

# CSC W/Z+jets group Common Ntuples

By

Alessandro Tricoli



Science & Technology Facilities Council  
Rutherford Appleton Laboratory

*Thanks to Stefano Rosati for the help on muon software!!*

CSC W/Z+jets Meeting, T&P Week, 6<sup>th</sup> June 2007, CERN

# Overview

- ❑ **Common ATHENA Ntuples** for physics analysis
  - ❑ Ntuples content
  - ❑ available Ntuples (Monika Wielers)
  
- ❑ **SpartyJet Ntuples:** providing multiple jet algorithms

# CSC W/Z+jets Common Ntuples

- As discussed in the last meeting, I agreed to produce common ntuples for the whole CSC W/Z+Jets group
  
- These are meant to be general purpose ntuples for physics studies
  - probably they are not the most suitable for specific performance studies (but let me know if only little extra info would be enough)
  - Idea is to keep them as light as possible, but with all the information the group might need for their physics analyses.
  - I received some requests from people in the group which I tried to fulfil
    - Let me know if any important information is missing (variables that you will definitely need!! Trying to keep the Ntuples as light as possible)
    - Get back to me as soon as possible!!
  
- A first version is ready for validation by the group:
  - check if distributions look reasonable to you
  - if you have any suspicion get back to me as soon as possible
  - A new updated version will come up soon after the feedback from the group

# Variables conventions and documentation

- All energy/momenta are in “GeV”

- Naming conventions:

- If special prefix `Gen_` used, i.e. “Gen\_xxx\_xxx”,

- means truth, i.e. `Gen_Jet_xxx_xxx`, `Gen_FinStatePart_xxx`, etc

- if not means detector level object, i.e. `Jet_xxx_xxx`, `Electron_xxx`, `Photon_xxx`

- prefix is object name, i.e. `Jet_`, `Electron_`, `Photon_` etc.

- suffix is the stored quantity, i.e. `_eta`, `_phi`, `_m`, `_et` etc.

- between prefix and suffix there might be additional information, such as algorithms used etc., i.e. `Jet_KT06_et`, `Jet_TopoCONE04_et`, etc.

- Extra documentation

- you can always refer to my source code on lxplus:

- `/afs/cern.ch/user/t/tricoli/scratch0/testarea/12.0.6/PhysicsAnalysis/AnalysisCommon/UserAnalysis`

- maybe a TWiki page for the whole CSC W/Z+Jets group can be useful

# Ntuple Content: Trigger Info

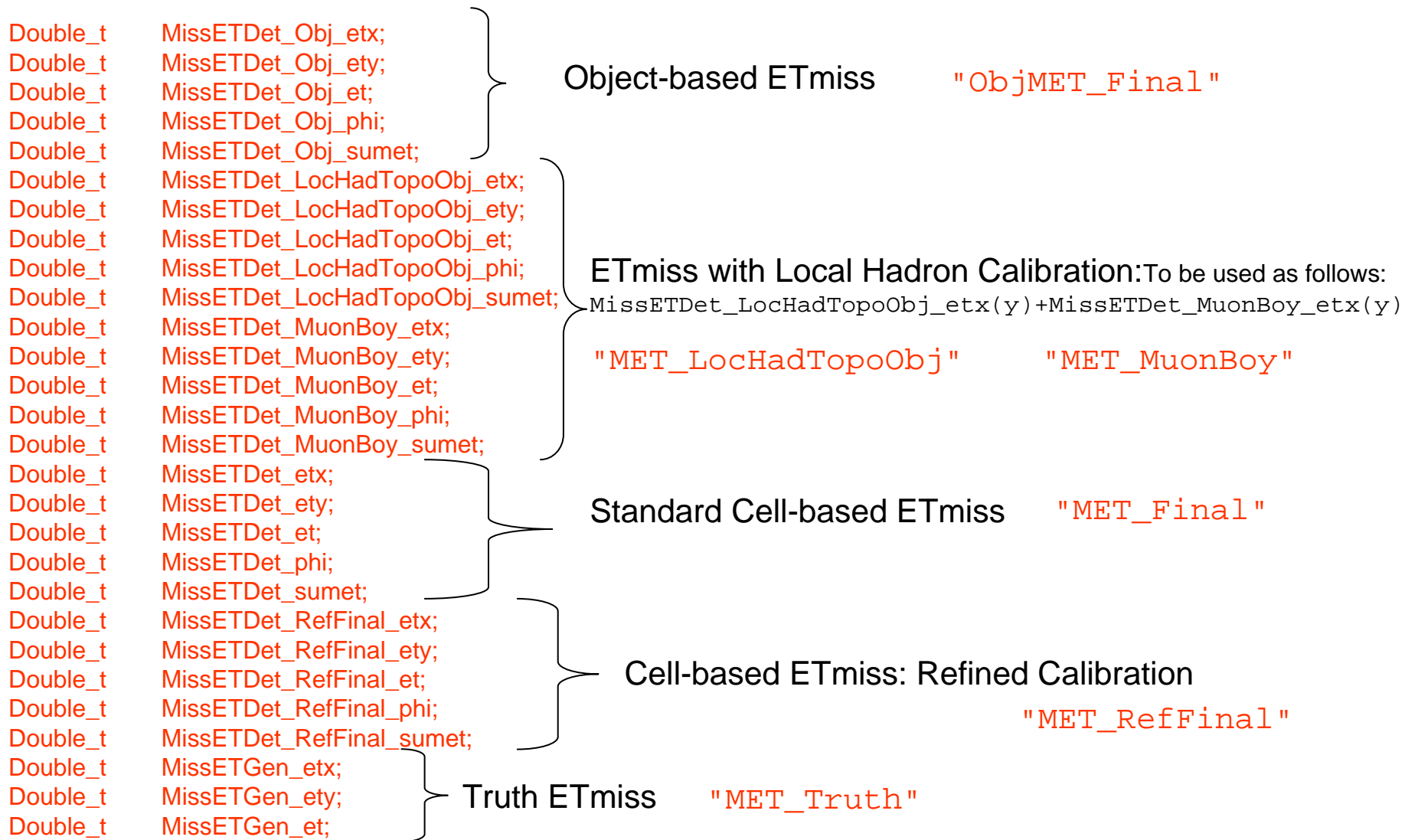
```
// Declaration of leave types
Int_t      RunNumber;
Int_t      EventNumber;
Char_t     StreamESD_ref[153];
Char_t     Stream1_ref[153];
Char_t     Token[153];
Int_t      Run;
Int_t      Event;
Int_t      Time;
Double_t   Weight;
Int_t      IEvent;
```

**Standard Run, Event info**

```
Bool_t     e25i_L1Dec;
Bool_t     e25i_L2Dec;
Bool_t     e25i_EFDec;
Bool_t     e15ie15i_L1Dec;
Bool_t     e15ie15i_L2Dec;
Bool_t     e15ie15i_EFDec;
Bool_t     e60_L1Dec;
Bool_t     e60_L2Dec;
Bool_t     e60_EFDec;
Bool_t     mu6_L1Dec;
Bool_t     mu61_L1Dec;
Bool_t     mu20_L1Dec;
Bool_t     mu6_L2Dec;
Bool_t     mu61_L2Dec;
Bool_t     mu20_L2Dec;
Bool_t     mu6_EFDec;
Bool_t     mu61_EFDec;
Bool_t     mu20_EFDec;
```

**Some Trigger menu decisions (L1,L2,EF)**

# Ntuple Content: Missing ET



# Ntuple Content: Electrons

```
vector<int>      *Electron_author;  
vector<double>  *Electron_etcone;  
vector<double>  *Electron_etcone20;  
vector<double>  *Electron_etcone30;  
vector<double>  *Electron_etcone40;  
vector<double>  *Electron_et37;  
vector<double>  *Electron_e233;  
vector<double>  *Electron_e237;  
vector<double>  *Electron_e277;  
vector<double>  *Electron_ex;  
vector<double>  *Electron_ey;  
vector<double>  *Electron_eta;  
vector<double>  *Electron_phi;  
vector<double>  *Electron_et;  
vector<double>  *Electron_eoverp;  
vector<double>  *Electron_isEM;  
vector<double>  *Electron_hasTrack;  
vector<double>  *Electron_px;  
vector<double>  *Electron_py;  
vector<double>  *Electron_pt;  
vector<double>  *Electron_d0toVtx;  
vector<double>  *Electron_z0toVtx;  
vector<double>  *Electron_d0errToVtx;  
vector<double>  *Electron_z0errToVtx;  
vector<double>  *Electron_hasCharge;  
vector<double>  *Electron_charge;  
Int_t          Electron_n;
```

} Impact parameters, d0 and z0:  
Not filled...investigating reasons!!

# Ntuple Content: Photons & Primary Vertex

```
vector<double> *Photon_ex;  
vector<double> *Photon_ey;  
vector<double> *Photon_ez;  
vector<double> *Photon_e;  
vector<double> *Photon_eta;  
vector<double> *Photon_phi;  
vector<double> *Photon_et;  
vector<double> *Photon_isEM;  
Int_t Photon_n;
```

```
vector<double> *Photon_cnvAngleMatch;  
vector<double> *Photon_cnvTrackMatch;  
vector<double> *Photon_cnvVtx_x;  
vector<double> *Photon_cnvVtx_y;  
vector<double> *Photon_cnvVtx_z;
```

==0 ??

Probably not properly filled in AOD

Not filled.

Probably info not available in AOD

```
vector<double> *PrimaryVertex_x;  
vector<double> *PrimaryVertex_y;  
vector<double> *PrimaryVertex_z;  
vector<double> *PrimaryVertex_fitChi2;  
vector<int> *PrimaryVertex_fitNDoF;  
vector<double> *PrimaryVertex_covXX;  
vector<double> *PrimaryVertex_covYY;  
vector<double> *PrimaryVertex_covZZ;  
vector<double> *PrimaryVertex_covXY;  
vector<double> *PrimaryVertex_covXZ;  
vector<double> *PrimaryVertex_covYZ;  
Int_t PrimaryVertex_n;
```

Not filled.

Probably info not available in AOD

**Covariant matrix**

"VxPrimaryCandidate"

Photons

Primary Vertices

# Ntuple Content: Muons

```
vector<int>    *MuonMuid_author;  
vector<double> *MuonMuid_charge;  
vector<double> *MuonMuid_px;  
vector<double> *MuonMuid_py;  
vector<double> *MuonMuid_pz;  
vector<double> *MuonMuid_pt;  
vector<double> *MuonMuid_p;  
vector<double> *MuonMuid_eta;  
vector<double> *MuonMuid_phi;  
vector<int>    *MuonMuid_isLowPt;  
vector<int>    *MuonMuid_isMediumPt; ← Not filled  
vector<int>    *MuonMuid_isHighPt;  
vector<double> *MuonMuid_fitChi2;  
vector<int>    *MuonMuid_fitNDoF;  
vector<double> *MuonMuid_matchChi2;  
vector<int>    *MuonMuid_matchNDoF;  
vector<int>    *MuonMuid_bestMatch;  
vector<double> *MuonMuid_d0toVtx;  
vector<double> *MuonMuid_z0toVtx;  
vector<double> *MuonMuid_d0errToVtx;  
vector<double> *MuonMuid_z0errToVtx;  
vector<double> *MuonMuid_etcone20;  
vector<double> *MuonMuid_etcone30;  
vector<double> *MuonMuid_etcone40;  
vector<double> *MuonMuid_nucone20;  
vector<double> *MuonMuid_nucone30;  
vector<double> *MuonMuid_nucone40;  
Int_t        MuonMuid_n;
```

d0 and z0 Not filled...  
investigating reasons!!

**Muid-Muons** and  
similarly for **Staco-muons**

"MuidMuonCollection"

"StacoMuonCollection"

# Ntuple Content: TopoClusters & TrackRecord

```
vector<int>    *CaloCal_TopoClus_inBarrel;  
vector<int>    *CaloCal_TopoClus_inEndcap;  
vector<double> *CaloCal_TopoClus_eta0;  
vector<double> *CaloCal_TopoClus_phi0;  
vector<double> *CaloCal_TopoClus_basicEnergy;  
vector<double> *CaloCal_TopoClus_eta;  
vector<double> *CaloCal_TopoClus_phi;  
vector<double> *CaloCal_TopoClus_m;  
vector<double> *CaloCal_TopoClus_e;  
Int_t        CaloCal_TopoClus_n;
```

==0, Not filled in AODs

"CaloCalTopoCluster"

**CaloCalTopoCluster:**  
Calibrated TopoClusters  
Used as input to *SpartyJet*  
(see later slides)

```
vector<double> *TrackRecord_MuonEntry_e;  
vector<double> *TrackRecord_MuonEntry_x;  
vector<double> *TrackRecord_MuonEntry_y;  
vector<double> *TrackRecord_MuonEntry_z;  
vector<double> *TrackRecord_MuonEntry_px;  
vector<double> *TrackRecord_MuonEntry_py;  
vector<double> *TrackRecord_MuonEntry_pz;  
vector<double> *TrackRecord_MuonEntry_pdg;  
vector<double> *TrackRecord_MuonEntry_barcode;  
vector<double> *TrackRecord_MuonEntry_time;  
Int_t          TrackRecord_MuonEntry_n;
```

TrackRecord  
"MuonEntryRecordFilter"

# Ntuple Content: Jets

```
vector<double> *Jet_KT06_ex;  
vector<double> *Jet_KT06_ey;  
vector<double> *Jet_KT06_eta;  
vector<double> *Jet_KT06_phi;  
vector<double> *Jet_KT06_et;  
vector<double> *Jet_KT06_e;  
vector<int> *Jet_KT06_btag_ntag;  
vector<double> *Jet_KT06_btag_w_ip2d;  
vector<double> *Jet_KT06_btag_w_ip3d;  
vector<double> *Jet_KT06_btag_w_sv1;  
vector<double> *Jet_KT06_btag_w_sv2;  
vector<double> *Jet_KT06_btag_w_cmb;  
vector<double> *Jet_KT06_btag_w_lf2d;  
vector<double> *Jet_KT06_btag_w_svbu;  
vector<double> *Jet_KT06_btag_w_lhsig;  
Int_t Jet_KT06_n;
```

B-tagging algorithms

**KT D=0.6**

***Calo Tower Jets***

Similarly for

**KT04, CONE04 CONE07**

"xxxTowerParticleJets"

**KT D=0.6**

***TopoCluster Jets***

Similarly for

**TopoKT04,**

**TopoCONE04**

**TopoCONE07**

"xxxTopoParticleJets"

B-tagging algorithms

```
vector<double> *Jet_TopoKT06_ex;  
vector<double> *Jet_TopoKT06_ey;  
vector<double> *Jet_TopoKT06_eta;  
vector<double> *Jet_TopoKT06_phi;  
vector<double> *Jet_TopoKT06_et;  
vector<double> *Jet_TopoKT06_e;  
Int_t Jet_TopoKT06_n;  
vector<int> *Jet_TopoKT06_btag_ntag;  
vector<double> *Jet_TopoKT06_btag_w_ip2d;  
vector<double> *Jet_TopoKT06_btag_w_ip3d;  
vector<double> *Jet_TopoKT06_btag_w_sv1;  
vector<double> *Jet_TopoKT06_btag_w_sv2;  
vector<double> *Jet_TopoKT06_btag_w_cmb;  
vector<double> *Jet_TopoKT06_btag_w_lf2d;  
vector<double> *Jet_TopoKT06_btag_w_svbu;  
vector<double> *Jet_TopoKT06_btag_w_lhsig;
```

# Ntuple Content: Truth Jets

```
vector<double> *Gen_Jet_CONE04_eta;  
vector<double> *Gen_Jet_CONE04_phi;  
vector<double> *Gen_Jet_CONE04_et;  
vector<double> *Gen_Jet_CONE04_e;  
vector<int>    *Gen_Jet_CONE04_flav;  
Int_t         Gen_Jet_CONE04_n;
```

```
vector<double> *Gen_Jet_CONE07_eta;  
vector<double> *Gen_Jet_CONE07_phi;  
vector<double> *Gen_Jet_CONE07_et;  
vector<double> *Gen_Jet_CONE07_e;  
vector<int>    *Gen_Jet_CONE07_flav;  
Int_t         Gen_Jet_CONE07_n;
```

Jet flavour Not filled...  
investigating

**CONE R=0.4, 0.7**  
**Truth Jets**  
Similarly for  
**KT04, KT06**

"xxxTruthParticleJets"

# Ntuple Content: Truth Particles (1)

```
vector<double> *Gen_FinStatePart_eta;  
vector<double> *Gen_FinStatePart_phi;  
vector<double> *Gen_FinStatePart_pt;  
vector<double> *Gen_FinStatePart_m;  
vector<double> *Gen_FinStatePart_pdgId;  
vector<double> *Gen_FinStatePart_status;  
vector<double> *Gen_FinStatePart_barcode;  
Int_t          Gen_FinStatePart_n;
```

All Final State and Stable  
truth particle (status==1)

```
vector<double> *Gen_HardProclnPart_x; ←  
vector<double> *Gen_HardProclnPart_pdgId;  
vector<double> *Gen_HardProclnPart_barcode;  
vector<double> *Gen_HardProclnPart_status;  
Int_t          Gen_HardProclnPart_n;
```

Momentum fraction

The two incoming partons  
Initiating the *hard process*  
Before ISR-Parton Shower

Double\_t

q2;

Alpgen event energy scale  $Q^{**2} = M_W^{**2} + P_T^{W**2}$

# Ntuple Content: Truth Particles (2)

```
vector<double> *Gen_HardProcOutPart_pdgId;  
vector<double> *Gen_HardProcOutPart_status;  
vector<double> *Gen_HardProcOutPart_barcode;  
vector<double> *Gen_HardProcOutPart_eta;  
vector<double> *Gen_HardProcOutPart_pt;  
vector<double> *Gen_HardProcOutPart_phi;  
vector<double> *Gen_HardProcOutPart_px;  
vector<double> *Gen_HardProcOutPart_py;  
vector<double> *Gen_HardProcOutPart_pz;  
vector<double> *Gen_HardProcOutPart_e;  
Int_t          Gen_HardProcOutPart_n;
```

All outgoing partons  
from the *hard process*  
(i.e *W/Z and q/g*)  
Before ISR-Parton Shower

```
vector<double> *Gen_HardProcOutPartPostISR_pdgId;  
vector<double> *Gen_HardProcOutPartPostISR_status;  
vector<double> *Gen_HardProcOutPartPostISR_barcode;  
vector<double> *Gen_HardProcOutPartPostISR_eta;  
vector<double> *Gen_HardProcOutPartPostISR_pt;  
vector<double> *Gen_HardProcOutPartPostISR_phi;  
vector<double> *Gen_HardProcOutPartPostISR_px;  
vector<double> *Gen_HardProcOutPartPostISR_py;  
vector<double> *Gen_HardProcOutPartPostISR_pz;  
vector<double> *Gen_HardProcOutPartPostISR_e;  
Int_t          Gen_HardProcOutPartPostISR_n;
```

All outgoing partons  
from the *hard process*  
(i.e *W/Z and q/g*)  
After ISR-Parton Shower

# Ntuple Content: Truth Particles (3)

```
vector<double> *Gen_WZ_pdgId;  
vector<double> *Gen_WZ_status;  
vector<double> *Gen_WZ_barcode;  
vector<double> *Gen_WZ_m;  
vector<double> *Gen_WZ_pt;  
vector<double> *Gen_WZ_eta;  
vector<double> *Gen_WZ_phi;  
vector<double> *Gen_WZ_e;  
Int_t      Gen_WZ_n;
```

All W and Z particles in event record  
(beware same particle can appear more than once  
in the event record)

```
vector<double> *Gen_WZLept_pdgId;  
vector<double> *Gen_WZLept_status;  
vector<double> *Gen_WZLept_barcode;  
vector<double> *Gen_WZLept_eta;  
vector<double> *Gen_WZLept_pt;  
vector<double> *Gen_WZLept_phi;  
vector<double> *Gen_WZLept_px;  
vector<double> *Gen_WZLept_py;  
vector<double> *Gen_WZLept_pz;  
vector<double> *Gen_WZLept_e;  
Int_t      Gen_WZLept_n;
```

$e^\pm, \mu^\pm, \nu_e^\pm, \nu_\mu^\pm$  from W or Z decays  
(beware not all are final state, i.e. status==1,  
some of them are before the ISR-Parton Shower)

# Ntuple Generation (M. Wieliers)

- ❑ New AODs available with correct G4 range cut of 30 microns
- ❑ Ntuple Generation on the GRID by PANDA thanks to Monika
  - ❑ we have so far  $W \rightarrow \text{enu} + \text{Jets}$  and  $W \rightarrow \text{munu} + \text{Jets}$  test Ntuples
  - ❑ See Monika's talk for details and how to retrieve the Ntuples from the GRID

# SpartyJet

- ❑ For each ATHENA Common Ntuple,  
I will also provide additional SpartyJet Ntuples
  
- ❑ What is SpartyJet?
  - ❑ It is a generic routine for jet analysis
    - ❑ *Authors: Pierre-Antoine Delsart, Kurtis Geerlings, Joey Huston*
  - ❑ See two presentations given on tuesday in the Jet Group  
<http://indico.cern.ch/materialDisplay.py?contribId=96&sessionId=3&materialId=slides&confId=16155>
  
  - ❑ Briefly: SpartyJet allows the user to
    - ❑ Analyse and compare many more jet algorithms than in the AOD
      - ❑ tuning their parameters at will
    - ❑ Test and develop new algorithms
    - ❑ Works in ROOT environment

# Reasons to use SpartyJet

- It is Important to compare different jet algorithms
  - for some events, there is ambiguity about the assignment of clusters/particles to the jet and the jet algorithm must make decisions that impact precision measurements
  - full understanding of complicated events may require the use of multiple algorithms (especially at parton level)
- In the AOD we have **Cone R=0.4, 0.7, KT D=0.4, 0.6** reconstructed with *Calorimeter Towers, Topological Clusters (and Truth)*
- With **SpartyJet** we can compare many more jet algorithms, i.e. JetClu, Midpoint, SISCone etc. and also user defined algorithms

# SpartyJet Ntuples

- ❑ I feed the **Athena Common Ntuples** (extracted from AODs) into **SpartyJet**
  - ❑ **Truth:** final state, stable and interacting *Truth Particles* ( "*SpC1MC*" )
    - ❑ status==1
    - ❑ isGenInteracting ==true
    - ❑ excluding muons
  - ❑ **Reconstructed Clusters:** calibrated *TopoClusters*
    - ❑ "CaloCalTopoCluster"
      - ❑ they are ~well calibrated (e/h, dead material) in AOD
      - ❑ total jet energy scale can still be wrong at the level of ~5%
        - ❑ because no jet energy scale corrections are applied
      - ❑ but with the latest cluster calibration a flat response to QCD jets is expected.
- ❑ For each ATHENA Common Ntuple I will provide 2 additional Ntuples
  - ❑ One SpartyJet Ntuple with additional jets reconstructed from **Calibrated TopoClusters**
  - ❑ One SpartyJet Ntuple with additional jets reconstructed from **Truth Particles**
- ❑ These additional Ntuples can be easily merged together and with the Athena Common Ntuples (via `addFriend` ROOT method)
  - ❑ example code on how to read them together will be provided.

# Common Ntuples

- ATHENA Common Ntuples are ready for validation by the group
  - Please get back to me soon, if you find any problem!!
  - if you have any further request let me know now please
- SpartyJet Ntuples will be also provided with extra jet algorithms
- We are planning to produce Ntuples for
  - W/Z + Jets in electron and muon channels
  - backgrounds?
- Writing, Running and Maintaining the Ntuple code can be rather time consuming so we might need help from the group.