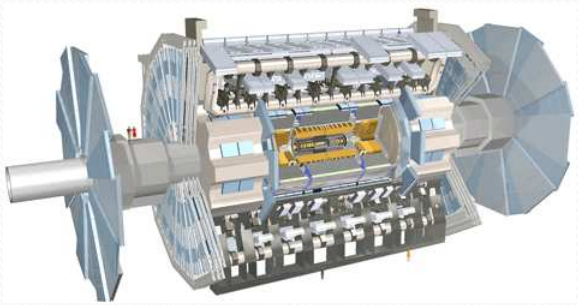
# *Evaluation of the jet calibration performance with the TruthToolCode software*

**Zineb Idrissi, Caterina Doglioni, Rajaà Cherkaoui El Moursli**

**Faculty of sciences Rabat – ATLAS Experiment**



# Outline

**"How well can we reconstruct jets? "**

**This Talk:** The performance of the jet reconstruction is evaluated with the TruthToolCode package.

- Introduction
- TruthToolCode package
- Manipulation of the TruthToolCode package
- Example plots
- Running the TruthToolCode in the grid
- Conclusion

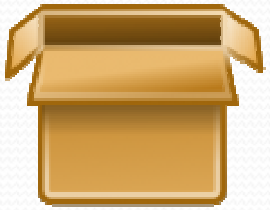
# Introduction

➔Work on the TruthToolCode within the JetPerformance Software package. This tool has been developed to make performance plots of the jet energy response in Monte Carlo events.

## Work Envirenment : The RootCore Framework

- Standalone makefile structure
- Depends on ROOT
- TheTruthToolCode is one of its packages.

# The TruthToolCode package



*JetPerformance2011*

- D3PDMakerReaderSkeleton
- TruthToolPrunSubmission
- CloseByJetsStudies
- **TruthToolCode**
- …
- TruthToolCodeTop
- GammaJetCode

*Offer a common infrastructure to estimate the performance of the jet reconstruction*

❑The performance is evaluated in term of jet response in kinematic and spatial bins (eta, pT, NPV)

**Cmt :** Contains the MakeFile for compilation.

**Root :** Where all of our C++ source files go.

**Run :** It contains the run.cxx macros and the onfigFileJES2011.config.

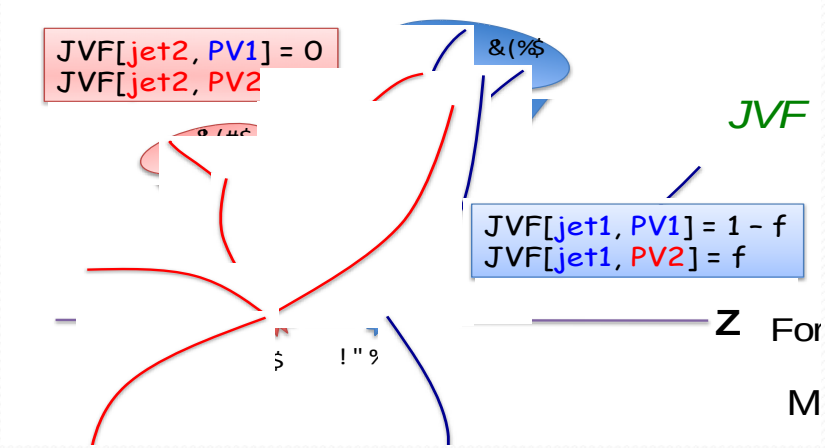
**TruthToolCode :** Where all of our C++ header files are located.

# Jet-Vertex-Fraction (JVF)

The jet-vertex fraction is an important parameter for jets studies because it lets us understand if a jet is coming from a pile-up vertex.

- The jet-vertex-fraction (JVF) is the sum of the pT of all matched tracks coming from a given vertex divided by the total track PT for tracks matched to the jet coming from all vertices.

$$JVF(jet_i, vtx_j) = \frac{\sum_k P_T(trk_k^{jeti}, vtx_j)}{\sum_n \sum_p P_T(trk_p^{jeti}, vtx_n)}$$




- JVF is defined for each jet with respect to the primary vertex (PV). So this means that:

✓ JVF=1 for jets coming from the hard scattering
✓ JVF=0 for pile-up jets
✓ JVF=-1 if there are no matched tracks

# Integration of JVF variable in the TruthToolCode

1

*TruthToolCode.cxx/h*

In these files we tried to make the code more general and to substitute the NPV variable by any other variable. In this case we used the JVF variable which can be accessed starting from the D3PD

2

*configFileJES2011.config*

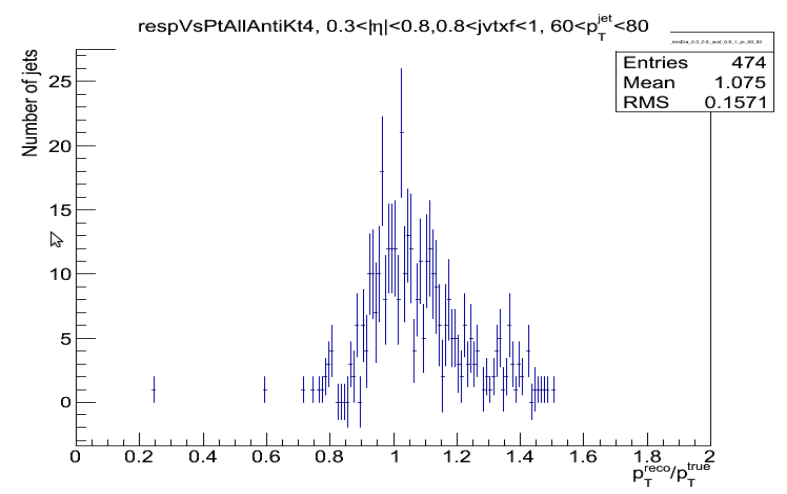
In this file we configure the JVF binning.

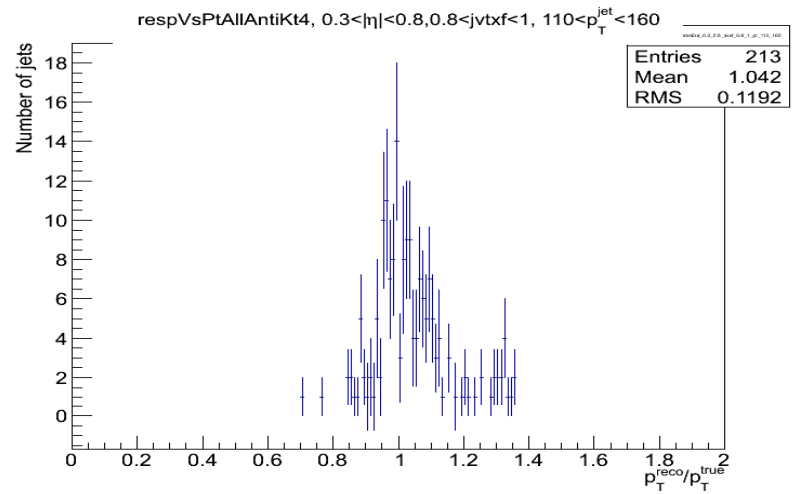
3

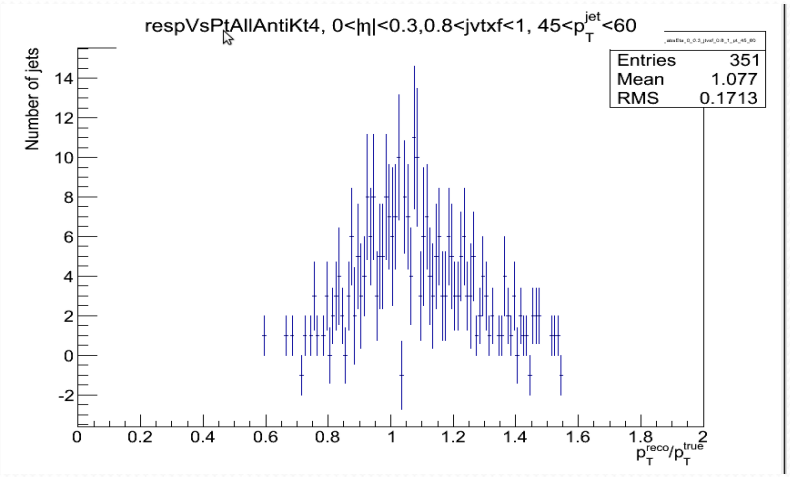
*JetPerf4DHistogram.cxx/h*

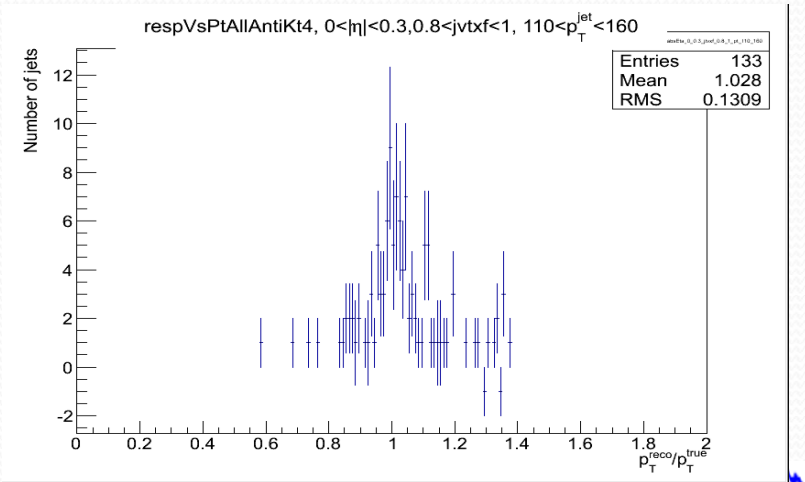
Here we configure the changes in the name and title of the histograms.

# Some Results



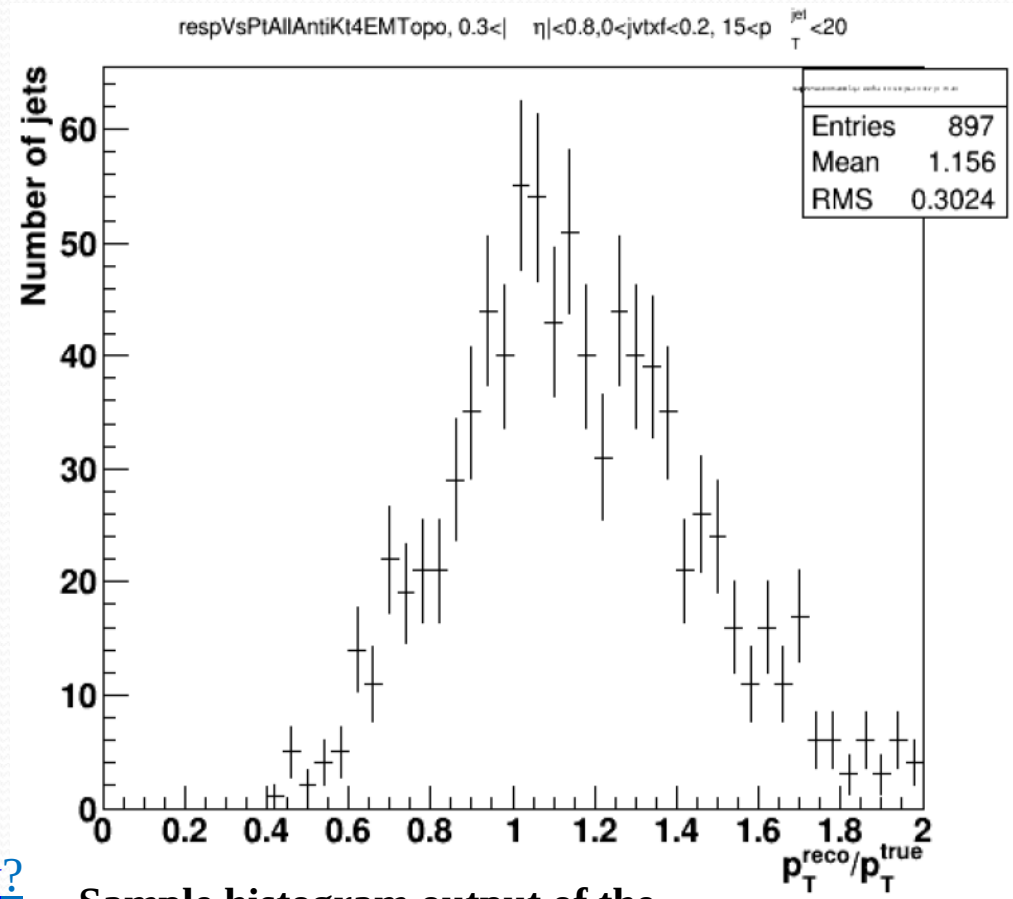
# Running the TruthToolCode with the new variable in the grid

- We used the scripts in the TruthToolPrunSubmission package to submit our jobs to the grid.

- We have run on the Monte Carlo dataset [mc11_7TeV.105013.J4_pythia_jetjet.merg NTUP_JETMET.e815_s1273_s1274_r2923 2900_p832/]

- We have retrieved the output files using the dq2 commands. As we have many jobs we have to merge their output with the hadd command in ROOT.

http://panda.cern.ch/server/pandamon/query?ui=user&name=Zineb%20Idrissi&force=yes8




**Sample histogram output of the TruthToolCode**

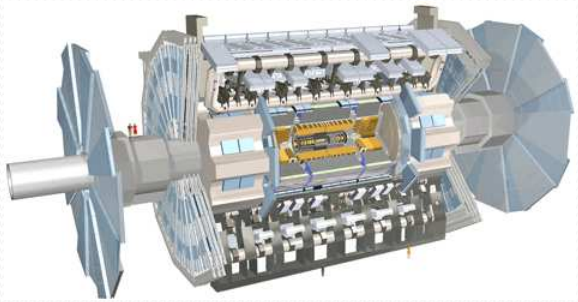
# Conclusion

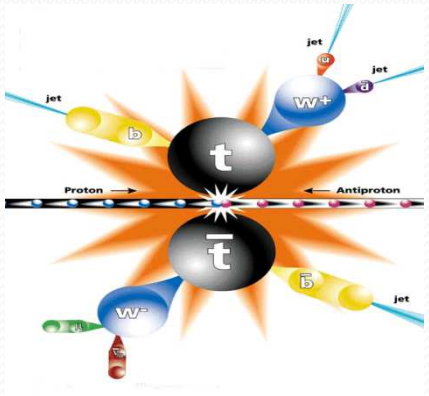
- **Measuring jets is an important step of the LHC program.**
- In this work we modified the TruthToolCode to include the JVF variable and prepare the code to be more generic
- **The next step of this work will be further modification of the code so that we can replace the JVF variable with any variable chosen by the user. In this way the code will be more general and we can test the jet performance as a function of many different jet properties.**

# *ATLAS Framework for Top Physics Analysis*

**Zineb IDRISSI, Nacim HADDAD, Farida Fassi IMLAHI, Rajaà Cherkaoui El Moursli**

**Faculty of sciences Rabat – IFIC Valencia Spain**





http://indico.ific.uv.es/indico/conferenceDisplay.py?showSession=all&showDate=all&view=standard&fr=no&confId=600

**Another view of the RootCore framework**

# TopRootCore

## Motivation

How can we study various physical phenomena in the same package? TopRootCore. (Top Boosted && Top Resolved)

Solution !!

- ❑ Create our own sub-packages which operate independentl
- ❑ Adapt its structures to the philosophy of TopRootCore

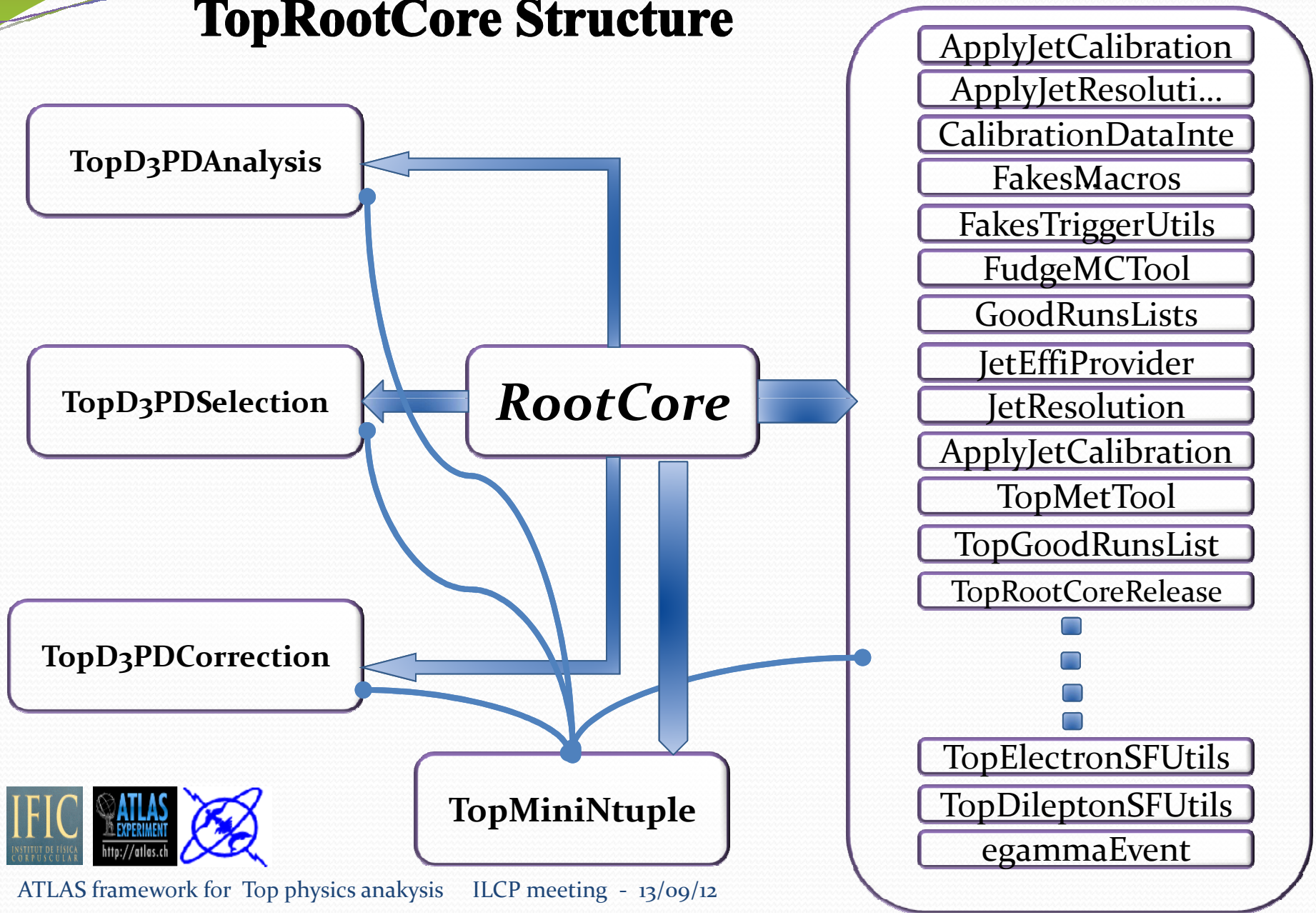
# Back-up

Inside the output file, there are many histograms, one per bin of pT, eta, JVF. Histograms are called like:

**respVsPtAllAntiKt4EMTopo_absEta_n_l_jvtxf_p_q_pt_x_y**

- **resp** = this histogram contains the response distribution for jets in this bin
- **VsPt** = this histogram is binned in pT
- **All** = this histogram contains all jets in the event
- **AntiKt4EMTopo** = jets in this histogram have been reconstructed using the Antikt algorithm and calibrated with the EM+JES calibration.
- **absEta_0.3_0.8** = jets in this histogram have pseudorapidities of 0.3 to 0.8 in absolute value.
- **jvtxf_0_0.2** = jets in this histogram have values of JVF from 0 to 0.2
- **pt_15_20** = jets in this histogram have a pT of 20 to 30 GeV

# TopRootCore Structure



# TopRootCore package Structure


**MyPackage** : contains the C++ *header file*

**cmt** : Contains in principle the "MakeFile » file

**Root** : Contains the source files and the « LinkDef.h » *(file for the construction of dictionaries)*

**util** : Contains the executable files

**scripts** : Package contains the scripts provided by default

**data** : Contains data files that are distributed in the package

+

**When executing, a set of directories are created**

**obj** **bin** **StandAlone**