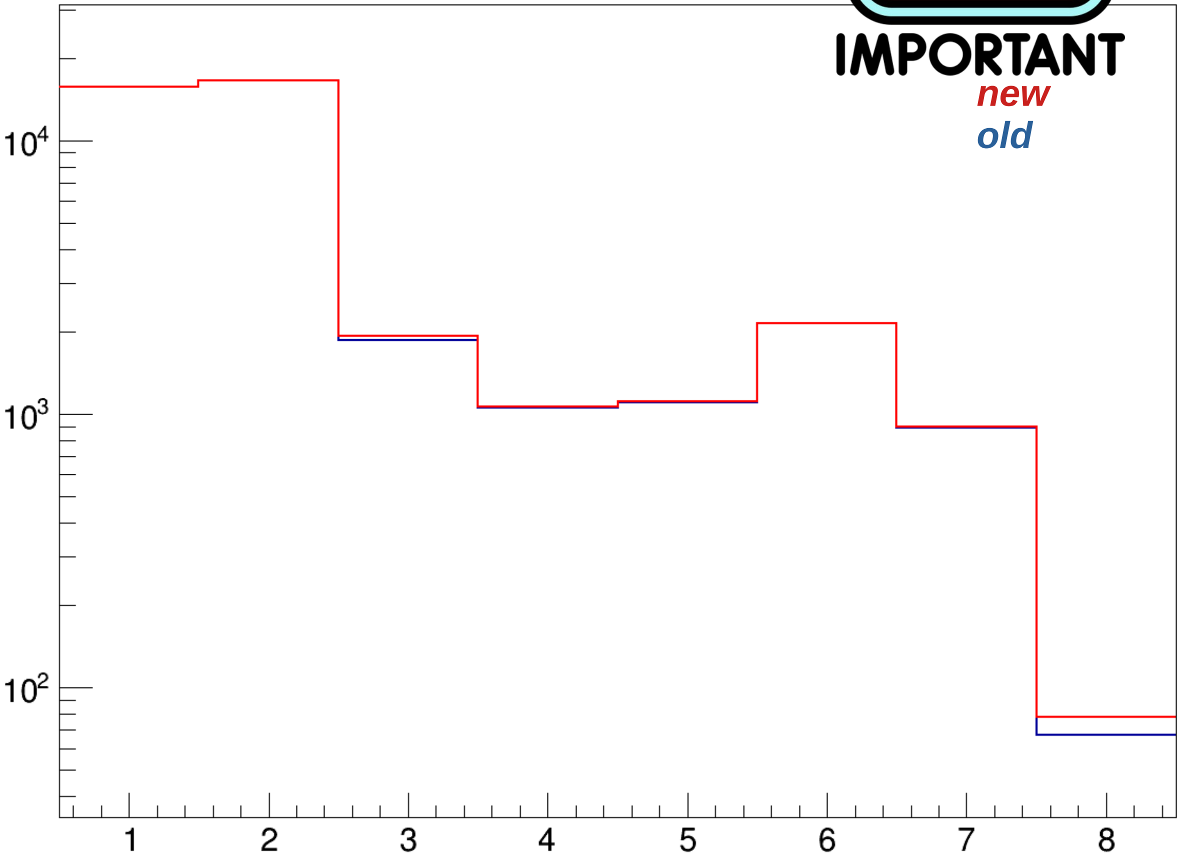


```
aEventCutsMap["BMcut"] = 1;
aEventCutsMap["NTracksCut"] = 1;
aEventCutsMap["TWnum"] = 1;


aTrackCutsMap["TWclone"] = 0;
aTrackCutsMap["TrackQuality"] = 1;
aTrackCutsMap["VTXposCut"] = 1;


// Variables of my Yields
aVariablesList.push_back("Charge");
FillYield(d, aEventCutsMap, aTrackCutsMap, aVariablesList, "yield
```

old analysis



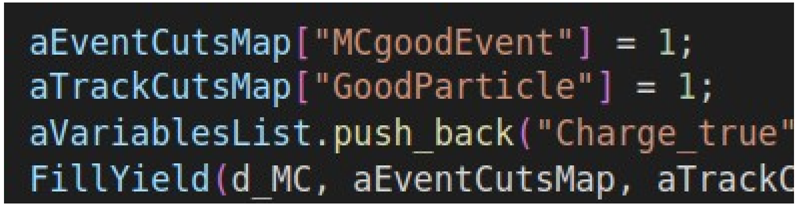
```
 // // ================== Chi2 cuts + multitrack
 if (VTok && chi2Cut &&
 residualCut && nt > 1 &&
 hasSameTwPoint.at(it) == false &&
 nt_TW == myTWNtuPt->GetPointsN()){
   MyReco("MChi2TWTngt");
 }
```



Examples

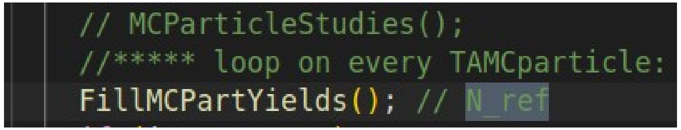
Analysis comparison, MC level

new analysis

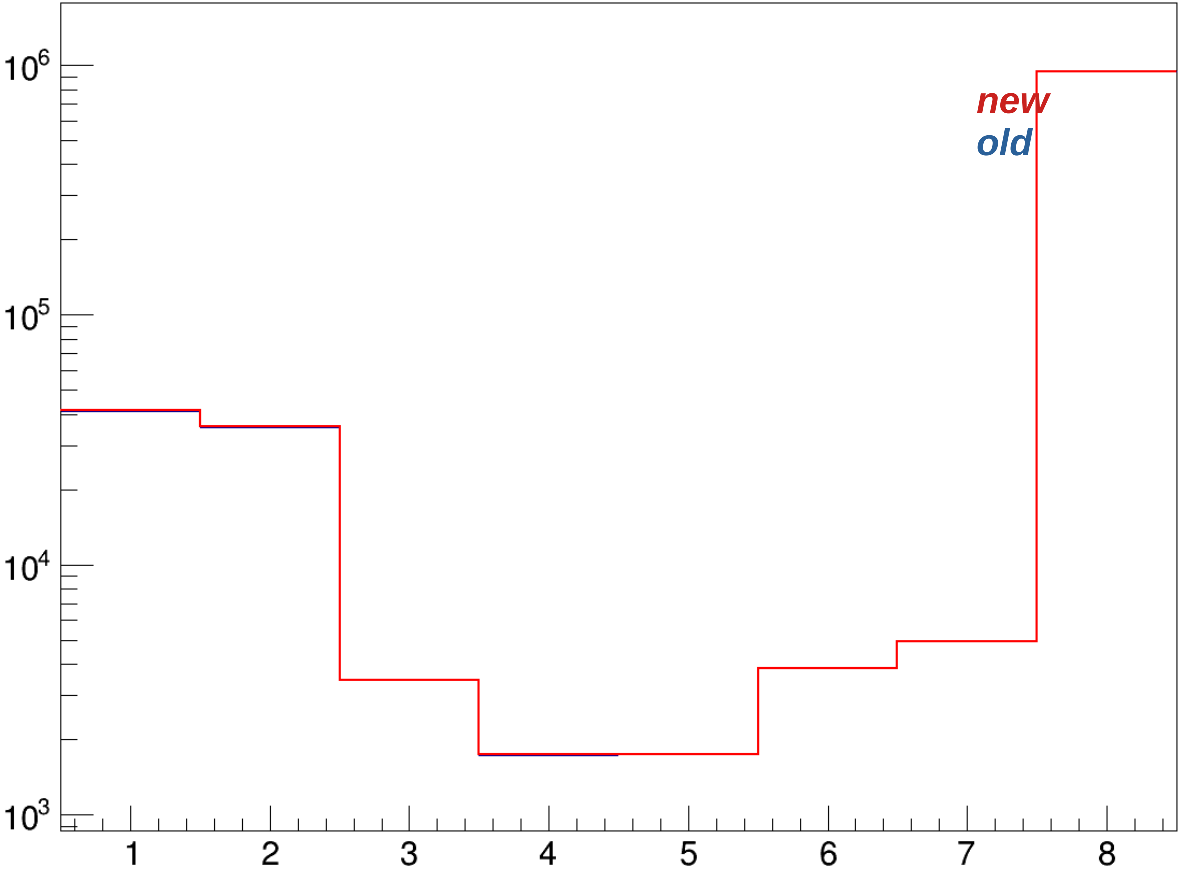


```
aEventCutsMap["MCgoodEvent"] = 1;
aTrackCutsMap["GoodParticle"] = 1;
aVariablesList.push_back("Charge_true"
FillYield(d_MC, aEventCutsMap, aTrackC
```

old analysis

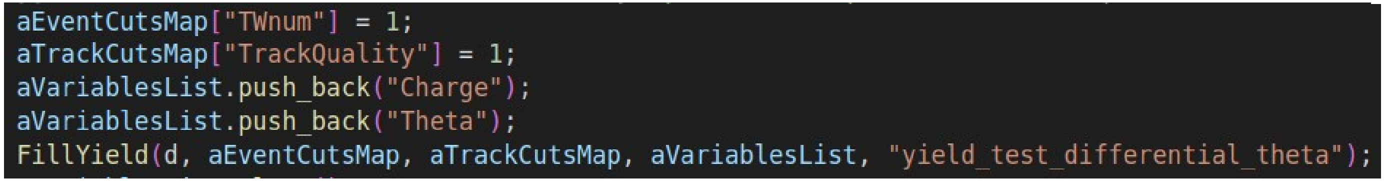


```
    // MCParticleStudies();
    //***** loop on every TAMCparticle:
    FillMCPartYields(); // N_ref
```

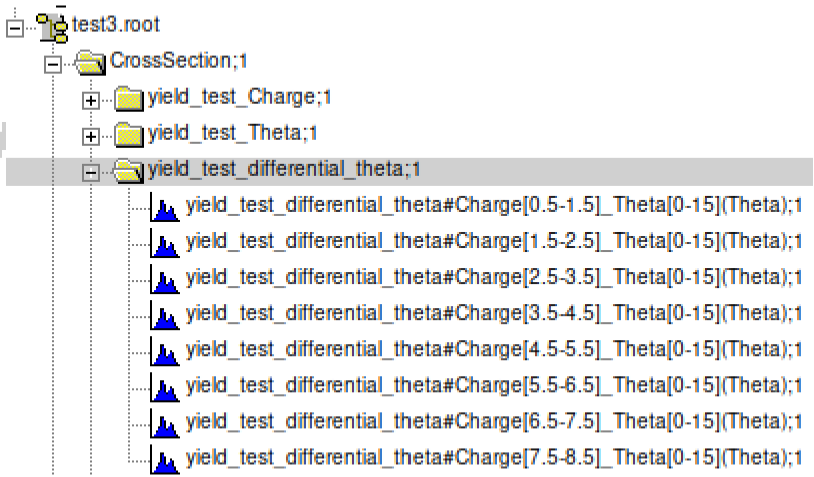


Examples

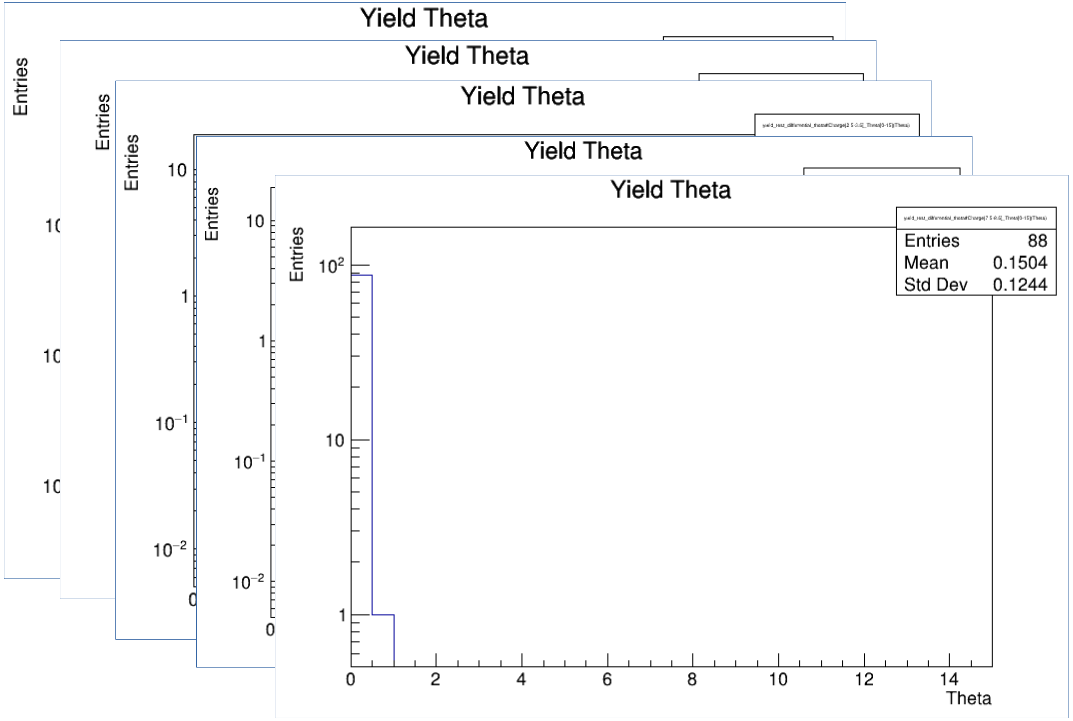
Theta yield, differential in charge



```
aEventCutsMap["TWnum"] = 1;
aTrackCutsMap["TrackQuality"] = 1;
aVariablesList.push_back("Charge");
aVariablesList.push_back("Theta");
FillYield(d, aEventCutsMap, aTrackCutsMap, aVariablesList, "yield_test_differential_theta");
```



```
test3.root
  CrossSection;1
    yield_test_Charge;1
    yield_test_Theta;1
    yield_test_differential_theta;1
      yield_test_differential_theta#Charge[0.5-1.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[1.5-2.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[2.5-3.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[3.5-4.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[4.5-5.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[5.5-6.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[6.5-7.5]_Theta[0-15](Theta);1
      yield_test_differential_theta#Charge[7.5-8.5]_Theta[0-15](Theta);1
```



Conclusions

- new selections can be introduced in *TANAactNtuSelectionCuts* and *TANAactNtuMCref*
- closure test with the previous analysis framework (*DecodeGlbAna*) to see bugs and features
- refine details like paths and parameter files (f.e. parameters binning, “standard” cuts…)
- introduce the machinery for cross section measurements (efficiencies, purity… xsec)
- When everything is fixed, the framework could be easily applied to all the data takings (HIT22, CNAO23 ...)

Back up slides

Selection Cuts, 1

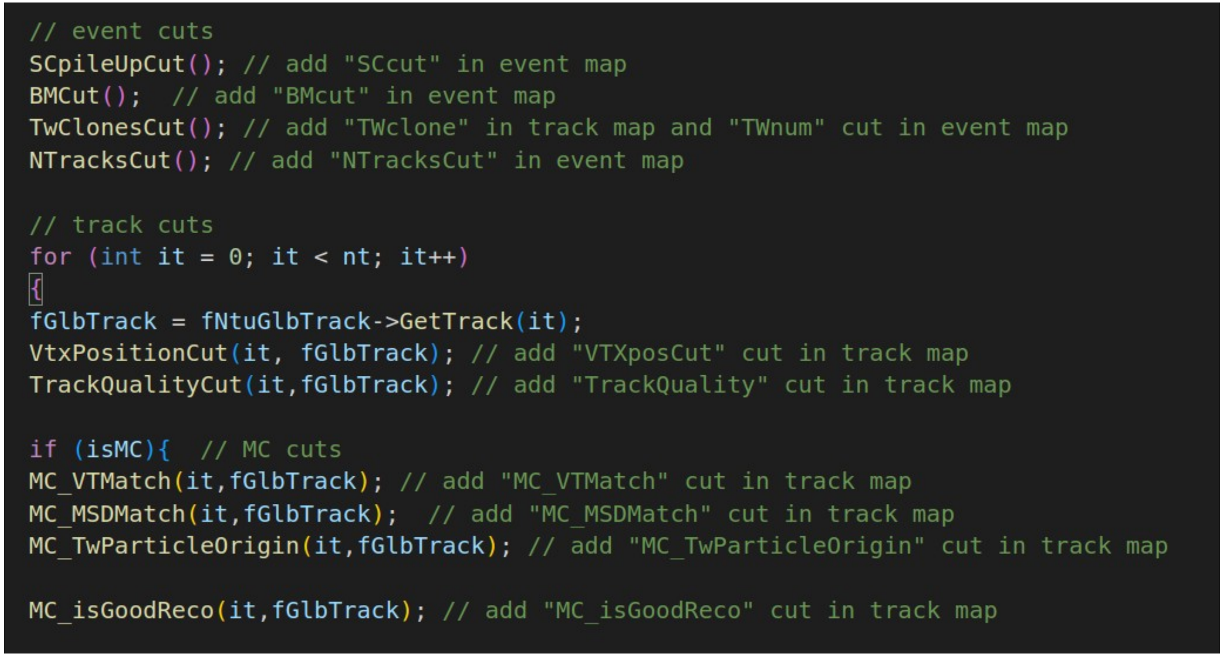
1

TANAactNtuSelectionCuts

In Loop( ):
For every reconstructed track, two cut maps are associated:

- *fEventCutsMap* for element-wise cuts
- *fTrackCutsMap* for track-wise cuts

- for every cut, a key of the map is generated
- an int value of **0,1** (or others if exception) is associated to each key



```
// event cuts
SCpileUpCut(); // add "SCcut" in event map
BMCut();  // add "BMcut" in event map
TwClonesCut(); // add "TWclone" in track map and "TWnum" cut in event map
NTracksCut(); // add "NTracksCut" in event map

// track cuts
for (int it = 0; it < nt; it++)
{
fGlbTrack = fNtuGlbTrack->GetTrack(it);
VtxPositionCut(it, fGlbTrack); // add "VTXposCut" cut in track map
TrackQualityCut(it,fGlbTrack); // add "TrackQuality" cut in track map

if (isMC){  // MC cuts
MC_VTMatch(it,fGlbTrack); // add "MC_VTMatch" cut in track map
MC_MSDMatch(it,fGlbTrack);  // add "MC_MSDMatch" cut in track map
MC_TwParticleOrigin(it,fGlbTrack); // add "MC_TwParticleOrigin" cut in track map

MC_isGoodReco(it,fGlbTrack); // add "MC_isGoodReco" cut in track map
```

Event cuts:
// Check if there is pile up in the SC, triggering an event
// Check if there is only one track in BM
// Check events with N° of tracks == N° of TW points
Check the tracks with the same TW point
// Check if N° of tracks for every event is > 1

Track cuts:
// Cuts about vtx position with the target dimensio
// Cuts about quality chi2 and residual of a track

// Compare the track with the MC_ID to infer if it is a good reco track

Reco quantities, 1

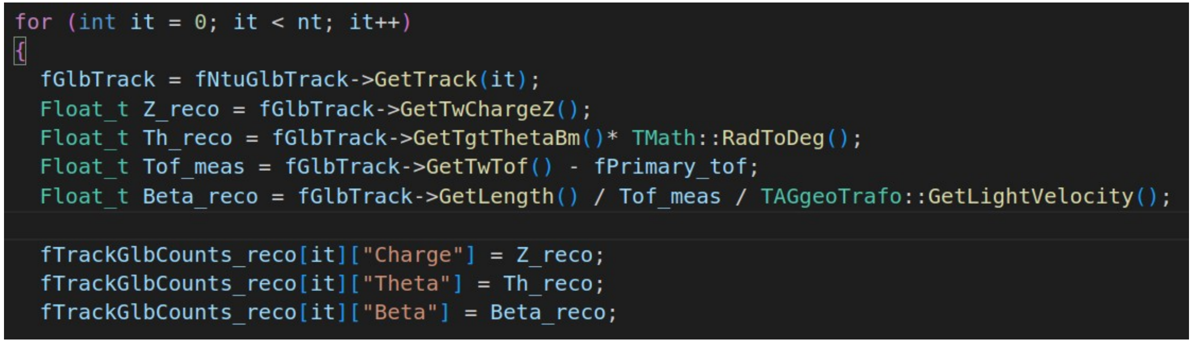
2

TANAactNtuGlbTrackCounts

In Loop( ):

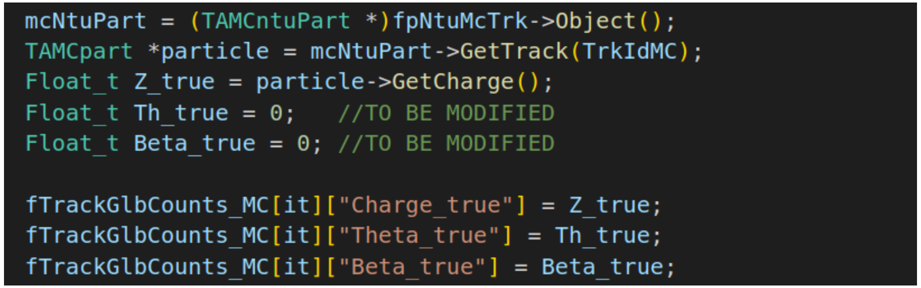
The idea is to create a map for every track, in which the main variables (the one for cross sections) are inserted

- *fTrackGlbCounts_reco* for **reconstructed** values of variables
- *fTrackGlbCounts_MC* for **true** values of variables



```
for (int it = 0; it < nt; it++)
{
  fGlbTrack = fNtuGlbTrack->GetTrack(it);
  Float_t Z_reco = fGlbTrack->GetTwChargeZ();
  Float_t Th_reco = fGlbTrack->GetTgtThetaBm()* TMath::RadToDeg();
  Float_t Tof_meas = fGlbTrack->GetTwTof() - fPrimary_tof;
  Float_t Beta_reco = fGlbTrack->GetLength() / Tof_meas / TAGgeoTrafo::GetLightVelocity();

  fTrackGlbCounts_reco[it]["Charge"] = Z_reco;
  fTrackGlbCounts_reco[it]["Theta"] = Th_reco;
  fTrackGlbCounts_reco[it]["Beta"] = Beta_reco;
```

// Charge
// Theta
// Beta



```
mcNtuPart = (TAMCntuPart *)fpNtuMcTrk->Object();
TAMCpart *particle = mcNtuPart->GetTrack(TrkIdMC);
Float_t Z_true = particle->GetCharge();
Float_t Th_true = 0;    //TO BE MODIFIED
Float_t Beta_true = 0; //TO BE MODIFIED

fTrackGlbCounts_MC[it]["Charge_true"] = Z_true;
fTrackGlbCounts_MC[it]["Theta_true"] = Th_true;
fTrackGlbCounts_MC[it]["Beta_true"] = Beta_true;
```

// Charge
// Theta
// Beta

Improved Interfaces

what we write

```
TTreeReader reader(data);
TTreeReaderValue<A> x(reader,"x");
TTreeReaderValue<B> y(reader,"y");
TTreeReaderValue<C> z(reader,"z");
while (reader.Next()) {
   if (IsGoodEntry(*x, *y, *z))
      h->Fill(*x);
}
```

what we *mean*

- full control over the event loop
- requires some boilerplate
- users implement common tasks again and again
- parallelisation is not trivial

RDataFrame: declarative analyses

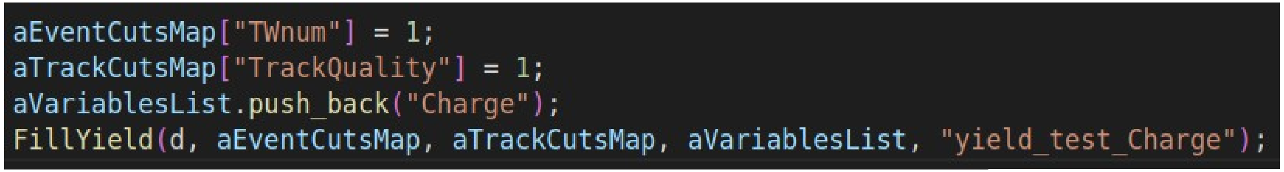
```
RDataFrame d(data);
auto h = d.Filter(IsGoodEntry, {"x","y","z"})
          .Histo1D("x");
```

- full control over *the analysis*
- no boilerplate
- common tasks are already implemented

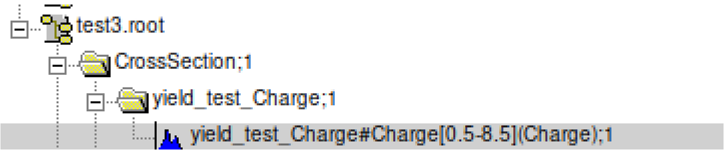
? parallelization is not trivial?

Examples

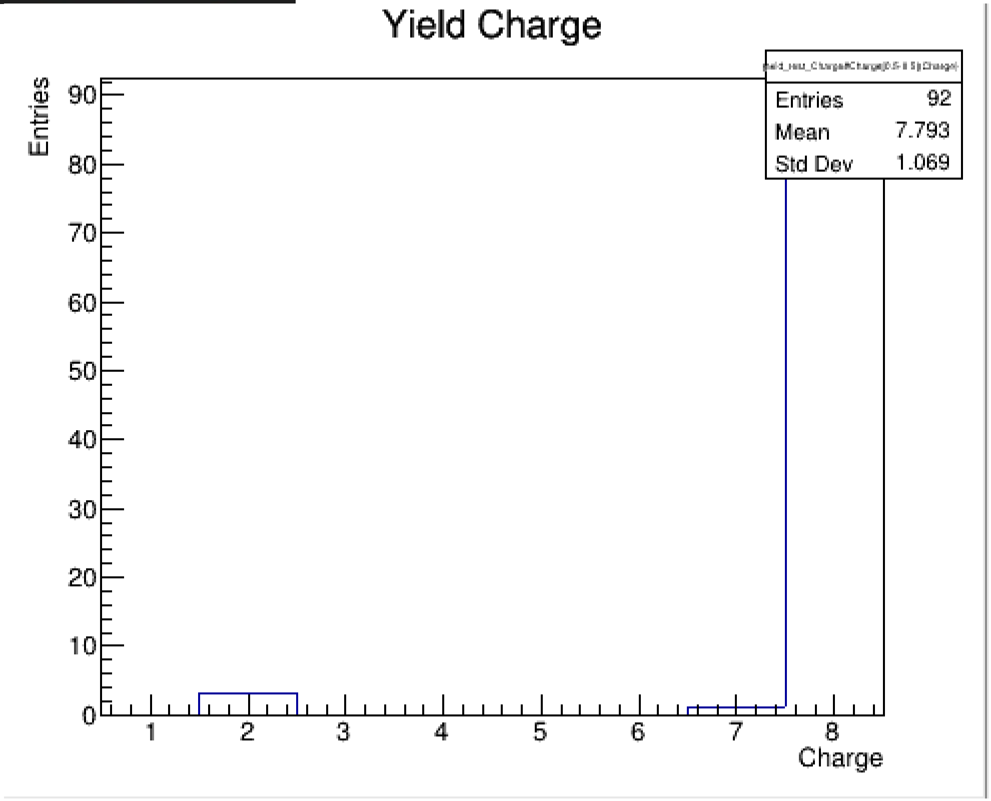
Charge yield



```
aEventCutsMap["TWnum"] = 1;
aTrackCutsMap["TrackQuality"] = 1;
aVariablesList.push_back("Charge");
FillYield(d, aEventCutsMap, aTrackCutsMap, aVariablesList, "yield_test_Charge");
```

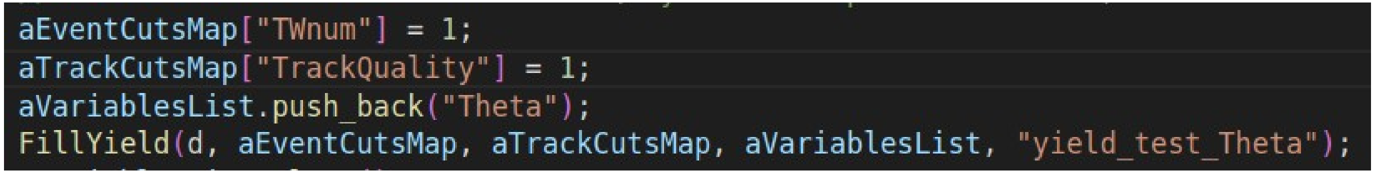


```
test3.root
  CrossSection;1
    yield_test_Charge;1
      yield_test_Charge#Charge[0.5-8.5](Charge);1
```

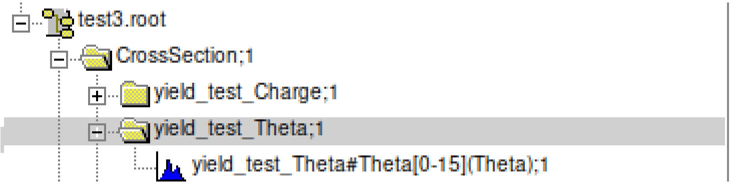


Examples

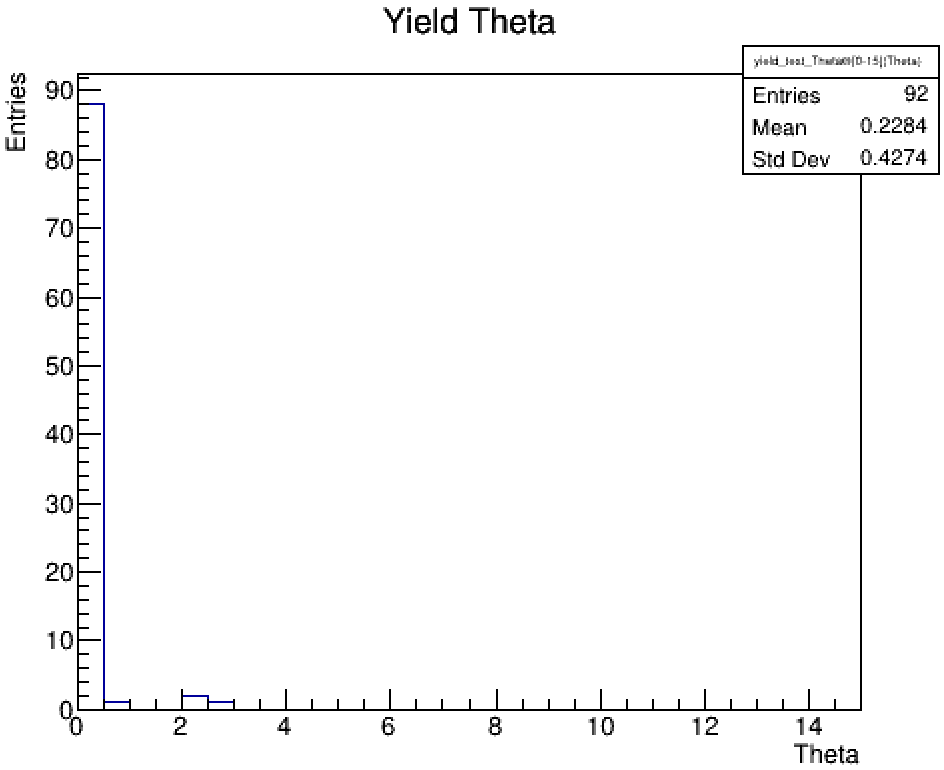
Theta yield



```
aEventCutsMap["TWnum"] = 1;
aTrackCutsMap["TrackQuality"] = 1;
aVariablesList.push_back("Theta");
FillYield(d, aEventCutsMap, aTrackCutsMap, aVariablesList, "yield_test_Theta");
```

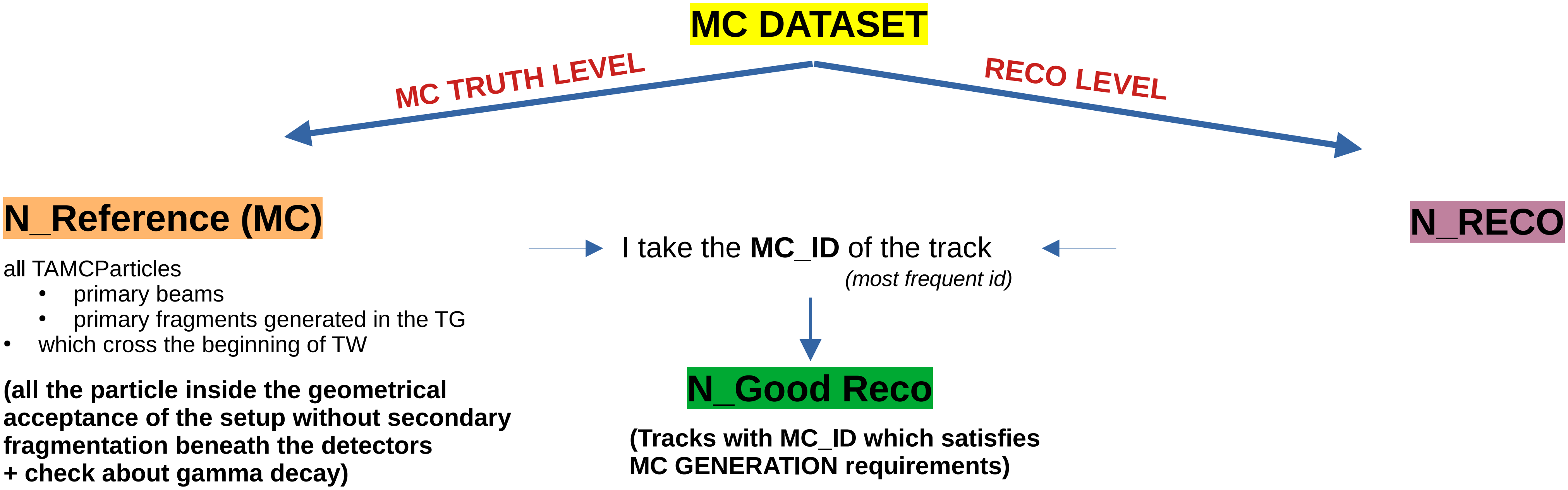


```
test3.root
  CrossSection;1
    yield_test_Charge;1
    yield_test_Theta;1
      yield_test_Theta#Theta[0-15](Theta);1
```



Analysis strategy

In the analysis, I am considering the following levels:



MC DATASET

MC TRUTH LEVEL

RECO LEVEL

N_Reference (MC)

all TAMCParticles

- primary beams
- primary fragments generated in the TG

- which cross the beginning of TW

(all the particle inside the geometrical acceptance of the setup without secondary fragmentation beneath the detectors + check about gamma decay)

N_RECO

I take the **MC_ID** of the track

(most frequent id)

N_Good Reco

(Tracks with MC_ID which satisfies MC GENERATION requirements)

Analysis strategy

To compute angular differential cross section:

$$\frac{d\sigma}{d\theta}(Z,\theta) = \frac{Y(Z,\theta)}{N_{beam}\ N_{target}\ \Omega_{\theta}\ \epsilon(Z,\theta)}$$

where:

Y:	fragment counts	**N_RECO**
N_{beam}:	n° of primary events	
N_{target}:	n° of scattering centers per unit area	
ε:	efficiency	**N_Good Reco / N_Reference (MC)**
Ω_{ϑ}:	angular phase space	