A Study on the Tracking Performance Modules

Giulia Casarosa

JG|U

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Alexander von Humboldt
Stiftung / Foundation

Virtual Tracking Meeting, ~ March 17th 2017

Outline

☑ *Tracking Acceptance*

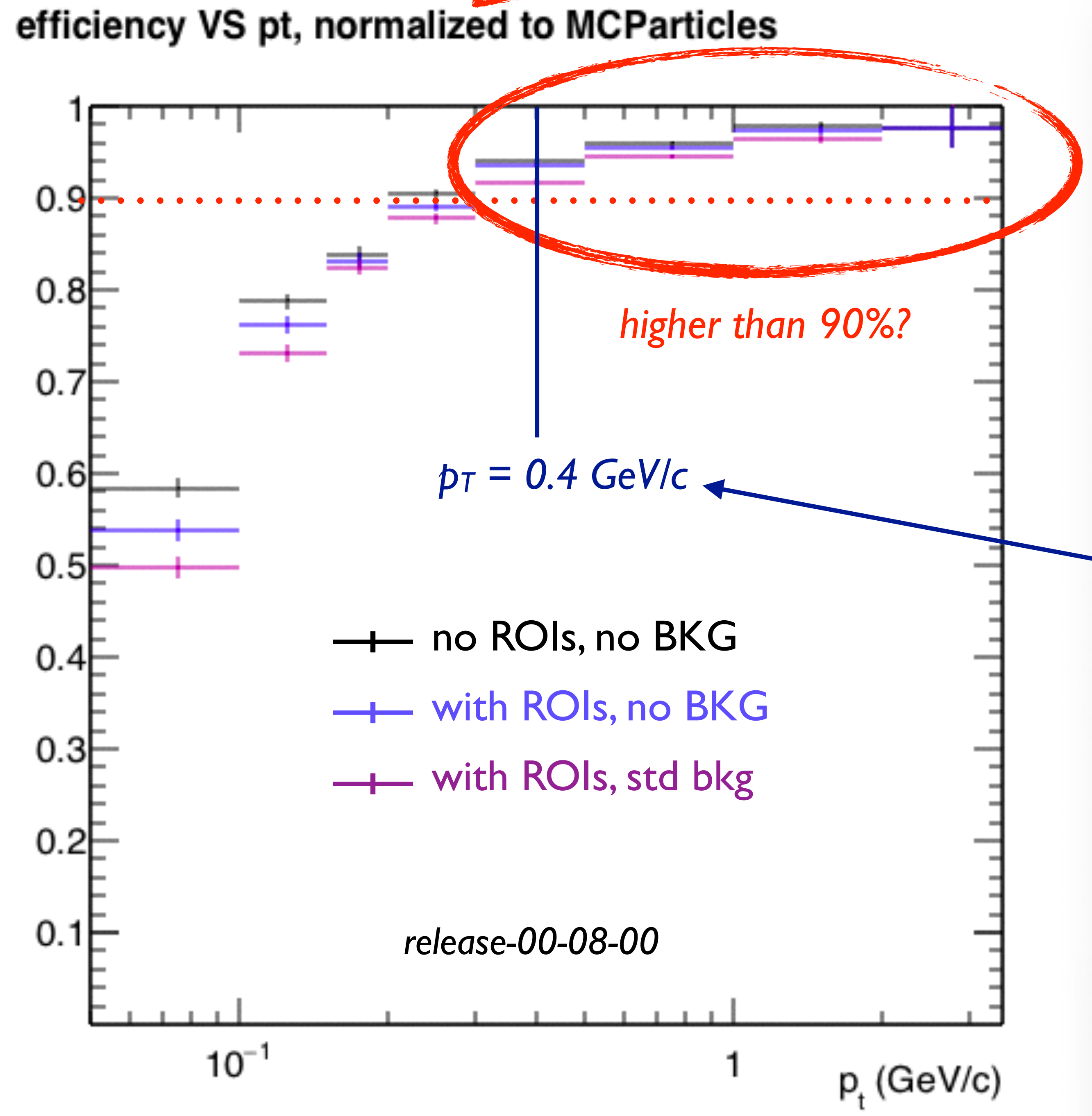
☑ *Performance Evaluation Modules*

  ○ *RecoTracks*

☑ *Plans*

Efficiency at High p_t

efficiency includes geometrical acceptance



from BaBar NIM:

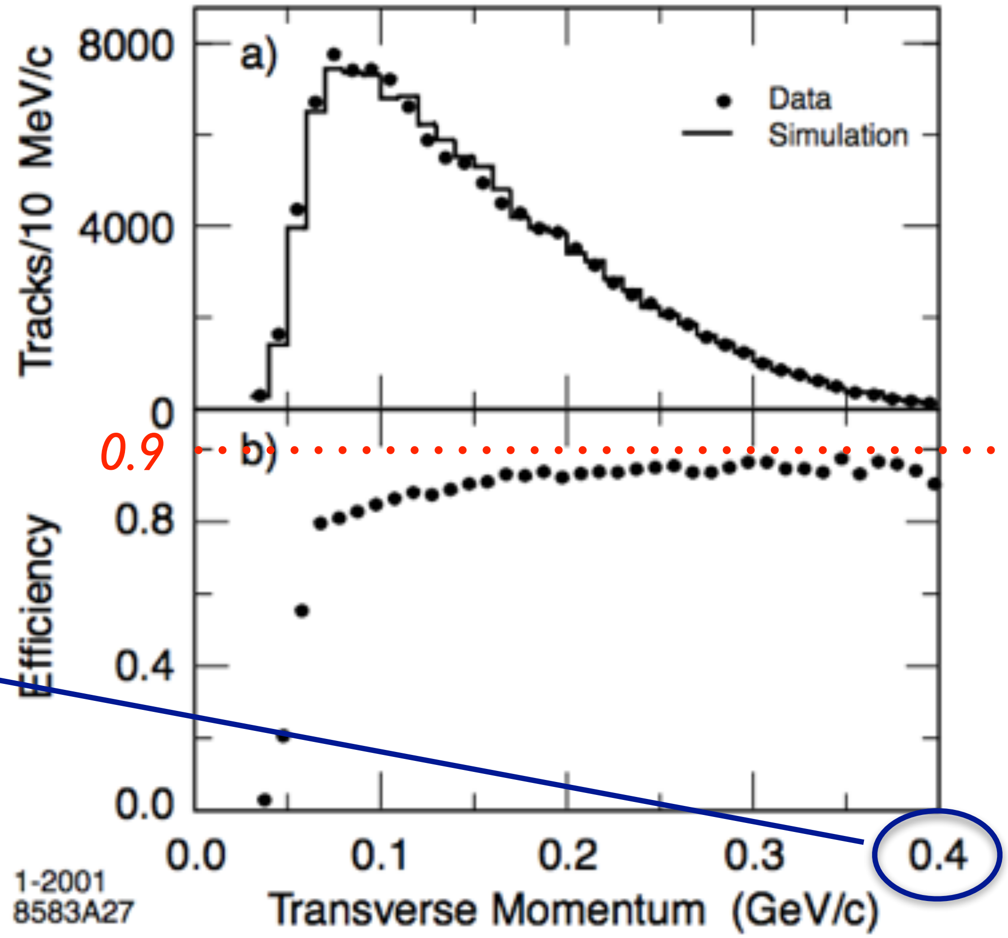


Figure 42. Monte Carlo studies of low momentum tracks in the SVT: a) comparison of data (contributions from combinatoric background and non-$B\overline{B}$ events have been subtracted) with simulation of the transverse momentum spectrum of pions from $D^{*+} \rightarrow D^0\pi^+$ in $B\overline{B}$ events, and b) efficiency for slow pion detection derived from simulated events.

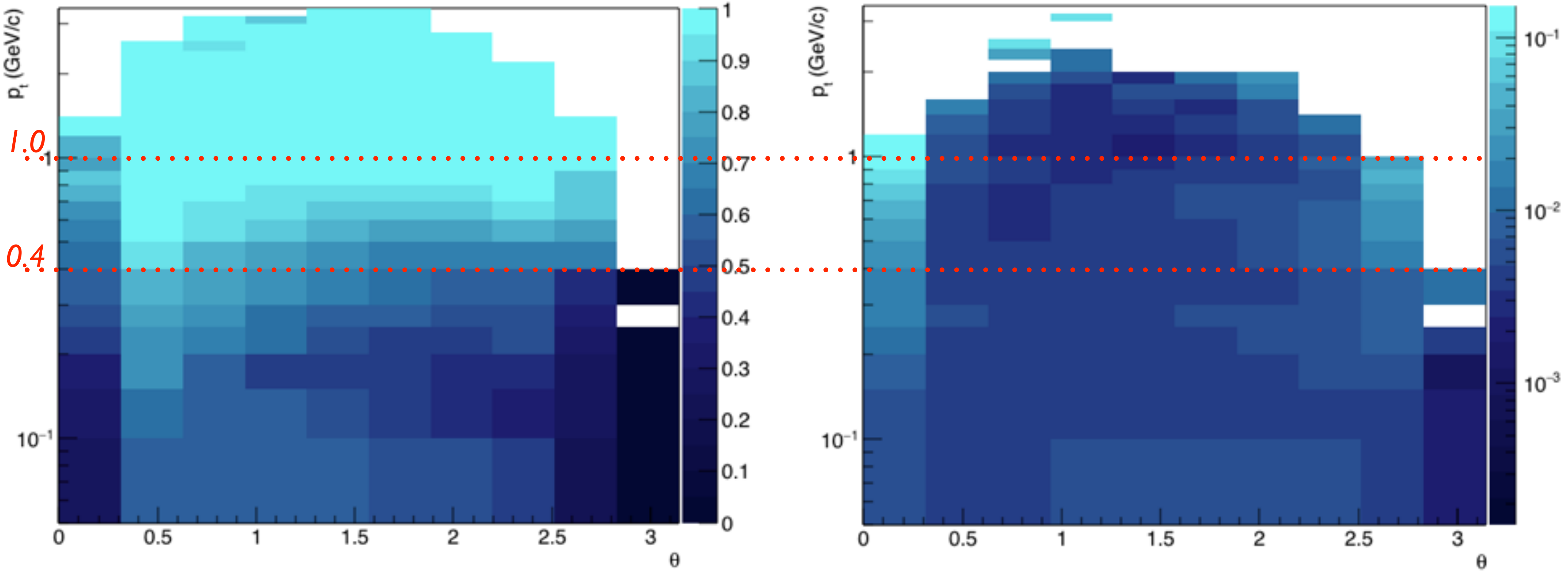
Acceptance

thanks to Bianca who provided the plots

- ➡ Define the acceptance using MCParticles and MCTrackCands as:

$$\text{acceptance} = \frac{\text{\#MCParticles with one associated MCTrackCand}}{\text{\# MCParticles}}$$

- ➡ Measure it with simulated pions from inside the beam pipe in 2D plot (θ, p_T) :

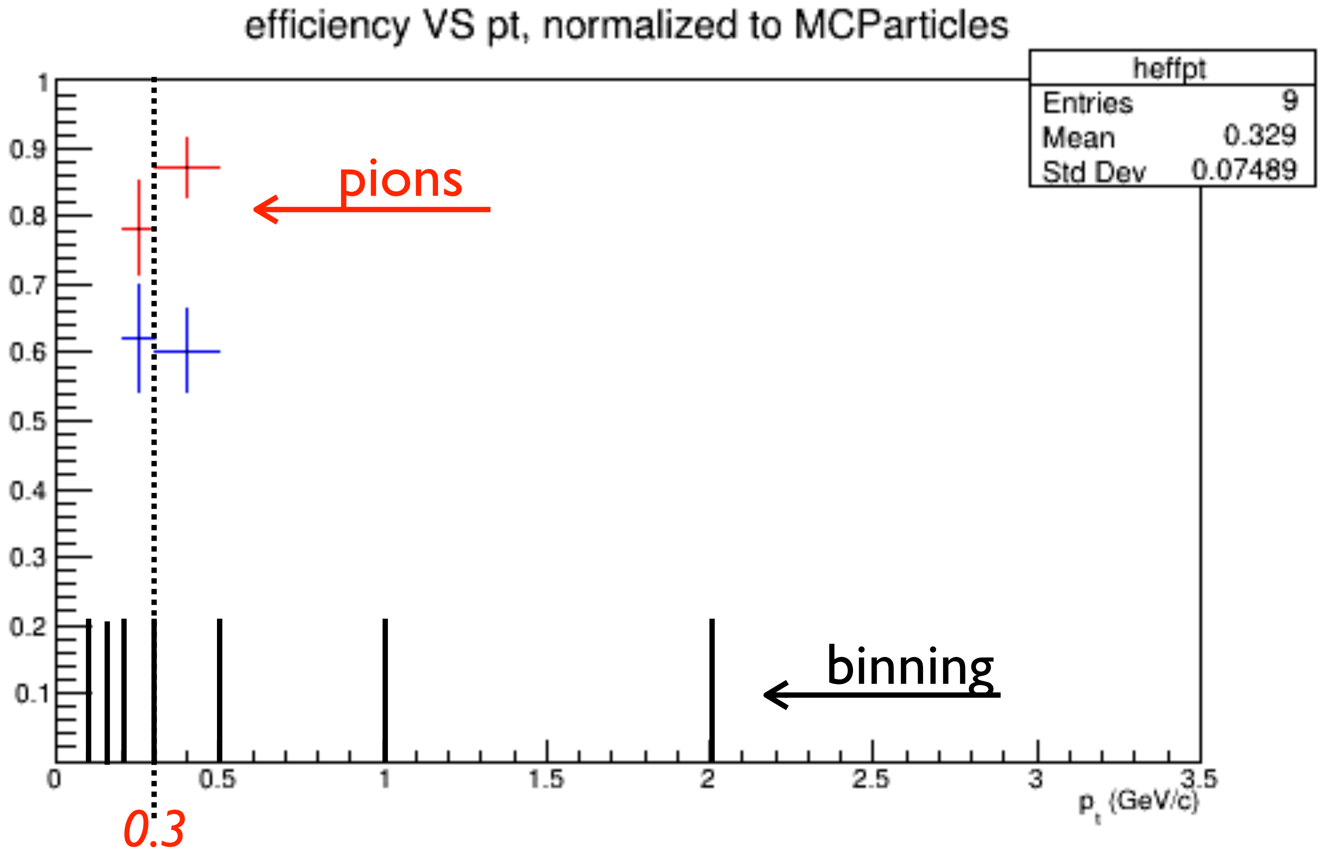


acceptance

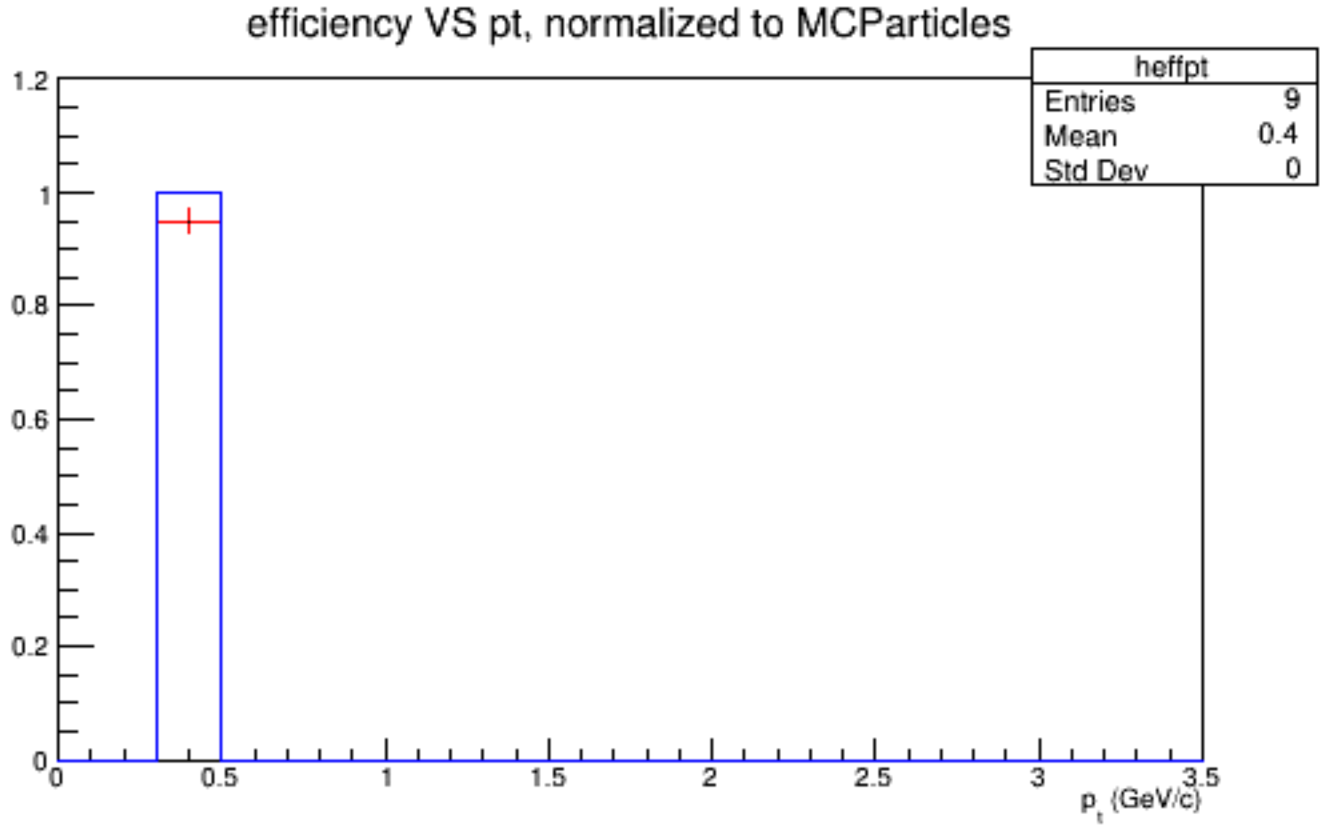
acceptance error

A Strange Behaviour

S. Spataro pointed me to some strange behaviour of the efficiency plots:



- ➡ shoot pions with ParticleGun
- ➡ p_T = **0.3** GeV/c, θ= 60 deg
- ➡ observe 2 non-empty bins

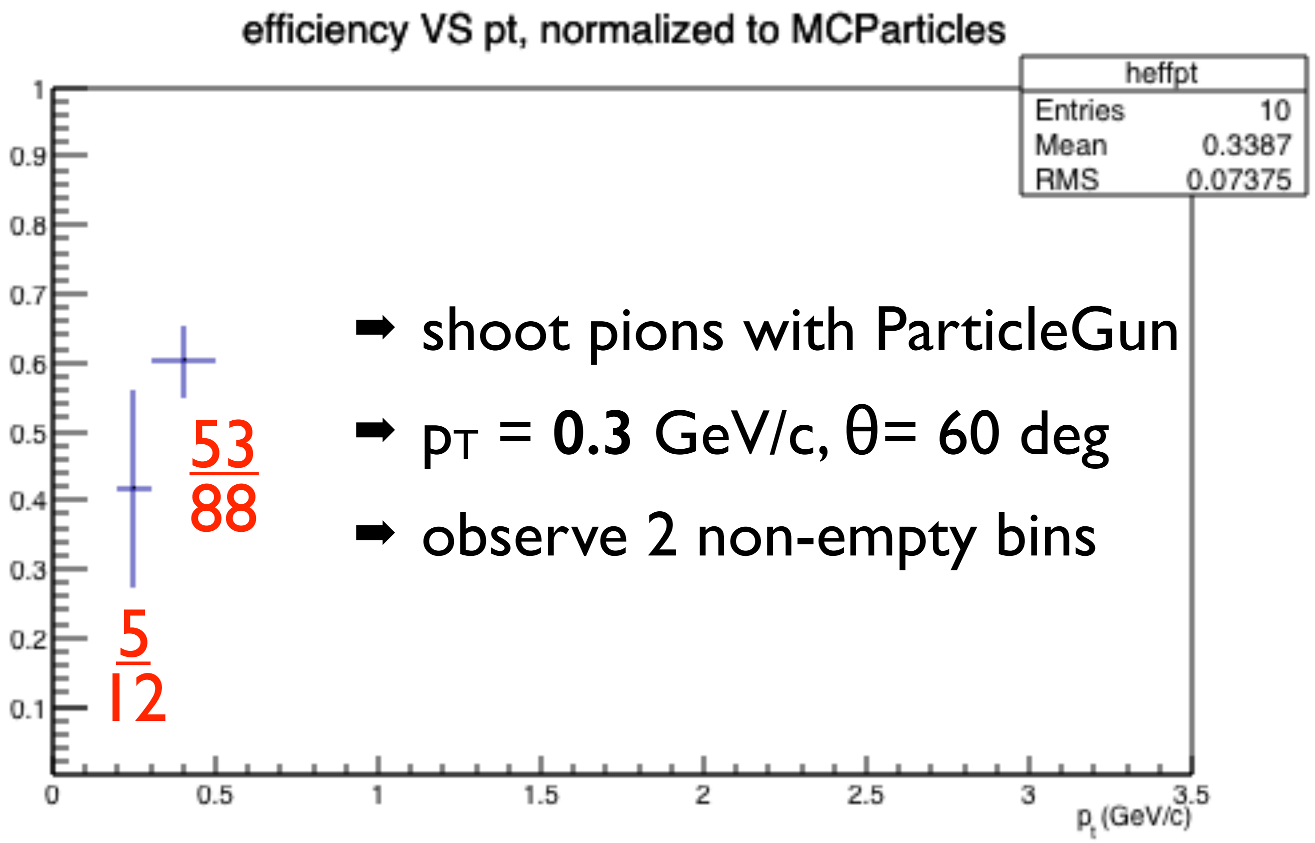


- ➡ shoot pions with ParticleGun
- ➡ p_T = **0.31** GeV/c, θ= 60 deg
- ➡ observe 1 non-empty bin

Investigation

$\frac{\varepsilon\text{-numerator}}{\varepsilon\text{-denominator}}$

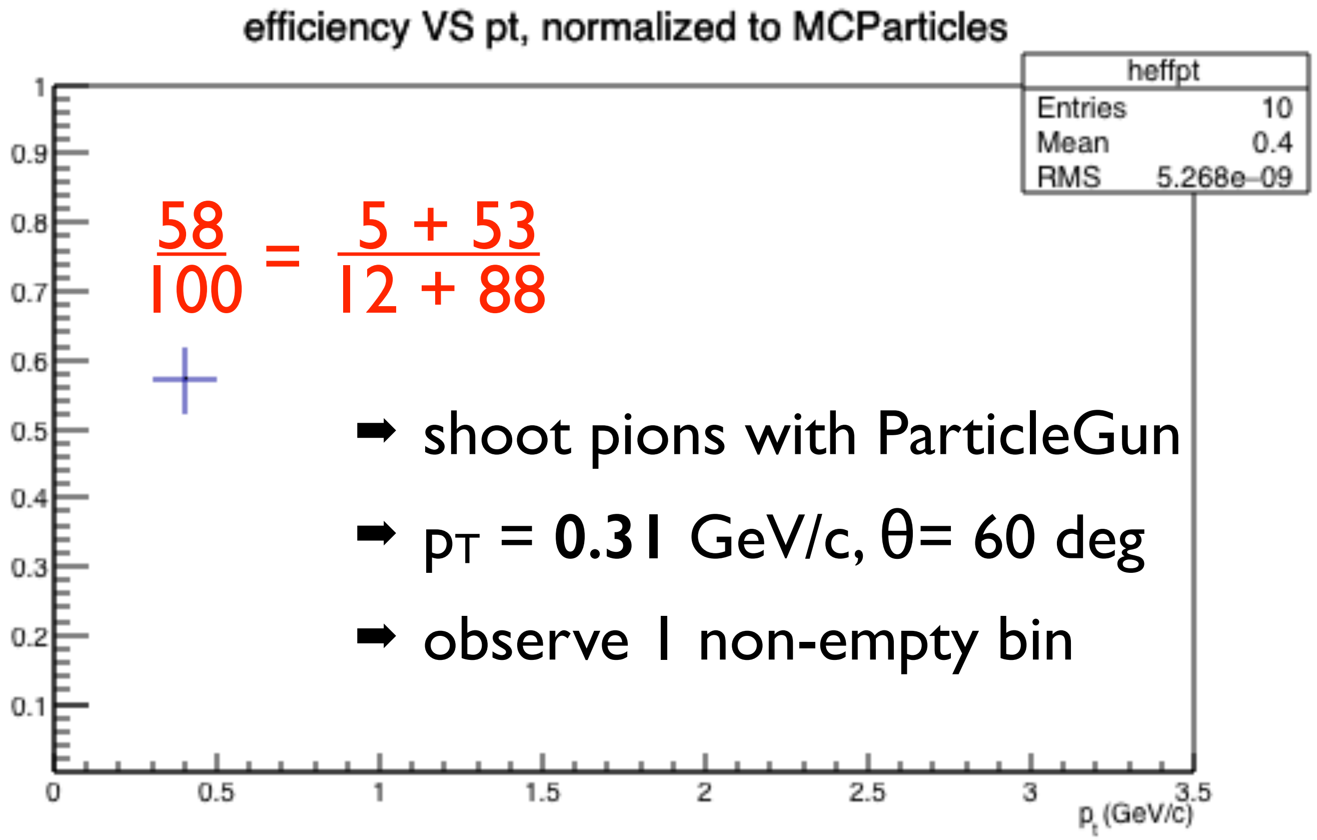
I was able to reproduce the behaviour and study it with some more tests:



$\frac{53}{88}$

$\frac{5}{12}$

- ➡ shoot pions with ParticleGun
- ➡ p_T = **0.3** GeV/c, θ= 60 deg
- ➡ observe 2 non-empty bins



$\frac{58}{100} = \frac{5 + 53}{12 + 88}$

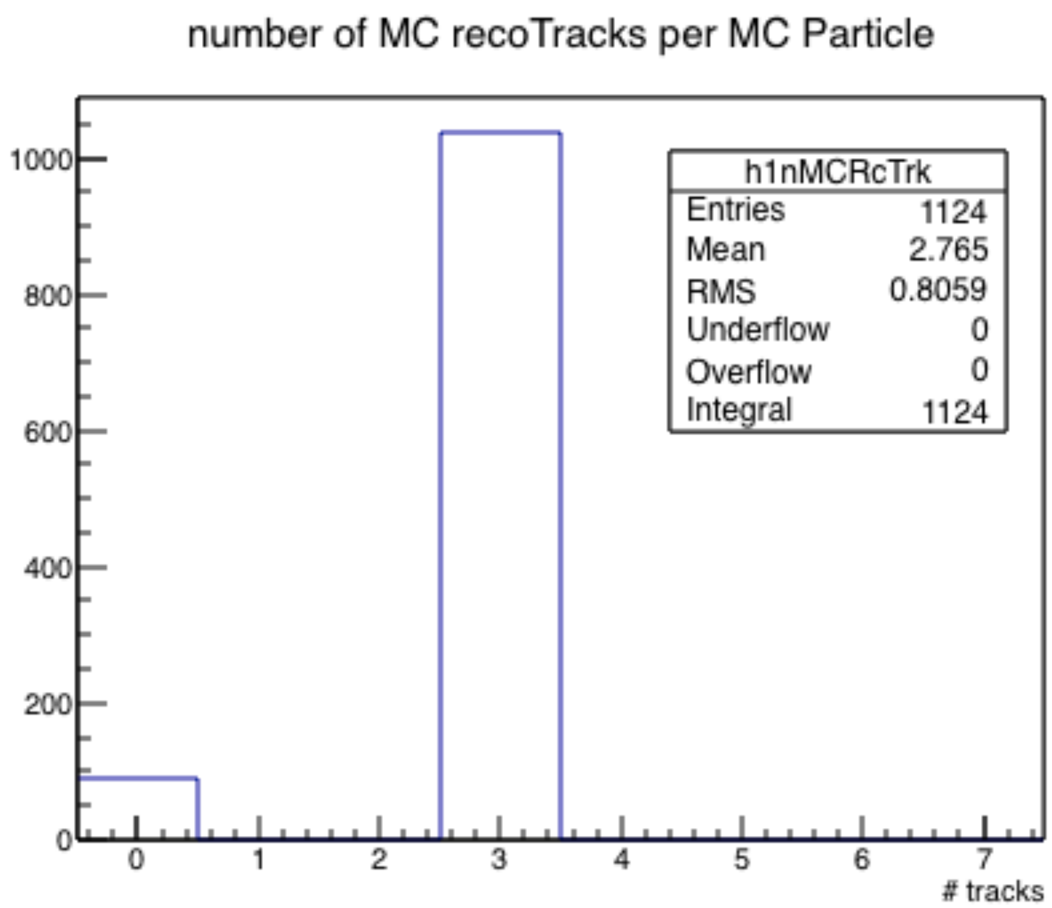
- ➡ shoot pions with ParticleGun
- ➡ p_T = **0.31** GeV/c, θ= 60 deg
- ➡ observe 1 non-empty bin

- ➡ the total number of entries is correct
- ➡ the efficiency is correctly estimated
- ➡ same behaviour observed for 1 GeV/c pions
- ➡ rounding is sometimes *wrong* and the entry goes into another bin
  - with p_T > 1 GeV/c + 1 eV/c only one bin is non-zero
  - otherwise, rounding problems show up

classified as:
not an issue

MCRecoTracks Related to MCParticles

- ➡ Used add_tracking_reconstruction()
  - ‣ with **MCTrackFinding = True**
- ➡ I expect either 0 or 1 MCRecoTrack per MCParticle
  - 0, ok because outside acceptance
  - or 3:
    - ‣ VXD, CDC, VXD+CDC MCRecoTracks?
    - ‣ shouldn't these StoreArrays have different names?

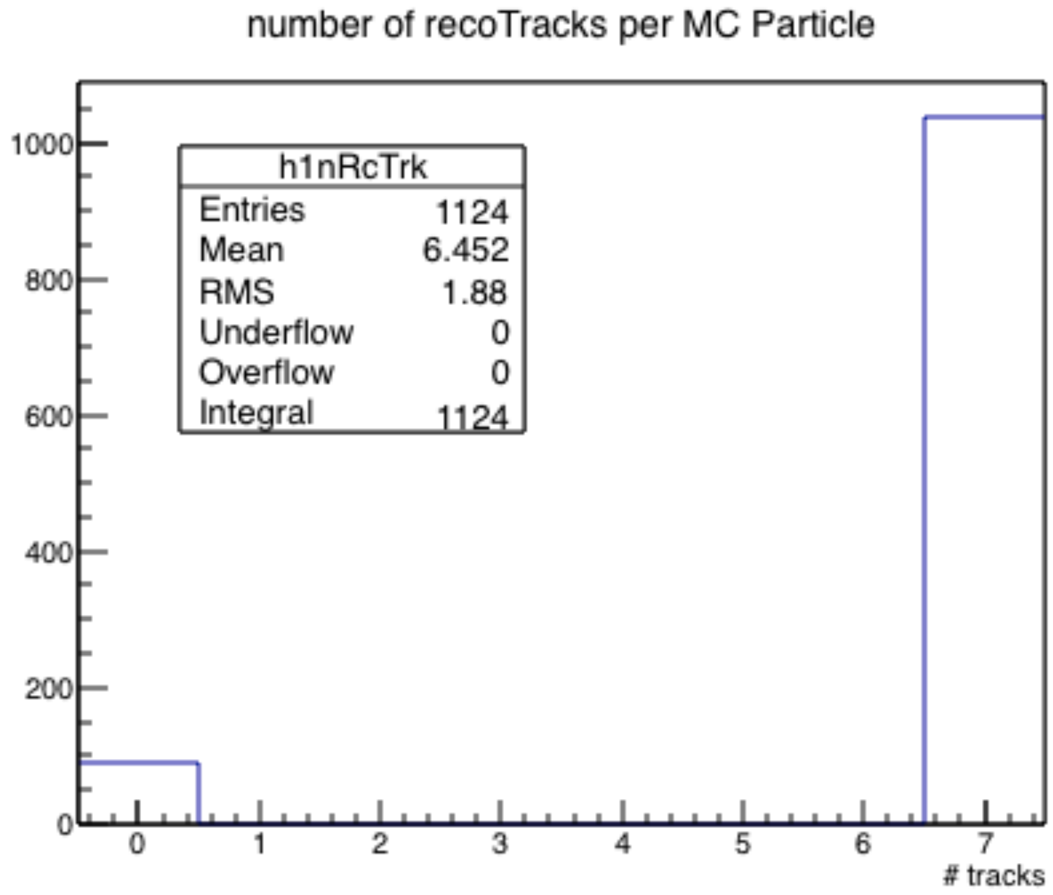


```
with m_MCRecoTracksName = MCRecoTracks
       m_RecoTracksName = RecoTracks
```

```
StoreArray<MCParticle> mcParticles(m_MCParticlesName);

BOOST_FOREACH(MCParticle & mcParticle, mcParticles) {

RelationVector<RecoTrack> MCRecoTracks_fromMCParticle =
      DataStore::getRelationsWithObj<RecoTrack>(&mcParticle,m_MCRecoTracksName);

    m_multiplicityMCRecoTracks->Fill(MCRecoTracks_fromMCParticle.size() );

 RelationVector<RecoTrack> RecoTracks_fromMCParticle =
      DataStore::getRelationsWithObj<RecoTrack>(&mcParticle,m_RecoTracksName);

    m_multiplicityRecoTracks->Fill(RecoTracks_fromMCParticle.size() );

}
```

RecoTracks Related to MCParticles

- ➡ Used add_tracking_reconstruction()
  - ‣ with MCTrackFinding = True
- ➡ I expect 1 RecoTrack per MCParticle
  - either 0 (outside acceptance)
  - or 7:
    - ‣ VXD, CDC, VXD+CDC MCRecoTracks +
    - ‣ VXD, CDC, VXD+CDC RecoTracks + ???



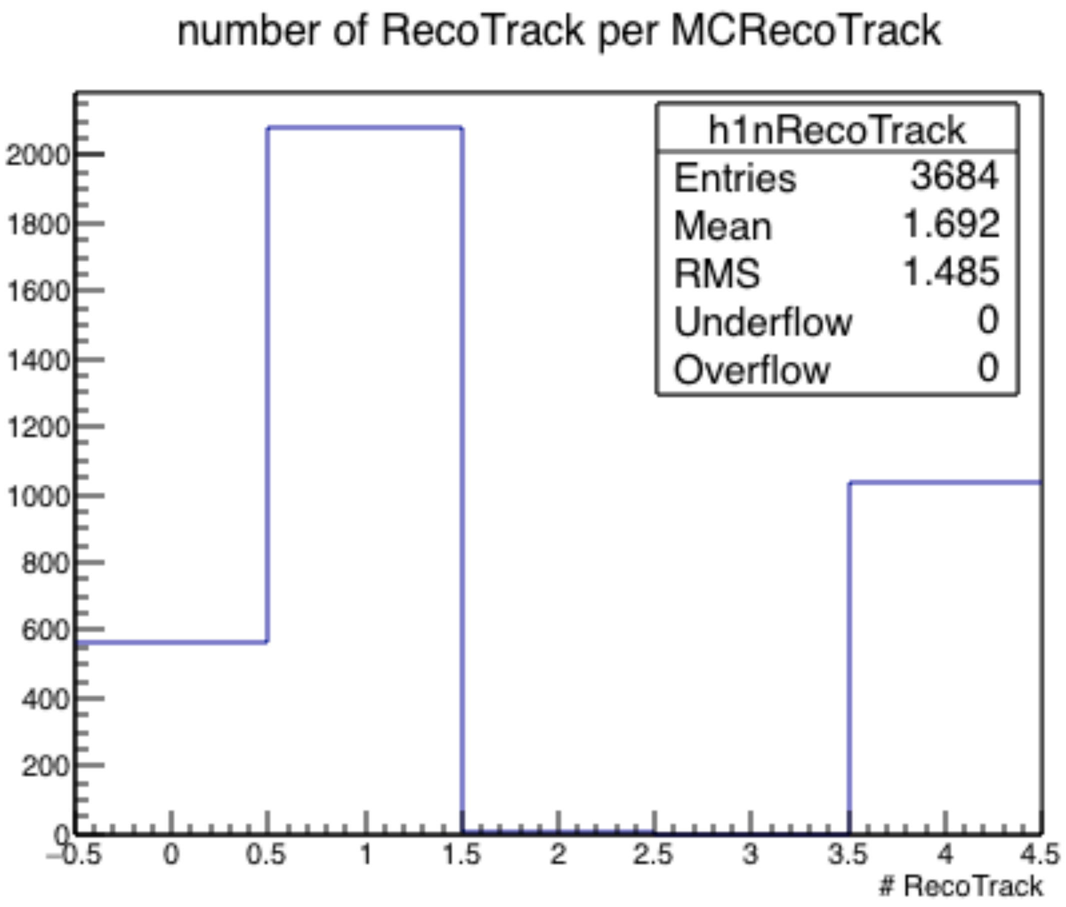
```
with m_MCRecoTracksName = MCRecoTracks
       m_RecoTracksName = RecoTracks
```

```
StoreArray<MCParticle> mcParticles(m_MCParticlesName);

 BOOST_FOREACH(MCParticle & mcParticle, mcParticles) {

 RelationVector<RecoTrack> MCRecoTracks_fromMCParticle =
       DataStore::getRelationsWithObj<RecoTrack>(&mcParticle,m_MCRecoTracksName);

    m_multiplicityMCRecoTracks->Fill(MCRecoTracks_fromMCParticle.size() );

  RelationVector<RecoTrack> RecoTracks_fromMCParticle =
       DataStore::getRelationsWithObj<RecoTrack>(&mcParticle,m_RecoTracksName);

    m_multiplicityRecoTracks->Fill(RecoTracks_fromMCParticle.size() );

 }
```

RecoTracks Related to MCRecoTracks

- ➡ Used add_tracking_reconstruction()
  - with MCTrackFinding = True

```
with m_MCRecoTracksName = MCRecoTracks
     m_RecoTracksName = RecoTracks
```

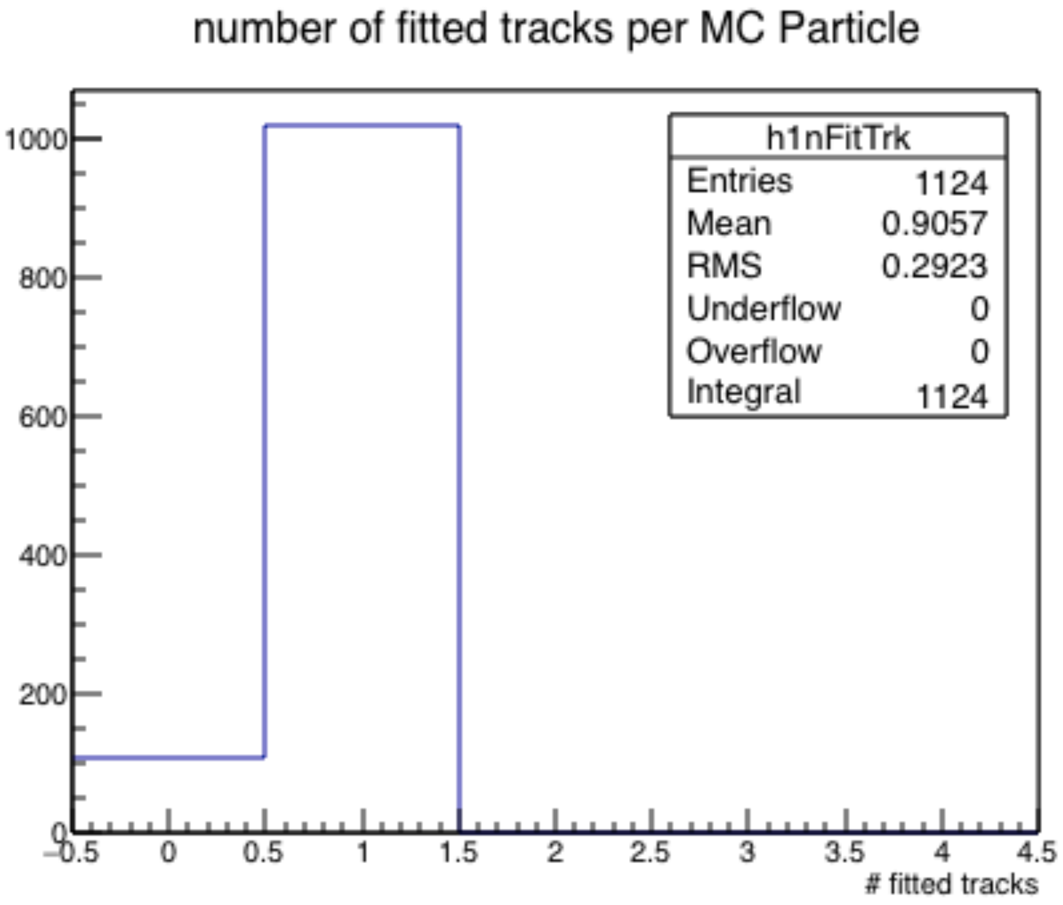


```
BOOST_FOREACH(RecoTrack & mcRecoTrack, mcRecoTracks) {


RelationVector<RecoTrack> RecoTracks_fromMCRecoTrack =
    DataStore::getRelationsWithObj<RecoTrack>(&mcRecoTrack);

 m_multiplicityRecoTracksPerMCRT->Fill(RecoTracks_fromMCRecoTrack.size());

}
```

➡ Maybe it does not make a lot of sense to look for RecoTracks when using MCTrackFinding

Tracks Related to MCParticles

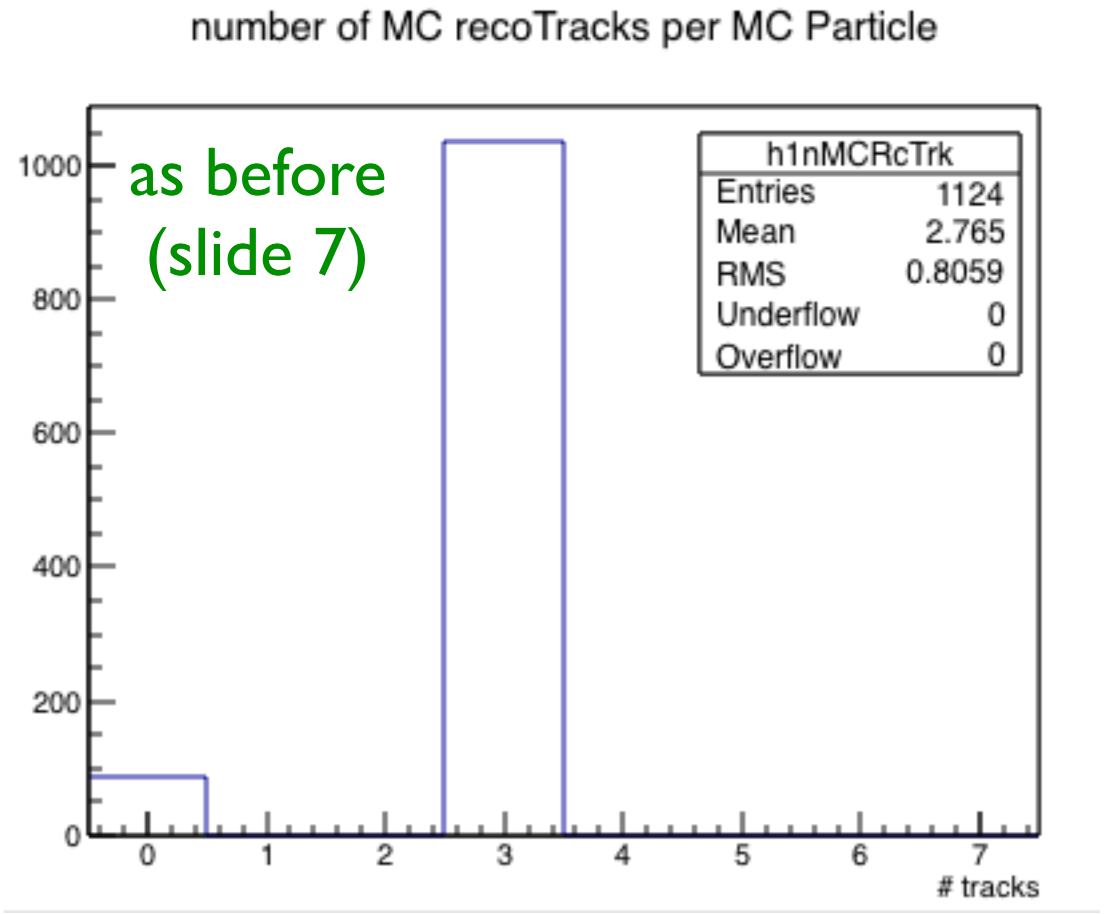
- ➡ Used add_tracking_reconstruction()
  - ‣ with MCTrackFinding = True
- ➡ I expect either 0 or 1 Track per MCParticle
  - 0, ok because outside acceptance
  - or 1, ok
- ➡ Only one RecoTrack StoreArray is passed to the DAF



```
StoreArray<MCParticle> mcParticles(m_MCParticlesName);

 BOOST_FOREACH(MCParticle & mcParticle, mcParticles) {

 RelationVector<Track> Tracks_fromMCParticle = DataStore::getRelationsWithObj<Track>(&mcParticle);
     m_multiplicityTracks->Fill(Tracks_fromMCParticle.size());

 }
```

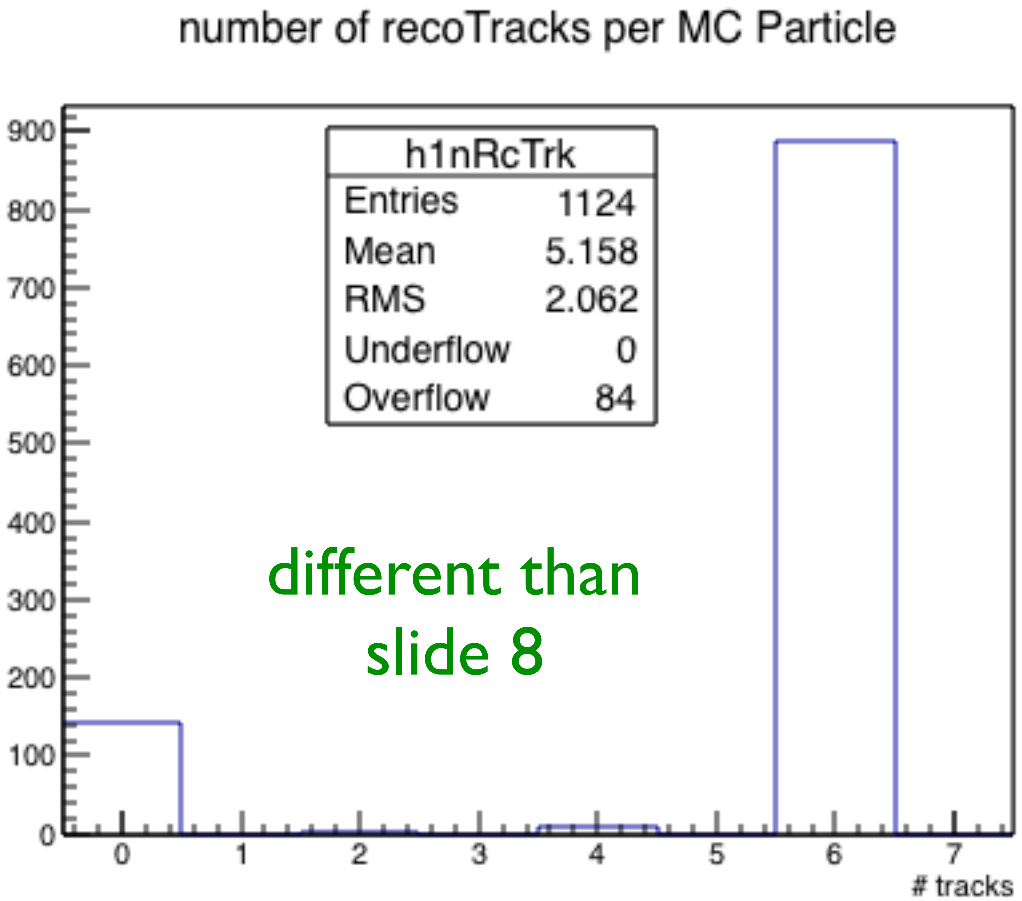
Moving to Real TrackFinder

- ➡ Used add_tracking_reconstruction()
  - with **MCTrackFinding = False**



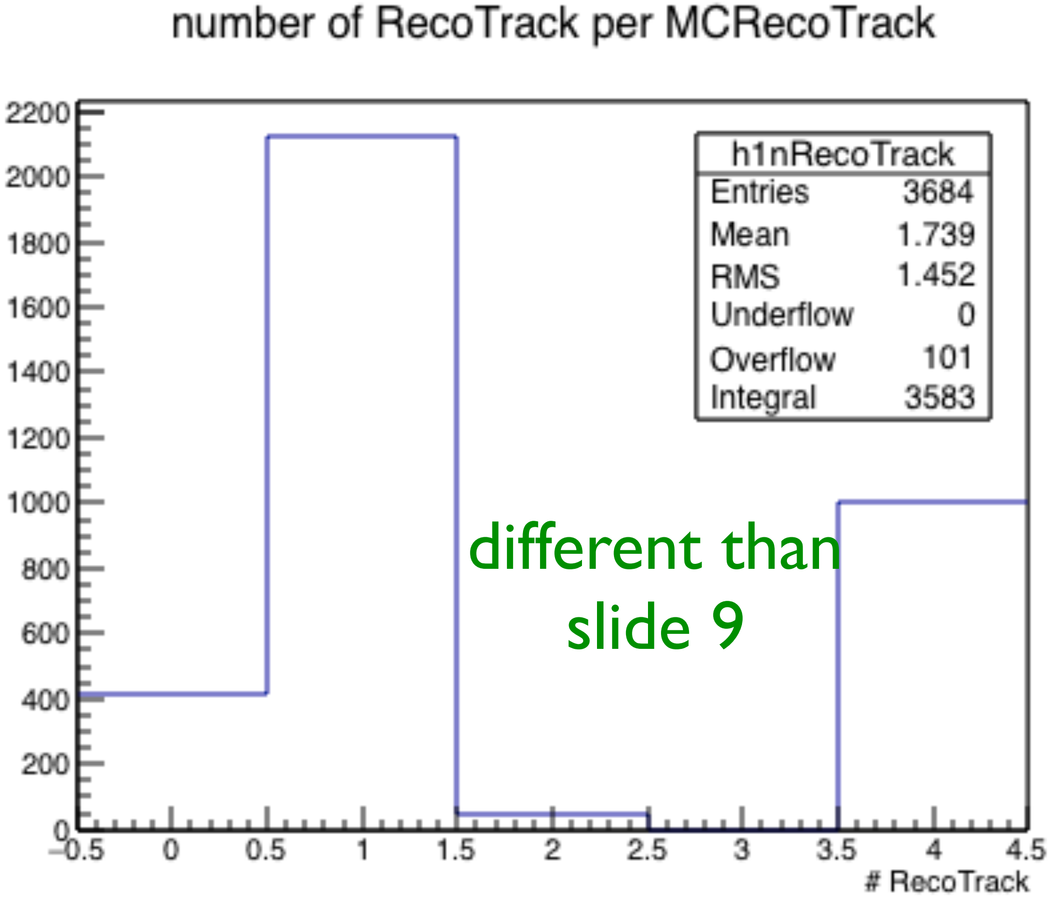
number of MC recoTracks per MC Particle

as before (slide 7)



number of recoTracks per MC Particle

different than slide 8



number of RecoTrack per MCRecoTrack

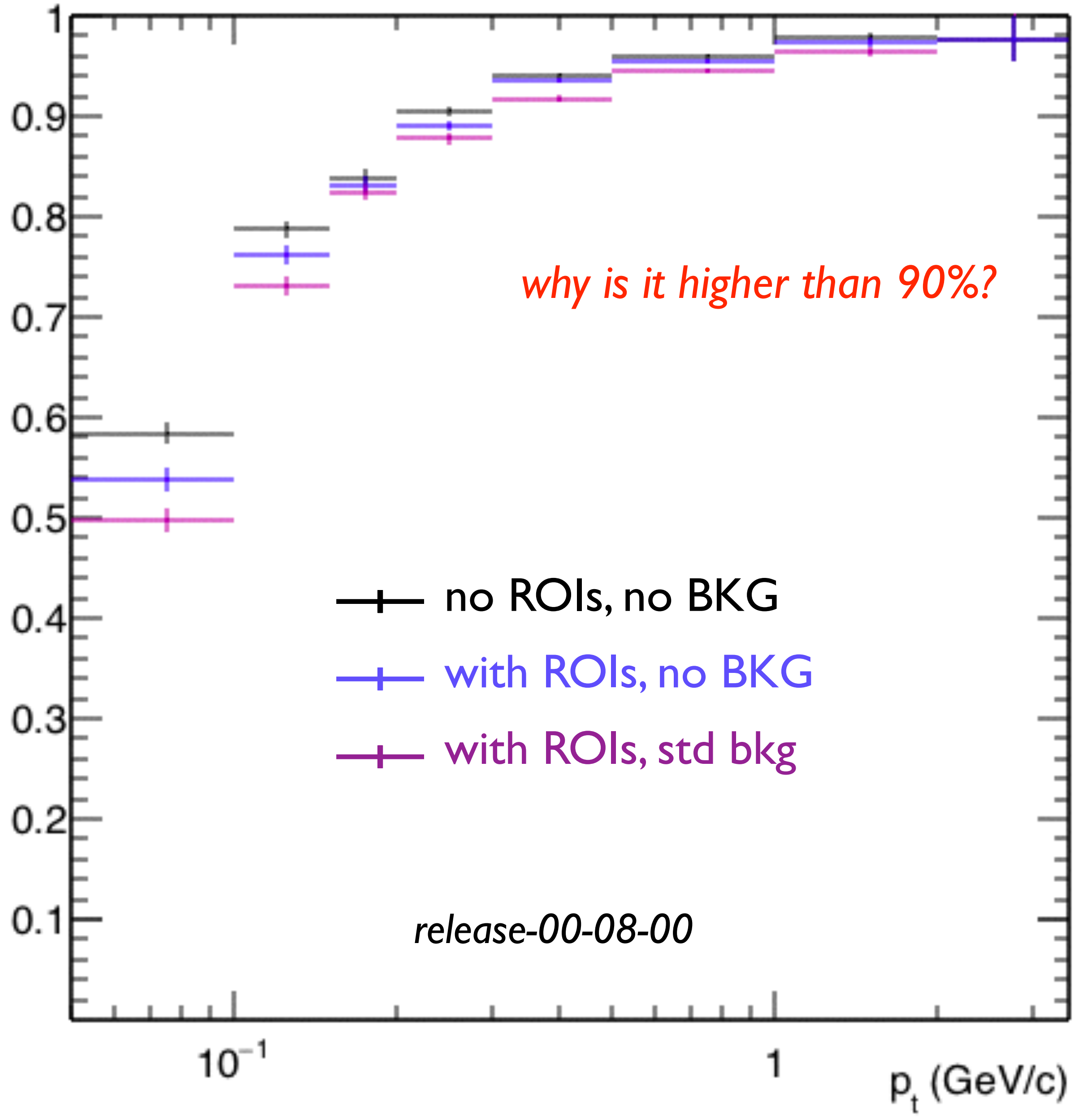
different than slide 9

Plans

- ➡ Define the correct use of RecoTracks in TrackingPerformanceEvaluation module
- ➡ RecoTrack transition in the other TrackingPerformanceValidation modules
  - V0FindingPerformance Module
  - TBAnalysis Module
- ➡ **TestBeam ROI Analysis**: compare ROIs done with VXDTF1 with ROIs done with VXDTF2
  - strategy:
    1. analyse two runs with possibly exactly the same conditions except VXDTF
    2. count the number of ROIs with a pixel "near" the center in the two runs
    3. the ratio of these two numbers estimates how better VXDTF2 is wrt VXDTF1
  - assumptions:
    4. fit + track extrapolation + ROI definition efficiencies cancels in the ratio
    5. PXD sensor efficiency cancels (i.e. different modules have same efficiency)

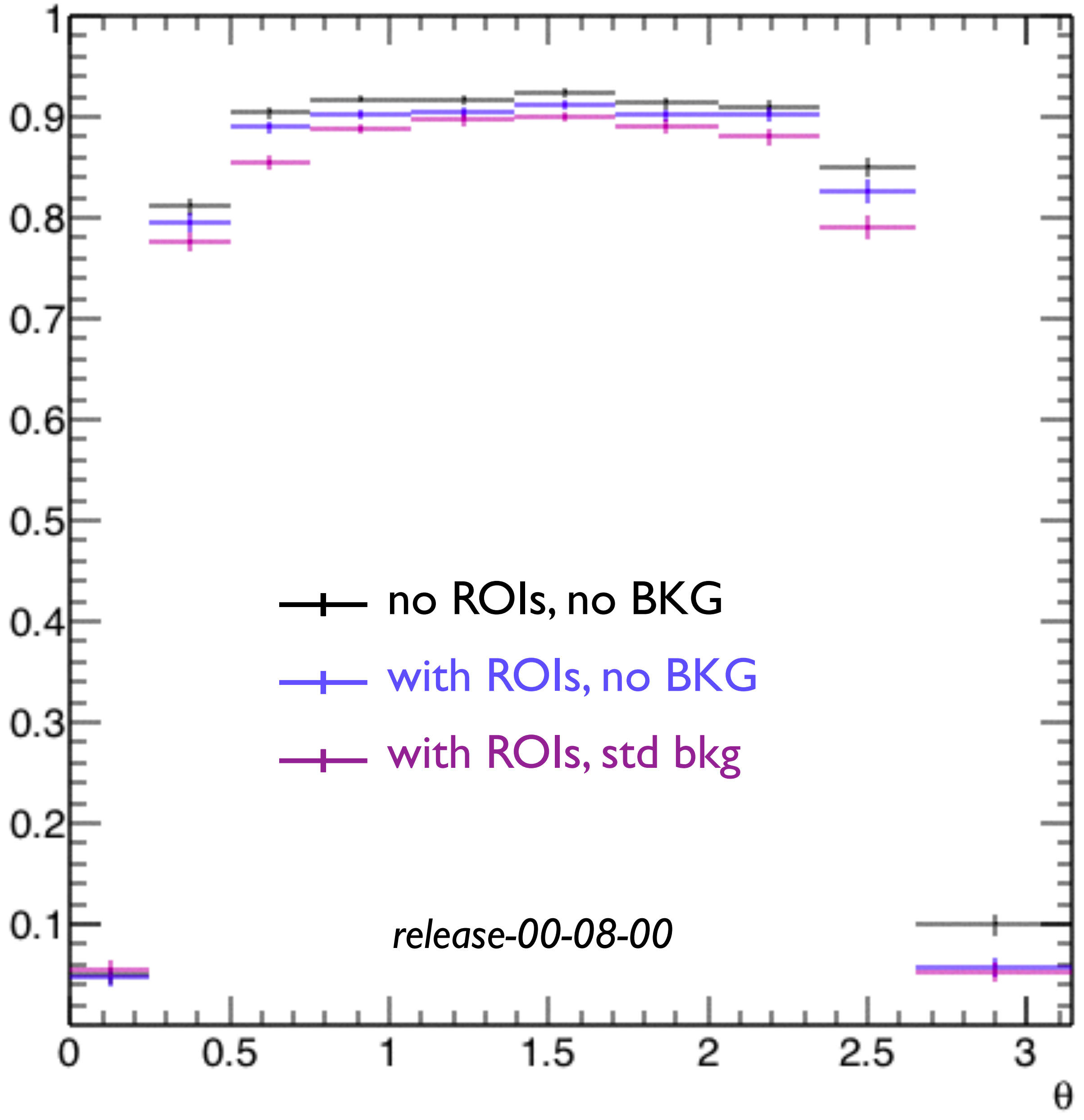
Efficiency vs p_T, and polar angle (1)

efficiency VS pt, normalized to MCParticles



why is it higher than 90%?

efficiency VS θ, normalized to MCParticles



➡ Background and PXD data reduction effects slightly depend on the transverse momentum and the polar angle of the track